ННІ «Інститут геології»
Київського національного університету імені Тараса Шевченка
Кафедра геоінформатики

Демидов В.К., Зацерковний В.І., Віршило І.В., Тішаєв І.В

ПРОГРАМУВАННЯ У СЕРЕДОВИЩІ ГІС

Навчальний посібник.
Редакція 1

Київ 2023

Навчальний посібник / В.К. Демидов, В.І. Зацерковний, І.В. Віршило, І.В Тішаєв – електронне видання, 2023. – 112 с.

Автори:
Демидов Всеволод Кирилович, кандидат фізико-математичних наук, доцент,
Зацерковний Віталій Іванович, доктор технічних наук, професор,
Віршило Іван Вікторович, кандидат геологічних наук, доцент,
Тішаєв Іван Васильович, кандидат фізико-математичних наук, доцент

Рецензенти:

Ляшенко Дмитро Олексійович, професор, доктор географічних наук
Кошляков Олексій Євгенович, професор, доктор геол. наук
Єгорченков Олексій Володимирович, професор, доктор технічних наук

Рекомендовано до публікації вченою радою ННІ «Інститут геології» протокол № 10 від 19 квітня 2023 р.

Приводяться основні принципи створення програмних рішень в середовищі ГІС. Розглядаються основні підходи до написання сценаріїв за допомогою ArcPy. Висвітлені базові положення щодо вибору інструментів для доступу та аналізу даних, візуалізації та представлення результатів. Розглядаються спеціальні модулі Python, та інші програмні рішення для роботи з геопросторовими даними. Основний акцент робиться на засвоєння здобувачами необхідних методичних та методологічних знань і практичних навичок програмування з метою розв'язання наукових і прикладних задач.

Розраховано на студентів бакалаврату та викладачів за спеціальністю 193 - Геодезія та землеустрій та 103 – Науки про Землю. Вимагає навичок програмування та знання ГІС.

ЗМІСТ

1. ВСТУП

Використання інструментів геообробки для роботи з геопросторовими даними в різних програмних середовищах, в тому числі в таких спеціальних як ArcGis чи QGis, є найважливішою частиною фактичної роботи з подібними даними. До інструментів геообробки відноситься велика кількість інструментів для вирішення типових ГІС-задач, починаючи від простої побудови буфера та накладання полігонів до комплексного регресійного аналізу та класифікації зображень. Набір конкретних інструментів буде залежати від конкретного середовища, доступ до таких інструментів також буде відрізнятися, але концептуальні принципи роботи з ними будуть подібними.

Однак вивчення інструментів геообробки дає можливість їх застосувати не тільки для однієї локальної задачі, наприклад створити буфер навколо об'єкта, але і для більш ґрунтовного аналізу з побудовою проекту. Отже, багато завдань, які необхідно автоматизувати, можуть бути звичайними, наприклад, перетворити групи даних з одного формату в інший. Однак геообробка дозволяє складати ланцюжки (набори) інструментів, коли фактично вихідні дані попереднього інструменту є вхідними для наступного. Це фактично дозволяє автоматизувати роботу та вирішити складні проблеми.

Для автоматизації роботи та вирішення складних задач можна використовувати різні підходи. Це може бути використання конкретної мови програмування і середовища з використанням спеціальних модулів. Наприклад Python з модулями Geopandas, Gdal, Numpy, Geoserver, з середовищем розробки PyCharm та візуалізацією на Folium. Або інший стек технологічних рішень, варіативність якого може бути значною. З іншого боку це може бути і вбудовані в спеціальні ГІС середовища інструменти, наприклад ModelBuilder в ArcGIS або модуль ArcPy.

Перелік технологічних рішень для автоматизації роботи з геопросторовими даними є достатньо великим, а з плином часу він тільки збільшується.

В даному навчальному посібнику розглядається тільки невелика частина з усього можливого переліку подібних рішень. Робиться акцент саме на ті рішення, які передбачають використання навичок програмування та створення власних проектів та окремих скриптів для вирішення прикладних задач в області ГІС.

Ймовірно, цей курс буде складнішим, ніж базові курси з основ ГІС. Крім того, для програмування в середовищі ГІС необхідно мати базові навички з основ програмування на Python. В тому числі знати базові типи даних в Python, вміти використовувати циклічні та умовні конструкції, писати функції та організовувати власні класи, знати як приєднувати модулі до власного проекту.

При розробці даного посібника було використано матеріали з різних джерел, в тому числі авторські. Однак значна частина матеріалів була використана з підручника Zandbergen Paul A. Python Scripting for ArcGIS, офіційної документації від ESRI по модулю ArcPy та курсу GEOG 485: ГІС Programming and Software Development від College of Earth and Mineral Sciences, The Pennsylvania State University. Автор висловлює вдячність правовласникам даних ресурсів, та радить використовувати їх як першоджерела при вивченні курсу “Програмування у середовищі ГІС”.

При вивченні даного посібника Ви побачите матеріал, який дуже схожий на те, що міститься в зазначених ресурсах. Це не обов’язково погано, адже коли Ви вивчаєте такий предмет, як програмування, може бути корисно пояснити ту саму концепцію з різних сторін.

Крім того, варто розуміти, що друковані матеріали достатньо швидко застарівають, адже технології розвиваються дуже стрімко. Тому варто використовувати даний посібник як концептуальний, де робиться акцент на конкретні інструменти і принципи їх використання. Конкретний же код може відрізнятися, залежно від версій програмного забезпечення, яку Ви використовуєте.

Автори також зауважують, що для вивчення даного курсу може бути корисним використання офіційної документації та курсів від компаній розробників. В тому числі курси Python for Everyone (від Esri Academy) або ArcPy Essentials (від Esri Academy) для більш глибокого вивчення матеріалу. Для даної редакції посібника буде використано матеріали адаптовані під версії ArcGIS Desctop 10.5 – 10.8, однак приклади можуть бути легко адаптовані під ArcGIS Pro.

В даному посібнику викладені лише базові принципи програмування у середовищі ГІС, а ідеї охоплюються різноманітні концепції. Тому, можливо, це здається непосильним для засвоєння. Однак, посібник проілюстровано великою кількістю прикладів, які допоможуть зрозуміти основний матеріал. Можливо, вам буде корисно швидко переглянути посібник, а потім більш детально проробити наведені приклади. Приклади, та матеріали до коду, надаються окремим файлом для здобувачів.

Варто пам’ятати, що очі бояться, а руки роблять. Цілком ймовірно, що Ви десь «застрягнете» під час курсу та вивчення даного посібника, тому перш ніж безнадійно розчаруватися, зверніться за допомогою до викладача, інших здобувачів або професійних форумів!

2. БАЗОВІ ПРИНЦИПИ ВИКОРИСТАННЯ MODEL BUILDER

2.1. Нащо потрібна автоматизація

Географічні інформаційні системи (ГІС) можуть маніпулювати та аналізувати масиви просторових даних з метою аналізу геопросторових даних різного походження та областей застосування, в тому числі даних моніторингових спостережень, геологічні дані, космічні знімки та інші. Фахівці в сфері ГІС-аналітики виконують різні операцій над даними, щоб зробити їх корисними для вирішення конкретної проблеми. Це включає відсікання, повторне перепроеціювання, буферизацію, злиття, мозаїку, вилучення підмножин даних і багато інших операцій. Для спеціального програмного забезпечення ArcGIS ці операції відомі як геообробка, і вони виконуються за допомогою наборів інструментів, які сконцентровані в ArcToolbox (*рис. 2.1*).

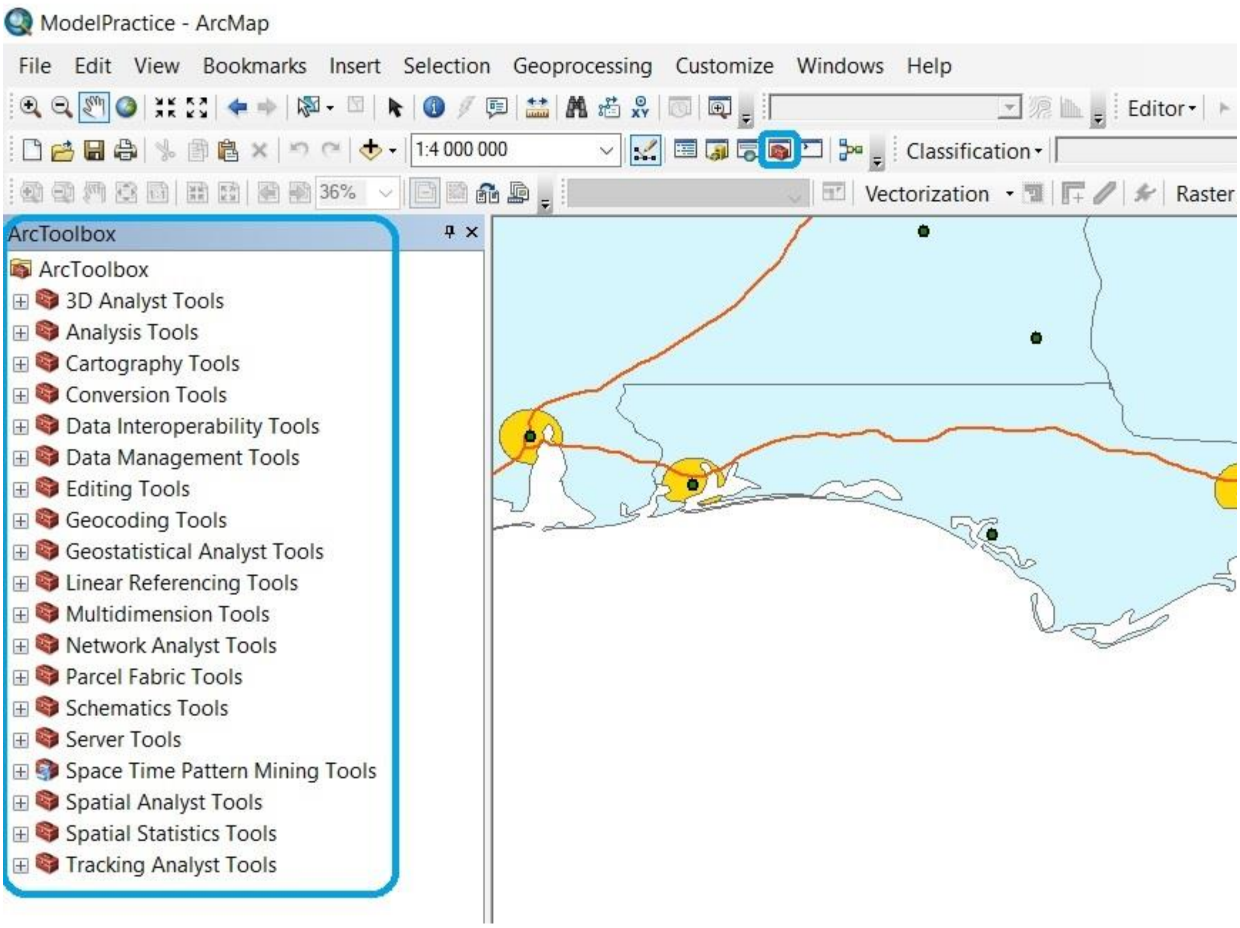


Рис.2.1 Набір інструментів геообробки, які сконцентровані в ArcToolbox

Для організації обробки та аналізу даних в ГІС необхідно обрати інструменти, які найбільше підходять для роботи з вашими даними. Ви вибираєте необхідні інструменти та запускаєте їх у належному порядку, щоб зробити відповідний аналіз і зробити свій власний проект.

Припустимо, Ви плануєте маршрут для геологічного вивчення місцевості, або туристичний. Ви можете використовувати один інструмент для вибору ділянок уздовж головних шляхів, для найшвидшого пересування, інший інструмент для вибору ділянок для дослідження, які розташовані не далі як 50 м. від доріг та інші інструменти для інших критеріїв вибору. Якби цей процес вибору обмежувався невеликою площею та був одноразовим, ймовірно, було б сенс виконувати роботу вручну.

Однак припустимо, що Ви відповідаєте за проведення такого ж аналізу для кількох ділянок і щоденно. Оскільки цей сценарій передбачає використання однакової послідовності інструментів для різних ділянок, він добре підходить для автоматизації. Є кілька основних переваг автоматизації таких завдань:

- Автоматизація полегшує роботу. Після того, як Ви автоматизуєте процес, вам не доведеться докладати стільки зусиль, запам'ятовуючи, які інструменти використовувати або правильну послідовність, у якій їх потрібно запускати.
- Автоматизація робить роботу швидше. Комп'ютер може відкривати та виконувати інструменти послідовно набагато швидше, ніж Ви можете виконати те саме завдання, наводячи "мишку" і клацаючи.

Автоматизація робить роботу більш точною. Кожного разу, коли Ви виконуєте завдання вручну на комп'ютері, є ймовірність помилки. Шанс помножується на кількість і складність кроків у вашому аналізі, а також на втому, яка виникає від багаторазового повторення завдання. Навпаки, коли автоматизоване завдання налаштовано, комп'ютеру можна довіряти виконувати ту саму послідовність кроків щоразу протягом потенційно нескінченної кількості циклів.

Платформа ArcGIS надає кілька способів для користувачів автоматизувати свої завдання геообробки. Ці параметри відрізняються кількістю навичок, необхідних для створення автоматизованого рішення, і діапазоном сценаріїв, які кожен може розглянути. Наведемо ці варіанти автоматизації, від такого, що вимагає найменших навичок програмування до такого, ще програмувати необхідно багато.

Перший варіант — побудувати модель за допомогою **ModelBuilder**. ModelBuilder — це інтерактивна програма, яка дозволяє користувачеві «об'єднувати» інструменти разом, використовуючи вихідні дані одного інструменту як вхідні дані іншого. Можливо, найпривабливішою особливістю ModelBuilder є те, що користувачі можуть автоматизувати досить складні робочі процеси ГІС без, фактично, необхідності безпосереднього програмування.

Деякі завдання автоматизації вимагають більшої гнучкості, ніж пропонує ModelBuilder, і для цих сценаріїв рекомендується писати короткі комп'ютерні програми або сценарії (скрипти). Основна частина даного посібника присвячена саме написанню сценаріїв.

Отже, *другий варіант* — написання скритпів. Сценарій зазвичай виконує певну послідовну процедуру кроків. У рамках сценарію Ви можете запускати інструменти ГІС окремо або з'єднувати їх разом. Ви можете вставити умовну логіку у свій сценарій для обробки випадків, коли слід запустити різні інструменти залежно від результату попередньої операції. Ви також можете включити ітерацію або цикли в сценарій, щоб повторити одну дію стільки разів, скільки необхідно для виконання завдання. Ви можете створити власні класи для написання складних шаблонів для різнопланових задач.

Існують спеціальні мови сценаріїв для написання сценаріїв, включаючи Python, EcmaScript (JavaScript), Ruby та інші. Ці мови є відносно нескладними для початківців для написання простого коду і мають низький рівень "входу" для розробників.

Хоча ArcGIS підтримує різні мови сценаріїв для роботи зі своїми інструментами, Esri окремо виділяє Python у своїй документації та включає Python з інсталяціями ArcGIS Desktop та Pro. В рамках даного посібника ми будемо розглядати виключно Python саме з цієї причини, а також з того факту, що Python можна використовувати для багатьох інших завдань при роботі з файлами та даними за межами ArcGIS. Ми не будемо розглядати основи мови Python, т.я. базові навички програмування в Python є необхідним критерієм для використання даного посібника.

Третім варіантом автоматизації, доступним на платформі ArcGIS, є *ArcGIS API* (Інтерфейс програмного забезпечення) для Python. Це середовище, в якому сценарії Python краще інтегровані з хмарними та серверними технологіями Esri (ArcGIS Online, Portal for ArcGIS, ArcGIS Enterprise). Код, написаний для взаємодії з API Python, часто пишеться в «ноутбук-середовищі», наприклад Jupyter Notebook. У середовищі Jupyter код можна виконувати поетапно з проміжними результатами, які відображаються між операторами Python.

Для завдань геообробки, які потребують підтримки взаємодії користувача з картою або іншими елементами інтерфейсу користувача, ArcGIS SDK (набір програмного забезпечення) пропонує можливість додавати спеціальні інструменти до інтерфейсу ArcGIS. ArcGIS SDK вимагає програмування на платформі .NET з використанням мови .NET, наприклад Visual Basic .NET або C#. Робота з об'єктною моделлю SDK забезпечує більшу гнучкість щодо того, що можна побудувати, ніж написання сценаріїв Python. Однак використання ArcGIS SDK вимагає вищого рівня навичок програмування в порівнянні з написанням сценаріїв в Python.

Нарешті, розробники, які хочуть створювати власні ГІС-додатки, які зазвичай зосереджені на наданні набагато більш вузьких функцій, ніж універсальний ArcGIS, можуть розробляти програми за допомогою ArcGIS Runtime SDK. Пакети SDK Runtime дозволяють створювати програми для настільних комп'ютерів Windows, Mac або Linux, а також для мобільних пристроїв iOS і Android, знову ж таки, що вимагає більшого рівня зусиль, ніж звичайний сценарій геообробки на Python.

2.2. Що таке ModelBuilder

ModelBuilder – це програма, яка використовується для створення, редагування та керування моделями. Моделі – це робочі процеси, які з'єднані один з одним у послідовності інструментів геообробки, передаючи вихідні дані одного інструмента до іншого інструменту як вхідні дані (*рис.2.2*). ModelBuilder також можна розглядати як візуальну мову програмування для побудови робочих потоків.

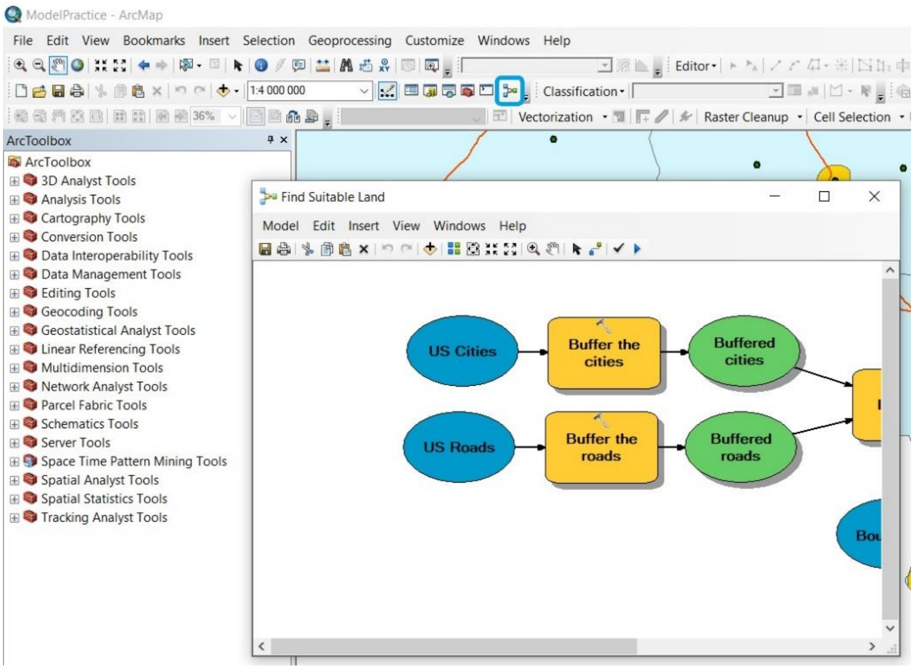


Рис.2.2 Приклад моделі, організованої в ModelBuilder з використанням інструментів геообробки, які сконцентровані в ArcToolbox

Ви можете перетягнути інструменти з панелі «Каталог» у модель і «з’єднати» їх, вказавши порядок, у якому вони мають працювати.

Хоча це в першу чергу посібник з програмування, ми присвятимо деякий час ModelBuilder з двох причин:

- ModelBuilder — це гарне середовище для вивчення інструментів ArcGIS. Вивчення того, як використовувати параметри інструментів для введення та виведення даних, а також візуального розуміння того, як працює моделювання ГІС. Коли Ви почнете використовувати Python, у вас не буде такої візуальної допомоги, щоб побачити, як підключені інструменти, які Ви використовуєте, але Ви все одно можете намалювати свою модель на папері так само, як ми подивимося в ModelBuilder.
- ModelBuilder може часто зменшувати кількість власного коду в Python, який вам потрібно зробити. Якщо ваша задача в ГІС не вимагає розширеної умовної та ітераційної логіки, Ви можете виконати свою роботу в ModelBuilder без написання сценарію. ModelBuilder також дозволяє експортувати будь-яку модель як код Python, тому, якщо Ви застрягли в реалізації деяких інструментів у сценарії, може бути корисно створити просту робочу модель у ModelBuilder, а потім експортувати її до Python, щоб побачити, як ArcGIS побудує код. (Експорт складної моделі не рекомендується новачкам через велику кількість коду, який ModelBuilder, як правило, створює під час експорту Python).

2.3. Короткий огляд ModelBuilder

Щоб відкрити ModelBuilder, натисніть кнопку *Geoprocessing* > *ModelBuilder* (*рис.2.3*). Щоб додати дані та інструменти на полотно ModelBuilder, перетягніть їх з вікна Каталог або Пошук, або скористайтеся кнопкою Додати . Зв'язати дані з інструментами можна у різний спосіб; один із них – використати інструмент Додати підключення (Add Connection) .

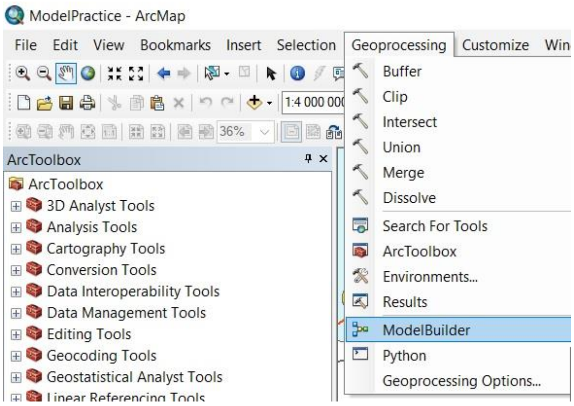


Рис.2.3 Запуск ModelBuilder через меню Geoprocessing

ModelBuilder має простий інтерфейс з меню, панелями інструментів та контекстними меню, як показано нижче (*рис 2.4*). Контекстні меню призначені для моделі в цілому та для будь-яких окремих елементів моделі (змінної, з'єднувача або інструменту). Ці меню відкриваються клацанням правою кнопкою миші. Порожня область моделі білого кольору, куди користувач перетягує інструменти і де вони зв'язуються зі змінними, називається полотном (canvas), а зовнішній вигляд і компонування інструментів і змінних пов'язаних один з одним називаються схемою моделі.

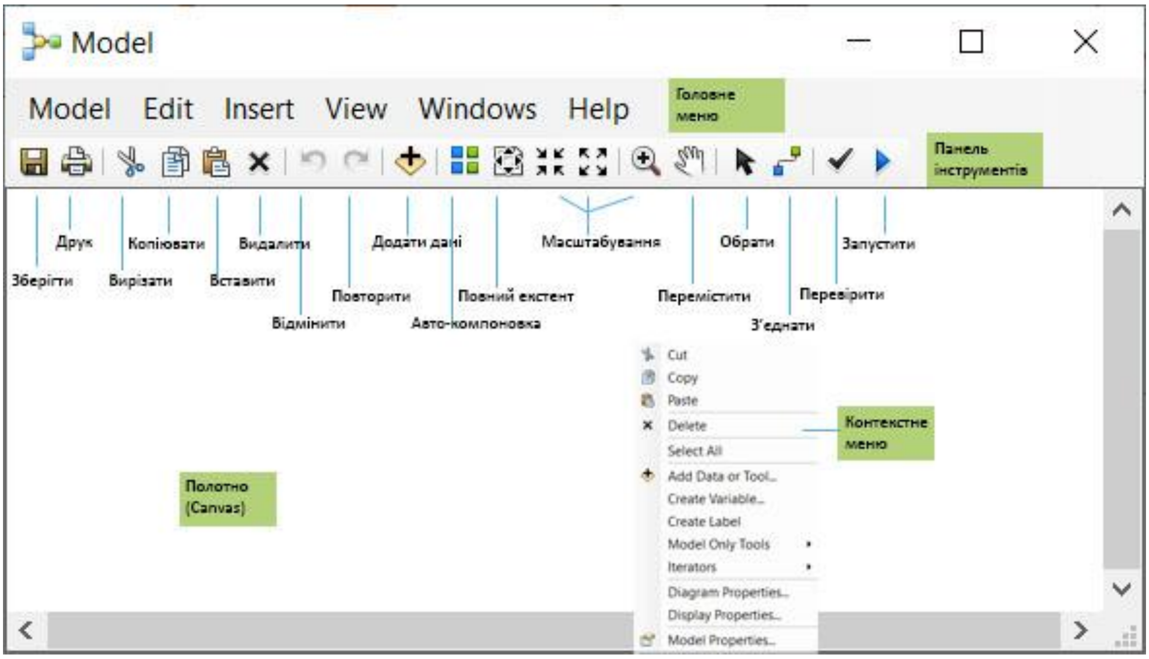


Рис.2.4 Інтерфейс ModelBuilder

У головному меню знаходиться шість спадаючих меню, перелічені в наступній таблиці (*таблиця 2.1*):

Таблиця 2.1

Menu	Опис
Model	Параметри для виконання, перевірки, перегляду повідомлень, збереження, друку, імпорту, експорту та закриття моделі. Це меню також можна використовувати для видалення проміжних даних та визначення властивостей моделі.
Edit	Вирізання, копіювання, вставка, видалення та вибір елементів моделі.
Insert	Додавання даних або інструменту, створення змінної, створення напису та додавання Інструментів лише моделі та Ітераторів.
View	Містить функцію автокомпонування, яка застосовує до моделі параметри, задані в діалоговому вікні Властивості схеми. Тут також можна знайти параметри зміни масштабу. Параметр Користувальницький масштаб дозволяє встановити особливий масштаб у відсотках. У меню Вид є рівні масштабу (25%, 50%, 75%, 100%, 200% і 400%) у відсотках від фактичного розміру.
Window	Містить вікно огляду, де можна переглянути модель, одночасно наблизивши певну частину моделі у вікні відображення. Поточне положення у вікні моделі позначено прямокутником у вікні Огляд. У процесі навігації у вікні ModelBuilder цей прямокутник переміщується відповідним чином.
Help	Доступ до довідки ArcGIS Desktop та вікна Про ModelBuilder.

Давайте трохи попрактикуємося з ModelBuilder, щоб вирішити реальний сценарій. Припустимо, Ви працюєте над задачею вибору місця, де вам потрібно вибрати всі райони, що знаходяться в межах 10 кілометрах від головного шосе та 10 кілометрах від міста. Вибрана область не може лежати в океані або за межами країни (Сполучених Штатів). Для вирішення проблеми потрібно створити буфери навколо *доріг* і *міст*, знайти перетин буферів, а потім обмежити результат контуром країни. Замість того, щоб вручну відкривати інструмент *Buffer* (двічі), потім інструмент *Intersect*, а потім інструмент *Clip*, Ви можете налаштувати використання всіх цих інструментів в ModelBuilder для виконання їх як одного процесу.

1. Створіть на своєму комп’ютері новий проект *ModelPractice.mxd* в папці **Lesson1** в *Catalog* (каталозі) .
2. Додайте шейп-файли *us_cities*, *us_roads* і *us_boundaries* до папки даних **Lesson1**, та додайте ці файли до фрейму проекту (*рис.2.5*).

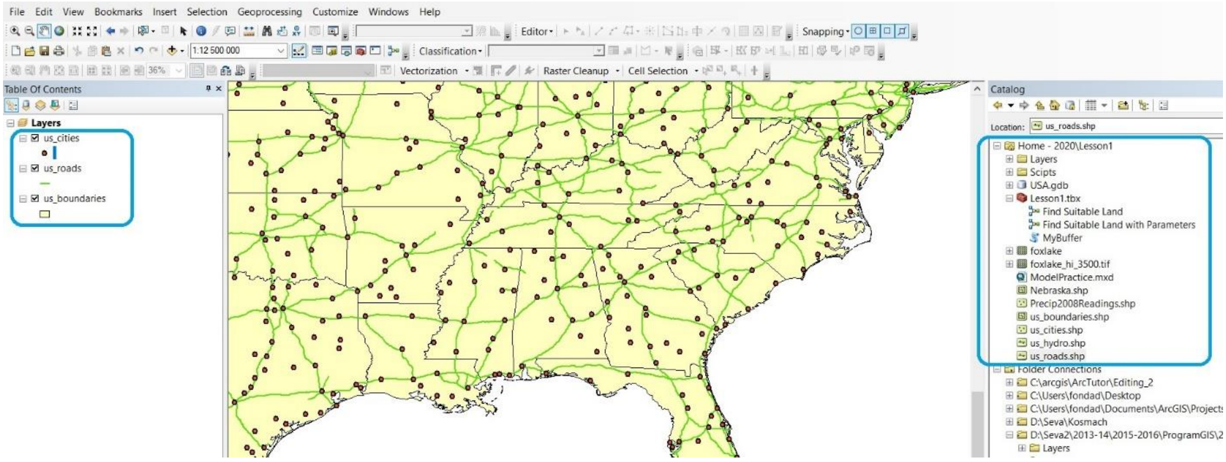


Рис.2.5 Проект ModelPractice.mxd з шарами даних us_cities, us_roads і us_boundaries

3. На панелі «*Catalog*» клацніть правою кнопкою миші на папці **Lesson1** і виберіть *New* > *Model*. Ви побачите, що вікно ModelBuilder з’явиться в середині вікна проекту.
4. На вкладці ModelBuilder натисніть кнопку *Properties* (Властивості).
5. В якості *Name* (імені) введіть *FindSuitableLand*, а для *Label* (мітки) — *Find Suitable Land*. *Label* – це те, що кожен побачить, коли відкриє ваш інструмент із *Catalog* (каталогу). Тому він може містити пробіли. *Name* – це те, що люди використовуватимуть, якщо коли-небудь запустять вашу модель з Python. Тому він не може містити пробіли. Адже імена модулів Python не можуть містити заборонені символи.
6. Натисніть **OK**, щоб закрити діалогове вікно властивостей моделі.
7. Тепер у вас є чистий canvas (полотно), на яке Ви можете перетягувати інструменти. Під час створення моделі (і під час написання сценаріїв Python) найкраще розбити вашу задачу на частини, якими можна керувати. Просту задачу вибору місця, яку ми поставили перед собою, можна розглядати як задачу на чотири кроки:

   - *Buffer* (Буферізація) міст
   - *Buffer* (Буферізація) доріг
   - *Intersect* (Перетин) отриманих буферів
   - *Clip* (обрізання) шарів по кордону США

   Розберемося з цими кроками по черзі, починаючи з *Buffer* (Буферізації) міст.

8. При відкритому ModelBuilder, перейдіть до панелі *ArcToolbox* > *Analysis Tools* (Інструменти аналізу) > *Proximity* (Близькість).
9. Оберіть інструмент *Buffer* і перетягніть його на полотно ModelBuilder. Ви побачите сіре прямокутне поле, що представляє інструмент *Buffer*, і

сірий овал, що представляє вихідні буфери, як результат. Вони з’єднані лінією, яка показує, що інструмент *Buffer* завжди створює набір вихідних даних.

У ModelBuilder інструменти представлені квадратами, а змінні — овалами. Зараз інструмент *Buffer* у центрі сірий, оскільки Ви ще не вказали необхідні параметри. Після заповнення всіх параметрів інструменту інструмент і змінна заповняться кольором.

10. У вікні ModelBuilder двічі клацніть прямокутник інструменту *Buffer*. Діалогове вікно інструмента тут таке саме, як якщо б Ви відкрили буфер безпосередньо з панелі *ArcToolbox*. Тут Ви можете заповнити всі параметри інструменту.
11. Для вхідних даних *Input Features* (вхідні просторові об'єкти) просто перейдіть по шляху до вашого шейп-файлу *us_cities* на диску. *Output Feature Class* (вихідні просторові об'єкти) буде заповнено автоматично.
12. Для параметра *Distance [value or field]* (Відстань [значення або поле]) введіть 10 кілометрів.
13. Для параметра *Dissolve Type* (Тип розчинення) виберіть *Dissolve all output features …* (Розчинити всі вихідні функції...), а потім натисніть **ОК**, щоб закрити діалогове вікно *Buffer*. Елементи моделі (інструменти та змінні) мають бути заповнені кольором, і Ви побачите новий елемент зліва від інструмента, що представляє вхідний клас об’єктів міст (*us_cities*).
14. Важливою частиною роботи з ModelBuilder є надання чітких міток (*labels*) для всіх елементів. Таким чином, якщо Ви поділитеся своєю моделлю, інші зможуть легко зрозуміти, що станеться, коли вона запуститься. Надання чітких міток також допоможе вам запам’ятати, що робить модель, особливо якщо Ви не працювали з моделлю деякий час.

    У ModelBuilder клацніть правою кнопкою миші елемент *us_cities.shp* (синій овал, крайній ліворуч) і натисніть *Rename* (Перейменувати). Назвіть цей елемент *US Cities*.

15. Клацніть правою кнопкою миші інструмент *Buffer* (жовто-оранжеве поле в центрі) і натисніть *Rename* (Перейменувати). Назвіть елемент *Buffer the cities*.
16. Клацніть правою кнопкою миші елемент виводу *Buffer* (зелений овал, крайній правий) та натисніть *Rename* (Перейменувати). Назвіть елемент *Buffered cities*. Ваша модель має виглядати так (*рис. 2.6*).

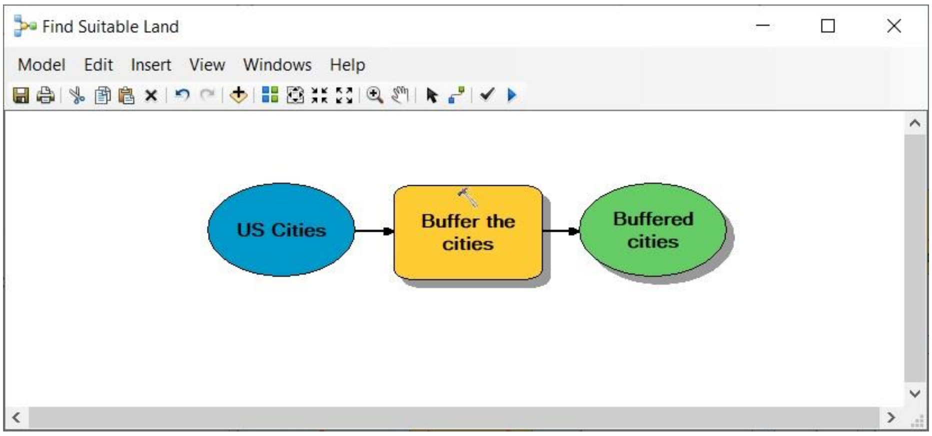


Рис.2.6 Вигляд моделі Find Suitable Land для кроку 16

17. Збережіть модель (вкладка *ModelBuilder* > *Save*). Варто це періодично робити.
18. Проробіть те саме, що щойно навчилися, додавши до моделі ще один інструмент *Buffer*. Цього разу налаштуйте інструмент так, щоб він буферизував шейп-файл *us_roads* на 10 кілометрів. Не забутьте встановити тип *Dissolve Type* (Тип розчинення) на *Dissolve all output features …* (Розчинити всі вихідні функції...) і додати значущі мітки. Тепер ваша модель має виглядати так (*рис. 2.7*).

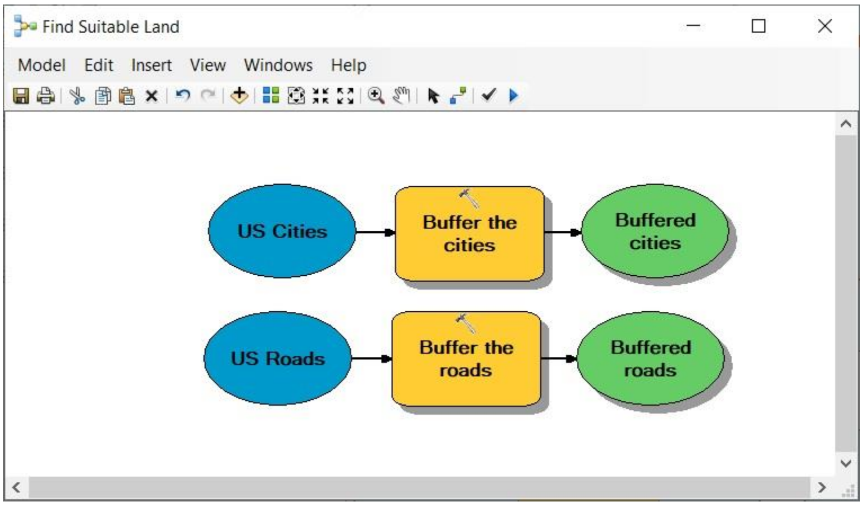


Рис.2.7 Вигляд моделі Find Suitable Land для кроку 18

19. Наступне завдання — знайти перетин (intersect) буферів. У списку панелі інструментів *ArcToolbox* перейдіть до *Analysis Tools* (Інструменти аналізу) > *Overlay* (Накладання) та перетягніть інструмент *Intersect* (Перетин) на свою модель. Розмістить його праворуч від наявних інструментів буфера.
20. Відмітимо ключові моменти, коли Ви об’єднуєте інструменти разом, встановлюючи вихідні дані ваших інструментів *Buffer* як вхідні дані для

інструменту *Intersect*. Клацніть елемент *Buffered cities* та перетягніть до елемента *Intersect*. Якщо Ви бачите невелике меню, клацніть *Input Features* (вхідні просторові об'єкти), щоб позначити, що буферизовані міста будуть використовуватися як вхідні дані для інструмента *Intersect*. Тепер стрілка вказуватиме від елемента *Buffered cities* до елемента *Intersect*.

21. Проробіть те саме, що щойно навчилися, щоб з'єднати буферизовані дороги з елементом *Intersect*. Знову, якщо з'явиться відповідний запит, натисніть *Input Features* (вхідні просторові об'єкти).
22. Перейменуйте вихідні дані операції *Intersect* на *Intersected buffers*. Якщо текст складається з кількох рядків, Ви можете клацнути та перетягнути краї елемента, щоб змінити його розмір. Ви також можете змінити порядок елементів на полотні, як вам подобається. Оскільки моделі можуть бути великими, ModelBuilder містить кілька кнопок навігації для збільшення та повного масштабування моделі в групі кнопок **View** на стрічці. Тепер ваша модель має виглядати так (*рис. 2.8*):

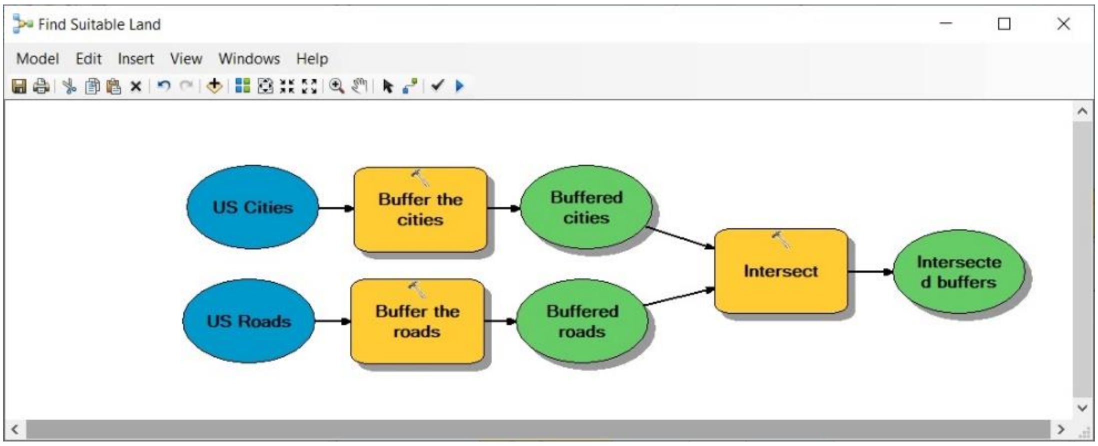


Рис.2.8 Вигляд моделі Find Suitable Land для кроку 22

23. Останнім кроком є обрізання отриманого результату від *Intersect* по контуру Сполучених Штатів. Це запобігає потраплянню будь-якої з виділених ділянок за межі країни або в океан. На панелі інструментів *ArcToolbox* перейдіть до *Analysis Tools* (Інструменти аналізу) > *Extract* (Витягнути) та перетягніть інструмент *Clip* (Кліп) у ModelBuilder. Розмістить цей інструмент праворуч від наявних інструментів.
24. Як Ви робили вище, встановіть елемент *Intersected buffers* як вхідні дані для інструмента *Clip*, вибираючи *Input Features* (вхідні просторові об'єкти), коли буде запропоновано. Зверніть увагу, що навіть коли Ви це робите, інструмент *Clip* не готовий до запуску (він все ще відображається у вигляді сірого прямокутника, розташованого праворуч). Вам потрібно надати в *Clip* форму, до якої будуть обрізані буфери.

25. У ModelBuilder двічі клацніть інструмент *Clip*. Налаштуйте *Clip Features*, (параметри інструменту - просторові об'єкти) перейшовши до шляху до *us_boundaries.shp*, а потім натисніть ОК, щоб закрити діалогове вікно. Ви помітите, що з'явився синій овал, що представляє елементи *Clip* (межі США).
26. Установіть значущі мітки для інших інструментів, як показано нижче. Нижче наведено приклад того, як можна позначити та розташувати елементи моделі (*рис. 2.9*).

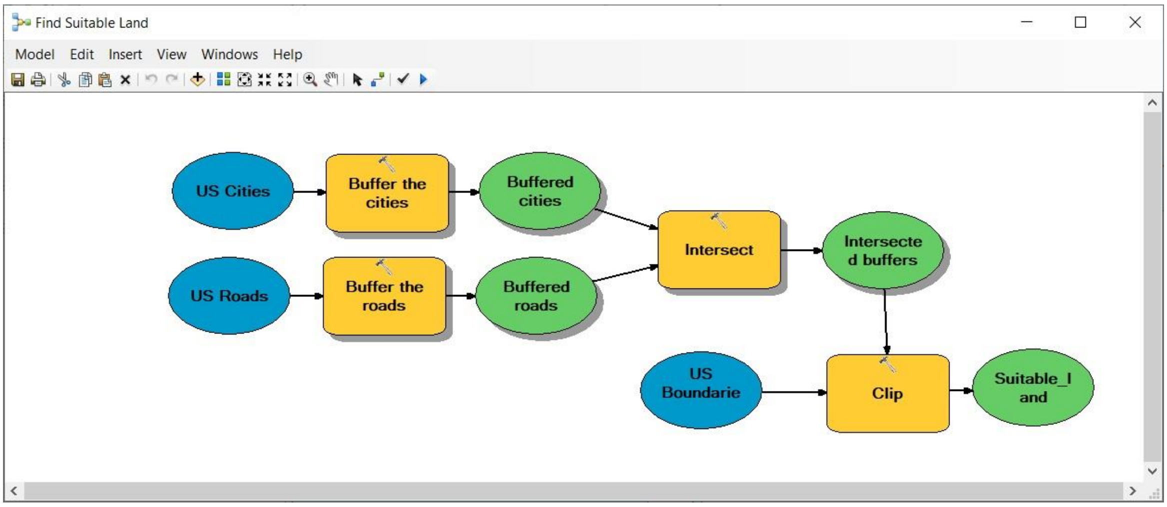


Рис.2.9 Вигляд результуючої моделі Find Suitable Land

27. Двічі клацніть кінцевий вихідний елемент (на зображенні вище з назвою *Suitable land*) і встановіть шлях до *suitable_land.shp* в папці **Lesson1**. Тут задається вихідний клас об'єктів вашої моделі, який буде записаний на диск.
28. Клацніть правою кнопкою миші на *Suitable land* та натисніть *Add to display* (Додати до відображення).
29. Збережіть модель знову. Збережіть проект *ModelPractice.mxd*.
30. Перевірте модель, натиснувши кнопку *Run* (Виконати) ▶. Після натиснення на кнопку запуску ModelBuilder Ви побачите вже знайоме вікно повідомлення геообробки **(geoprocessing**), яке повідомлятиме про будь-які помилки, які можуть виникнути. ModelBuilder також дає вам візуальну підказку про те, який інструмент запущено, підсвічуючи інструмент у червоний колір. (*Якщо модель виходить з ладу, спробуйте закрити* ModelBuilder *і запустити модель, двічі клацнувши її на панелі Catalog (каталогу). Ви отримаєте повідомлення, що модель не має параметрів. Це нормально [і вірно, як Ви дізнаєтеся нижче]. Все одно запустіть модель*).
31. Коли модель завершить роботу (це може зайняти деякий час), перегляньте вихідні дані на карті. Збільште масштаб штату Техас, щоб

переконатися, що кліп працював у прибережних районах. Вихід повинен виглядати подібно до цього (*рис. 2.10*).

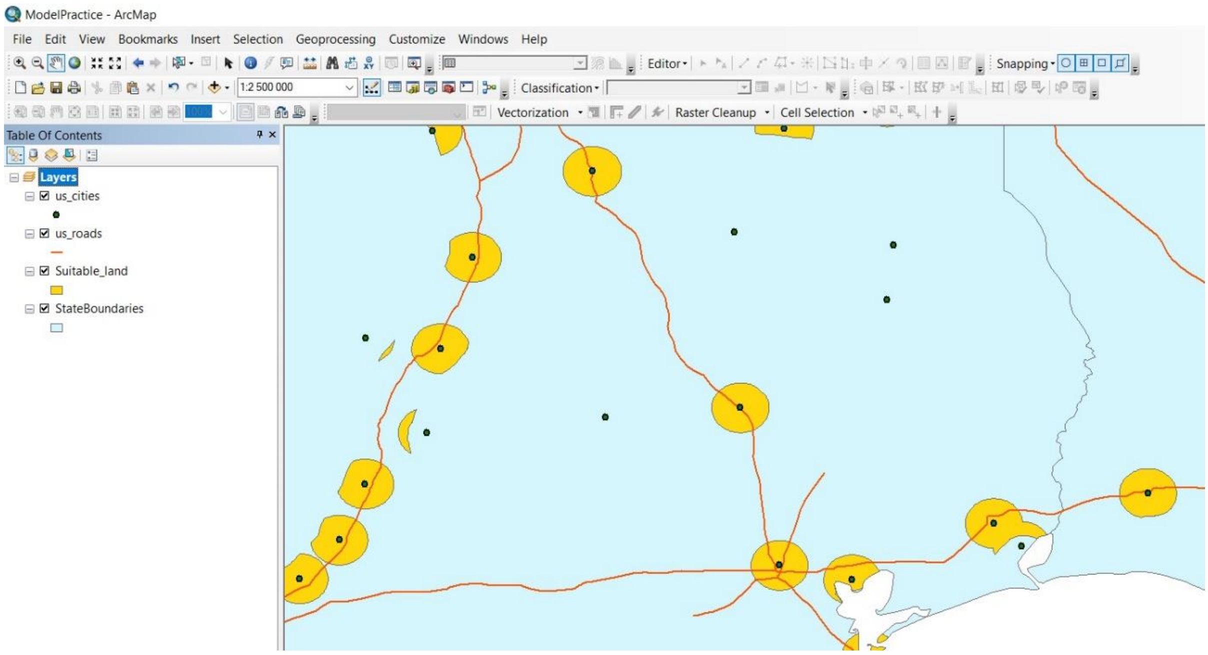


Рис.2.10 Результат роботи моделі Find Suitable Land

Ми щойно використали ModelBuilder, щоб об’єднати кілька інструментів і автоматизувати задачу в ГІС. Ви можете будь-коли двічі клацнути цю модель на панелі *Catalog (каталогу)* і запустити її так само, як інструмент. Якщо Ви це зробите, Ви помітите, що модель не має параметрів; Ви не можете змінити відстань буфера або функції введення. Правда в тому, що наша модель корисна для вирішення цієї конкретної проблеми вибору ділянок з цими конкретними наборами даних, але вона **не дуже гнучка**. Щоб зробити цю модель більш універсальною треба налаштувавши деякі змінні, як вхідні так і вихідні параметри.

2.3. Використання моделі з параметрами в ModelBuilder

Більшість інструментів, моделей і сценаріїв, які Ви створюєте за допомогою ArcGIS, мають параметри. ***Вхідні параметри*** — це значення, з якими інструмент (або модель, чи сценарій) починає свою роботу, а вихідні параметри представляють те, що інструмент видає після завершення роботи. Інструмент, модель або сценарій без параметрів можуть бути застосовані лише в обмеженому випадку з фактично фіксованими даними. Розглянемо модель *FindSuitableLand*, яку Ви щойно створили, яка використовує інструменти *Buffer*, *Intersect* і *Clip*. Ця модель була жорстко запрограмована для використання шейп-файлів *us_cities*, *us_roads* і *us_boundaries* і виведення шейп-файлу під назвою *Suitable land (suitable_land.shp)*. Іншими словами, якщо Ви хочете запустити модель з іншими наборами даних, вам потрібно буде відкрити ModelBuilder, двічі клацнути кожен елемент (*US Cities*, *US Roads*, *US Boundaries*, та *Suitable land*) і змінити написані шляхи до відповідних файлів в моделі. Вам доведеться виконати подібний процес, якщо Ви також хочете змінити буферні відстані, оскільки вони були жорстко закодовані до 10 миль. Модифікуємо цю модель, щоб використовувати деякі параметри, щоб Ви могли легко запускати її з різними наборами даних і відстанями буфера.

1. Відкрийте проект *ModelPractice.mxd* в папці **Lesson1** на своєму комп'ютері.
2. У вікні *Catalog (каталогу)* знайдіть модель, яку Ви створили раніше, яка має бути в розділі *Toolboxes* > *Lesson1* > *FindSuitableLand*.
3. Клацніть правою кнопкою миші модель *FindSuitableLand* та натисніть *Copy* (Копіювати). Тепер клацніть правою кнопкою миші на папку **Lesson1** та натисніть *Paste* (Вставити). Це створює нову копію моделі, з якою Ви можете працювати для додавання параметрів в модель. Використання такої копії моделі дозволяє легко почати все спочатку, якщо Ви зробили помилку.
4. Перейменуйте копію своєї моделі, наприклад на *Find Suitable Land With Parameters* або щось подібне.
5. В *Catalog (каталозі)* клацніть правою кнопкою миші на папці **Lesson1** на *Find Suitable Land With Parameters* та натисніть *Edit* (Редагувати). Ви побачите, як модель з'явиться в ModelBuilder.
6. Клацніть правою кнопкою миші елемент *US Cities* (повинен бути синім овалом) і натисніть *Parameter* (Параметр). Це означає, що той, хто запускає модель, повинен вказати набір даних для міст, які необхідно задати для аналізу, перш ніж модель зможе запуститися.
7. Відповідно тепер потрібна більш загальна назва для цього параметра, тому клацніть правою кнопкою миші елемент *US Cities* та натисніть *Rename* (Перейменувати). Змініть назву на просто *Cities*.
8. Незважаючи на те, що Ви «параметризували» міста, ваша модель все ще використовує набір даних ...\Lesson1\us_cities.shp. Це не матиме особливого сенсу, якщо Ви поділитеся своєю моделлю або набором інструментів з іншими людьми, тому що вони можуть не мати той самий шейп-файл us_cities, і навіть якщо вони його мають, він, ймовірно, не буде знаходитися на тому самому шляху на їхніх комп'ютерах.

   Щоб видалити набір даних за замовчуванням, двічі клацніть елемент *Cities* і видаліть шлях, а потім натисніть кнопку **OK**. Деякі елементи у вашій моделі можуть стати сірими. Це означає, що перед успішним запуском моделі необхідно надати значення моделі, але це буде робитися вже під час запуску моделі.

9. Тепер потрібно створити параметр для відстані буфера, який буде створено навколо міст. Клацніть правою кнопкою миші елемент, який Ви назвали *Buffer the cities*, і натисніть *Create Variable* (Створити змінну) > *From Parameter* (З параметра) > *Distance* [value or field] (Відстань [значення або поле]).
10. Попередній крок створив новий елемент *Distance* [value or field]. Перейменуйте цей елемент на *Cities buffer distance* і зробіть його параметром моделі. (Перегляньте наведені вище кроки, якщо Ви не впевнені, як перейменувати елемент або зробити його параметром моделі.) Для цього елемента Ви можете залишити значення за замовчуванням на 10 кілометрах. Ваша модель має виглядати подібно до цієї, хоча рядок заголовка вашого вікна може відрізнятися (*рис. 2.11*):

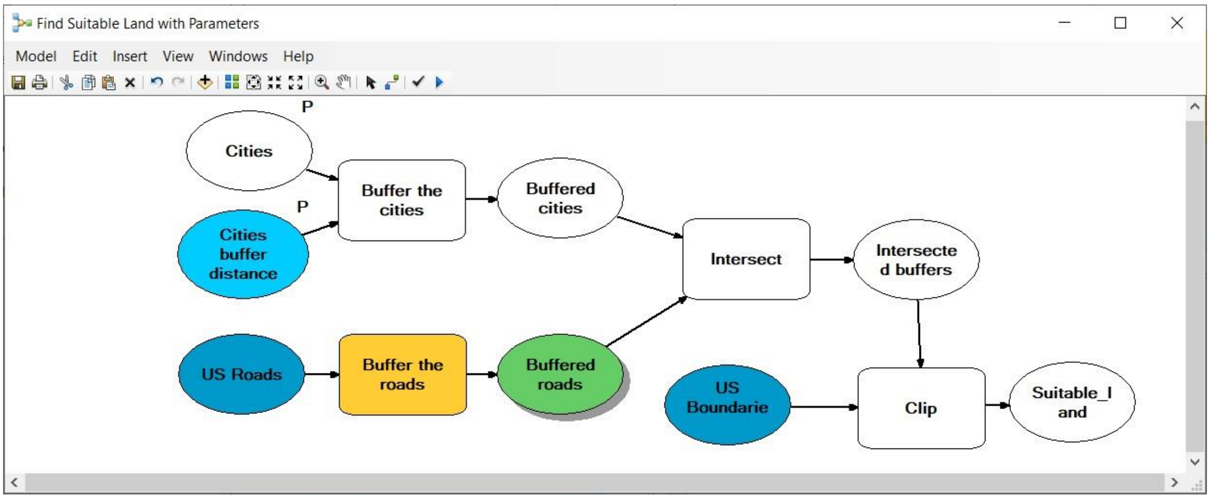


Рис.2.11 Результат зміни моделі з додаванням параметрів до Buffer the cities в моделі Find Suitable Land With Parameters

11. Проробіть те саме, що щойно навчилися. Перейменуйте елемент *US Roads* на *Roads*, зробіть його параметром моделі та видаліть значення за замовчуванням.
12. Повторюючи те, що Ви дізналися вище, створіть параметр для буферної відстані доріг (*Roads buffer distance*). Залиште значення за замовчуванням на 10 кілометрів.
13. Повторюючи те, що Ви дізналися вище, перейменуйте елемент *US Boundaries* на *Boundaries*, зробіть його параметром моделі та видаліть значення за замовчуванням. У параметра *Suitable land* видаліть значення за замовчуванням, та зробіть його параметром моделі. Ваша

модель має виглядати так (зверніть увагу на шість параметрів, позначені **P**) (*рис. 2.12*):

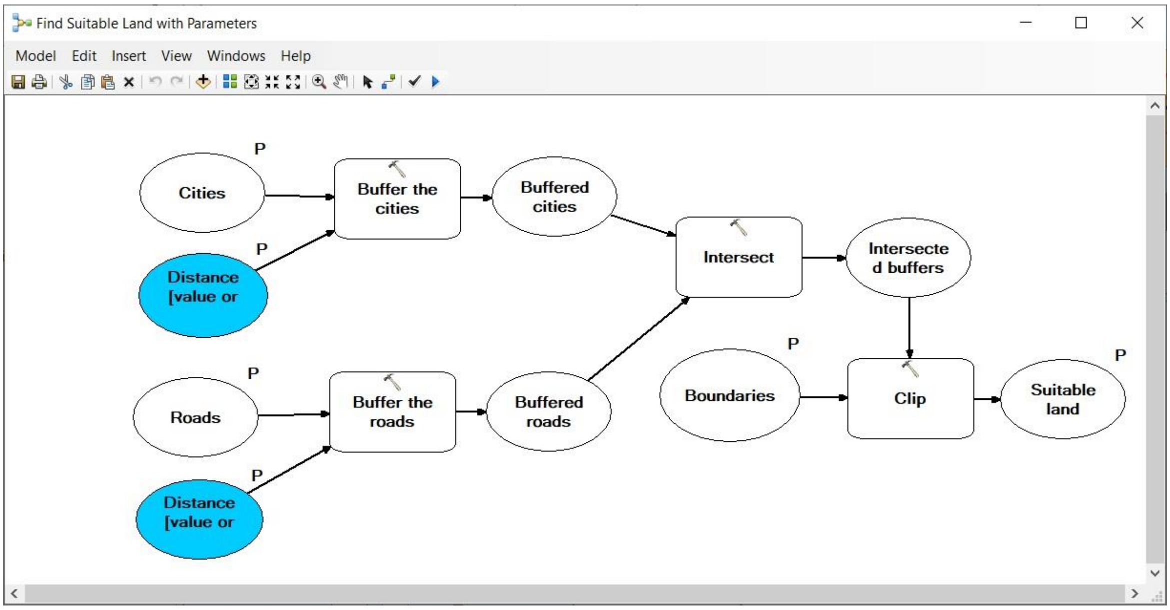


Рис.2.12 Результат зміни моделі з додаванням параметрів в моделі Find Suitable Land With Parameters

14. Збережіть і закрийте модель. Збережіть проект *ModelPractice.mxd*.
15. Двічі клацніть свою модель на папці **Lesson1** на *Find Suitable Land With Parameters* та перегляньте діалогове вікно інструмента. Це має виглядати так (*рис. 2.13*):

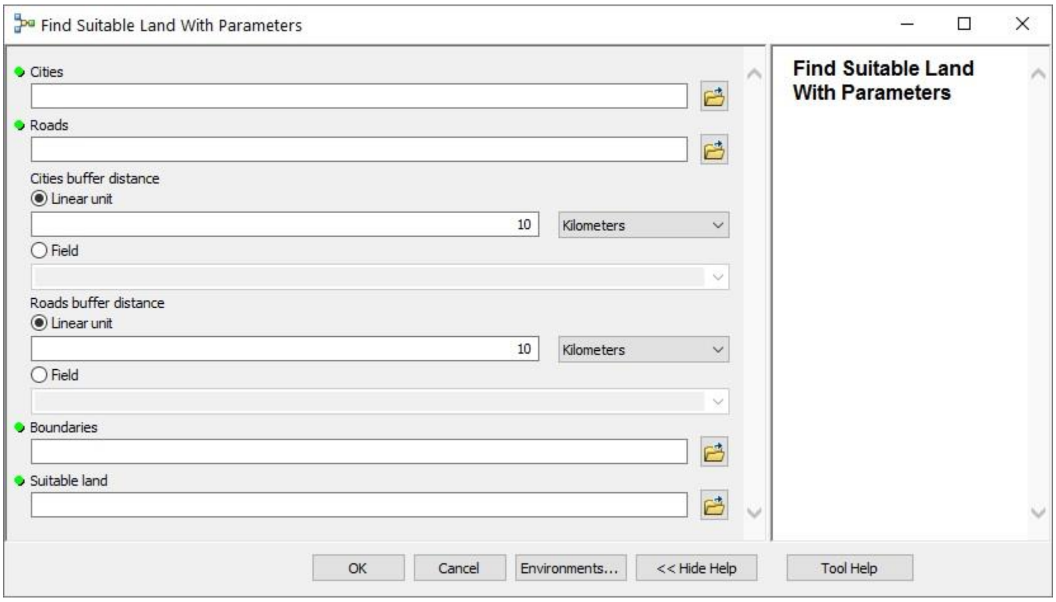


Рис.2.13 UI моделі Find Suitable Land With Parameters. Результат запуску

16. Люди, які користуються цією моделлю, зможуть переглядати будь-які набори даних міст, доріг та кордонів, а також зможуть контролювати відстань буфера. Зелені точки вказують на параметри, які повинні бути задані дійсними значеннями, перш ніж модель зможе працювати.
17. Перевірте свою модель, надавши шейп-файли *us_cities*, *us_roads* і *us_boundaries* для параметрів моделі. Якщо хочете, Ви можете спробувати змінити відстань буфера.

Наведений вище приклад продемонстрував, як Ви можете надавати значення як параметри за допомогою ModelBuilder. Вам потрібно вирішити, які значення Ви хочете, щоб користувач міг змінювати, і призначити їх як параметри. Коли Ви пишете сценарії Python, вам також потрібно буде визначати та виставляти параметри подібним чином.

2.4. Розширені концепції геообробки та ModelBuilder

Наразі ми трохи вивчили та попрактикувалися з ModelBuilder, і ми майже готові роздивитися роботу з Python. І хоча більш детально ми не будемо роздивлятися ModelBuilder, варто означити його додаткові можливості та матеріали, які Ви можете прочитати про ModelBuilder. Це особливо корисно, якщо Ви плануєте часто використовувати ModelBuilder у своїй роботі. Деякі елементи є спільними для фреймворку геообробки ArcGIS, що означає, що вони також застосовуються під час написання сценаріїв Python за допомогою ArcGIS.

Аналіз ГІС іноді заплутаний. Більшість інструментів, які Ви запускаєте, створюють набір вихідних даних, і коли Ви об'єднуєте багато інструментів, ці набори даних починають накопичуватися на диску. Esri запрограмувала поведінку ModelBuilder за замовчуванням так, що коли модель запускається з вбудованого інтерфейсу UI, всі набори даних, окрім кінцевого результату, які називаються проміжними даними, автоматично видаляються. З іншого боку, якщо модель запускається з ModelBuilder, проміжні набори даних залишаються у зазначених місцях.

ModelBuilder містить ряд елементів, які називаються ітераторами, які можуть виконувати цикли різними способами. Назви цих ітераторів, наприклад For і While, насправді імітують типи циклів, які можна програмувати на Python та інших мовах. У цьому посібнику ми не будемо зупинятися на вивченні ітерації в Python, що насправді є так само легко, як і навчитися використовувати ітератор у ModelBuilder. Щоб подивитись на те, як працює ітерація в ModelBuilder, Ви можете скористатися додатковими ресурсами в офіційній документації ArcGIS ModeBuilder щодо використання ітерацій у моделі.

Нижче наведені теми від Esri більш детально розповідають про додаткові можливості ModeBuilder:

- Списки
- Ітерація
- Зворотній зв'язок
- Логіка «якщо інакше» (if-then-else)
- Інтеграція моделі, скриптів та зовнішніх програм

Однак, в заключення даного розділу, варто згадати як отримати власне код, який створив та використовує ModeBuilder під час запуску моделі.

Розглянемо як отримати код в Python для зробленої моделі *Find Suitable Land With Parameters*. Для цього треба зробити декілька простих кроків:

1. Відкрийте проект *ModelPractice.mxd* в папці **Lesson1** на своєму комп'ютері.
2. У вікні *Catalog (каталогу)* знайдіть модель, яку Ви створили раніше, яка має бути в розділі *Toolboxes* > *Lesson1* > *Find Suitable Land With Parameters*.
3. В *Catalog (каталозі)* клацніть правою кнопкою миші на папці **Lesson1** на *Find Suitable Land With Parameters* та натисніть *Edit* (Редагувати). Ви побачите, як модель з'явиться в ModelBuilder.
4. В головному меню ModelBuilder натисніть на Model. Та у спадаючому списку оберіть *Export (Експорт) > To Python Script (В сценарій Python)* (*рис. 2.14*).

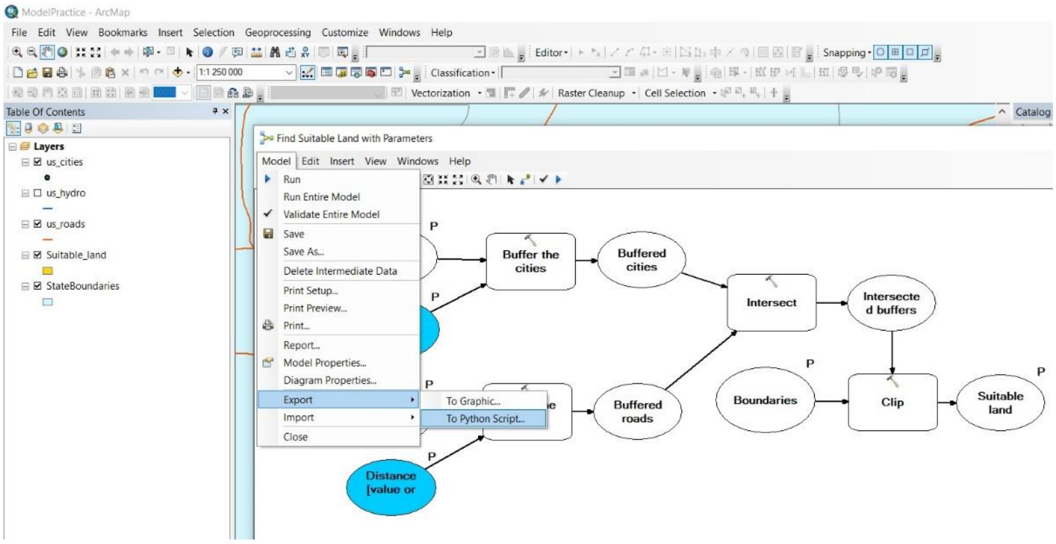


Рис.2.14 Експорт моделі Find Suitable Land With Parameters в сценарій Python

5. Створіть в папці **Lesson1** папку *Scripts*, де Ви будете зберігати сценарії для цього розділу посібника. Та збережіть сценарій з ім’ям find_suitable_land_param.py.
6. Відкрийте для вивчення файл find_suitable_land_param.py в будь-якому середовищі для програмування. Наприклад у вбудованому IDLE або PyCharm. Результат буде подібним до наведеного нижче (*рис. 2.15*):

```
*find_suitable_land_param.py - D:\Seva2\2013-14\2015-2016\ProgramGIS\2020\L5\2020\Lesson1\Scipts\find_suitable_land_param.py (2.7.12)*
File Edit Format Run Options Window Help
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------------
# find_suitable_land_param.py
# Created on: 2020-04-01 10:36:41.00000
#   (generated by ArcGIS/ModelBuilder)
# Usage: find_suitable_land_param <Cities> <Distance__value_or_field_> <Roads> <Distance__value_or_field___2_> <Boundaries> <Suitable_land>
# Description:
# ---------------------------------------------------------------------------

# Import arcpy module
import arcpy

# Script arguments
Cities = arcpy.GetParameterAsText(0)

Distance__value_or_field_ = arcpy.GetParameterAsText(1)
if Distance__value_or_field_ == '#' or not Distance__value_or_field_:
    Distance__value_or_field_ = "10 Kilometers" # provide a default value if unspecified

Roads = arcpy.GetParameterAsText(2)

Distance__value_or_field___2_ = arcpy.GetParameterAsText(3)
if Distance__value_or_field___2_ == '#' or not Distance__value_or_field___2_:
    Distance__value_or_field___2_ = "10 Kilometers" # provide a default value if unspecified

Boundaries = arcpy.GetParameterAsText(4)

Suitable_land = arcpy.GetParameterAsText(5)

# Local variables:
Buffered_cities = "D:\\ProgramGIS\\2020\\L5\\2020\\Lesson1\\USA.gdb\\cities_buffered"
Buffered_roads = "D:\\ProgramGIS\\2020\\L5\\2020\\Lesson1\\USA.gdb\\roads_buffered"
Intersected_buffers = "D:\\ProgramGIS\\2020\\L5\\2020\\Lesson1\\USA.gdb\\itersected_buffers"

# Process: Buffer the cities
arcpy.Buffer_analysis(Cities, Buffered_cities, Distance__value_or_field_, "FULL", "ROUND", "ALL", "", "PLANAR")

# Process: Buffer the roads
arcpy.Buffer_analysis(Roads, Buffered_roads, Distance__value_or_field___2_, "FULL", "ROUND", "ALL", "", "PLANAR")

# Process: Intersect
arcpy.Intersect_analysis("D:\\ProgramGIS\\2020\\L5\\2020\\Lesson1\\USA.gdb\\cities_buffered #;D:\\ProgramGIS\\2020\\L5\\2020\\Lesson1\\USA.gdb\\roads_buffered #", Intersected_buffers, "ALL", "", "INPUT")

# Process: Clip
arcpy.Clip_analysis(Intersected_buffers, Boundaries, Suitable_land, "")
Ln: 42 Col: 9
```

Рис.2.15 Експорт моделі Find Suitable Land With Parameters в сценарій Python

7. Проробіть те саме, що щойно навчилися для моделі *Find Suitable Land*.
8. Спробуйте дослідити ці коди із досвіду програмування в одному із класичних середовищ програмування Python.

Детально що означають ті чи інші рядки коду Ви можете зрозуміти і зараз. Однак для більш детального вивчення коду нам доведеться познайомитися з ArcPy та принципами організації об’єктів в ArcGIS.

Питання для самоконтролю:

1. Які способи автоматизації геообробки Ви можете навести?
2. Які переваги та недоліки надає використання ModelBuilder в порівнянні з класичним програмуванням на Python?

3. Як зробити використання ModelBuilder з різними даними більш гнучким в порівнянні з жорсткою задачею шляхів до файлів?
4. Чи є різниця, якщо Ви будете використовувати якійсь інструмент безпосередньо в *ArcToolbox*, чи в ModelBuilder?
5. Яку різницю Ви побачили в експортованих кодах для моделей *Find Suitable Land* та *Find Suitable Land With Parameters*?

3. ВСТУП ДО ARCPY

3.1. Що таке ArcPy

ArcPy - це пакет, пов'язаних з ArcGIS функцій і модулів. Його метою є створення основи для успішного та продуктивного виконання аналізу геопросторових даних, конвертації даних, управління даними та автоматизації картографії в Python.

Цей пакет забезпечує тісну та оригінальну взаємодію з користувачем у Python, надаючи функцію завершення коду (методи класів, тобто фактично після відповідного об’єкта ставимо точку і перед нами відкриється список властивостей та методів за цим об’єктом. Додатковою перевагою використання ArcPy у Python є те, що Python є мовою програмування загального призначення. Ця мова інтерпретується та вводиться динамічно. Крім того, він підходить для інтерактивної роботи та оперативного створення прототипів таких невеликих програм, як скрипти, але в той же час він має достатню потужність для створення великих додатків.

Програми ArcGIS, написані з використанням ArcPy, мають перевагу розробки додаткових модулів у численних сферах для професіоналів Python у ГІС, а також програмістів, що працюють в інших дисциплінах.

3.2. Імпорт ArcPy та написання простого сценарію

Додатки та скрипти ArcGIS написані з використанням ArcPy дозволяють отримати доступ до багаточисленних модулів Python, розроблених користувачами ГІС та програмістами, що працюють у різних галузях.

Але як почати використовувати ArcPy? Достатньо просто – імпортуйте модуль в тому середовищі, де Ви плануєте писати код:

```
>>> import arcpy
```

Після імпорту ArcPy Ви можете запустити всі інструменти геообробки, які входять до складу стандартних наборів інструментів, встановлених разом з ArcGIS:

- Analysis toolbox
- Cartography toolbox
- Conversion toolbox
- Data Management toolbox

- Editing toolbox
- Geocoding toolbox
- Linear Referencing toolbox
- Multidimension toolbox
- Spatial Statistics toolbox

Але, найкращий спосіб почати занурення у Python – це поглянути на невеликий фрагмент коду. Давайте візьмемо інструмент *Buffer*, який ми нещодавно запустили з панелі *Geoprocessing*, і запустимо його у вікні *ArcGIS Python*. Це вікно дозволяє вводити просту серію команд Python без написання повних постійних сценаріїв. Вікно Python — це чудовий спосіб почати роботу з Python.

Давайте створимо буфери на 10 кілометрів навколо міст.

1. Створіть новий порожній проект, або відкрийте проект *ModelPractice.mxd* в папці **Lesson1** на своєму комп'ютері.
2. Якщо Ви створили новий порожній проект, то додайте набір даних *us_cities.shp*.
3. На вкладці *Geoprocessing* натисніть кнопку *Python* (Вікно Python) . (*рис. 3.1*).

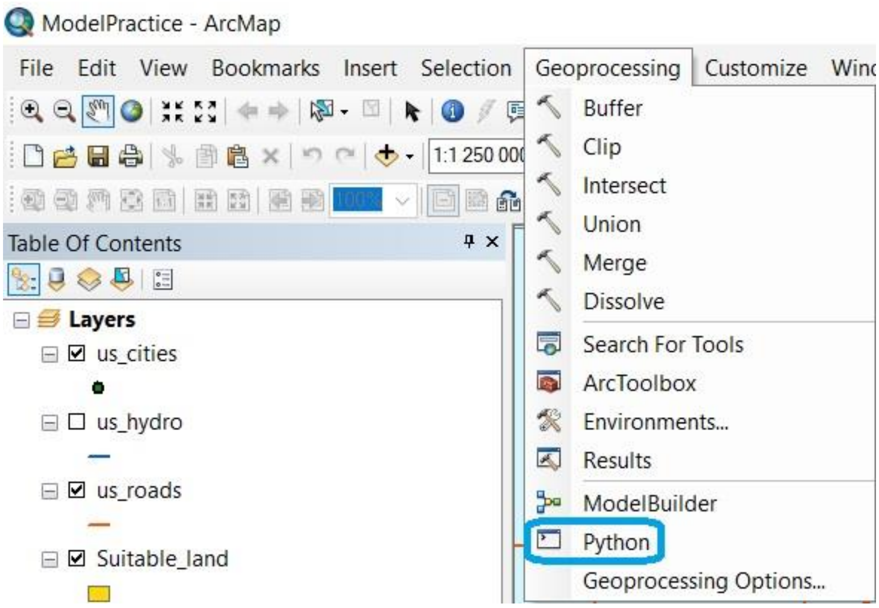


Рис.3.1 Запуск вікна Python в ArcMap

4. Введіть наступний код у вікні Python (*не вводьте* >>>. Символи просто включені, щоб показати Вам, де починаються нові рядки у вікні Python).

```
>>> import arcpy
>>> arcpy.Buffer_analysis("us_cities", "us_cities_buffered", "10 Kilometers", "", "", "ALL")
```

5. Збільште масштаб і Ви можете впевнитися, що буфери створені.

Ви щойно запустили свій перший сценарій на Python. Вам не обов'язково розуміти зараз все про код, який Ви написали в цьому вікні, але ось кілька важливих речей, на які слід звернути увагу.

Перший рядок сценарію — import arcpy — повідомляє інтерпретатору Python (який був встановлений під час інсталяції ArcGIS), що Ви збираєтеся працювати з деякими спеціальними функціями сценаріїв та інструментами, які входять до ArcGIS. Без цього рядка коду Python нічого не знає про ArcGIS, тому Ви помістите його вгору всього коду, пов'язаного з ArcGIS, який Ви будете програмувати в цьому посібнику. Отже, цей рядок коду ми будемо використовувати майже в усіх сценаріях, які Ви будете програмувати у Python для ArcGIS.

Другий рядок сценарію фактично запускає конкретний інструмент. Ви можете ввести arcpy, поставити крапку, а потім обрати або ввести будь-яку назву інструмента, щоб запустити інструмент у Python. Зверніть увагу, що тут Ви також ставите символ підкреслення, а потім назву панелі інструментів, яка містить інструмент Buffer (arcpy.Buffer_analysis). Це необхідно, оскільки деякі інструменти в різних панелях інструментів насправді мають однакові назви (наприклад, *Clip*, який є інструментом для відсікання векторів на панелі інструментів *Analysis* або інструментом для відсікання растрів у наборі інструментів Data Management (керування даними)).

3.3. Вибір середовища для роботи з ArcPy та Site package ArcPy

Вибір середовища для запуску сценаріїв з ArcPy є достатньо простою задачею, але важливою, адже створення комфортного середовища для розробника збільшує продуктивність роботи. Наступні питання та відповіді наведені в таблиці нижче (таблиця 3.1) допоможуть вам визначити, яке середовище буде найкращим вибором для вашого сценарію Вікно Python в ArcMap або класичні IDE, наприклад, PyCharm, Spyder та інші.

Таблиця 3.1

Вам необхідно	Вікно Python в ArcMap	IDE (PyCharm, Spyder)
Виконати один рядок коду?	X	X
Робота безпосередньо з шарами (layers) в ArcMap?	X	
Протестувати робочі процеси геообработки та легко перевірити результат?	X	

Попрацювати з інструментами для введення, редагування і перевірки синтаксису в коді?		Х
Попрацювати в інтегрованому середовищі налагодження (debugging)?		Х

Отже, Ви могли помітити в **розділі 3.2**, що вікно Python допомагає вам, відкриваючи різні параметри, які Ви можете ввести для кожного параметра. Це називається автозаповненням, і воно може бути дуже корисним, якщо Ви вперше намагаєтеся запустити інструмент і не знаєте, як точно ввести параметри. Ви також могли помітити, що деякий код з'являється, наприклад (Buffer() analysis і Buffer3D() 3d), коли Ви вводили ім'я функції. Ви можете використовувати стрілки вгору/вниз, щоб обирати альтернативи. Якщо Ви вибрали аналіз Buffer(), він з'явиться у вашому вікні Python (*рис. 3.2*).

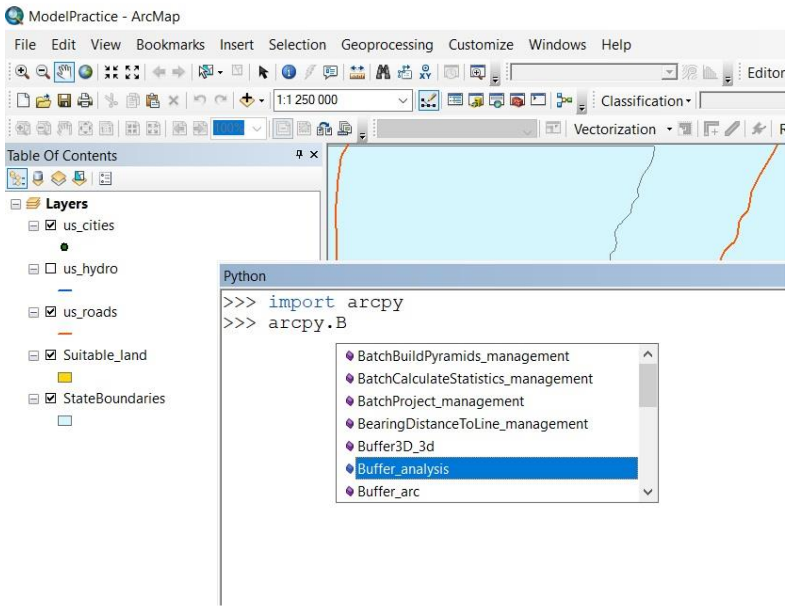


Рис.3.2 Приклад автозаповнення у вікні Python в ArcMap

Зауважте, що якщо Ви використовуєте доповнення коду, ваш код іноді виглядатиме дещо інакше - Esri постійно модифікує те, як влаштовані функції в ArcPy - вони працюють так само, але можуть мати трохи інший синтаксис. "Старий" спосіб все ще працює, тому Ви інколи можете побачити невідповідності коду в посібнику, онлайн-форумах, документації Esri тощо. Тож у цьому прикладі arcpy.Buffer_analysis(...) змінився на arcpy.analysis.Buffer(....) в версії ArcGis 10.8, і відображає те, що інструмент *Buffer* розташований у панелі інструментів *Analysis* у версії ArcGis 10.8 та Pro.

Функціональність Python в ArcGIS може варіюватися від простого виразу з використанням однієї з вбудованих функцій Python до складного скрипта для геообробки. Функції геообробки, поряд з іншими функціями, пов'язаними з ГІС, доступні через пакет Site packages ArcPy.

Ми його використовували раніше, коли імпортували відповідний модуль

```
>>> import arcpy
```

Думайте про Site package як про контейнер для *модулів* і *функцій*. Модулі являють собою контейнер функцій на основі теми, наприклад модуль представлення (mapping) або модуль просторового аналізу (spatial analyst). Site package ArcPy є контейнером пов'язаних з ГІС функцій і модулів (*рис. 3.3*).

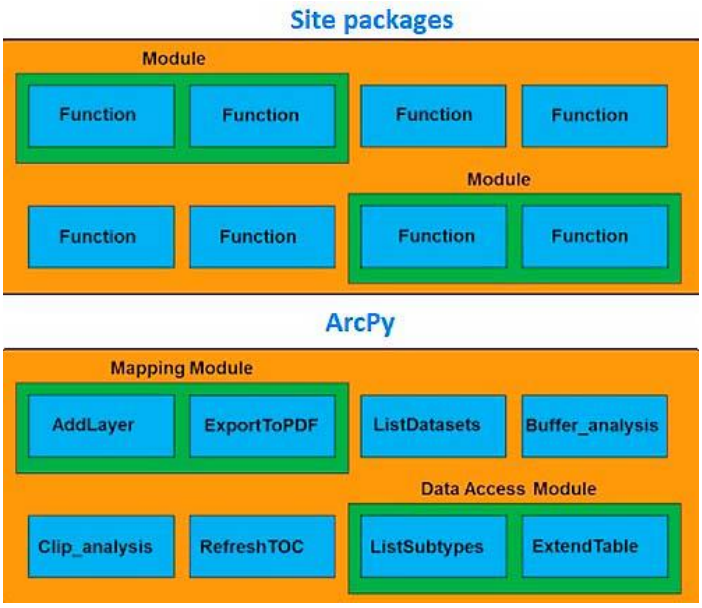


Рис.3.3 Site packages ArcPy в ArcMap (Python for Everyone)

3.4. Повторне використання коду Python в ArcGIS

Якщо Ви запускали на виконання якійсь інструмент ArcGIS ваш сценарій через Вікно Python в ArcMap або інші IDE, наприклад, PyCharm то результат роботи такого інструменту чи сценарію можна побачити не тільки у фреймі ArcMap, але у вікні *Results (Результати)* – перейдіть до вкладки *Geoprocessing* натисніть кнопку *Results*) . (*рис. 3.4*).

Припустимо, що ми використали фрейм *Providence.mxd* з шарами RI_Schools (точковий клас просторових об'єктів) та Providence (полігональний клас просторових об'єктів).

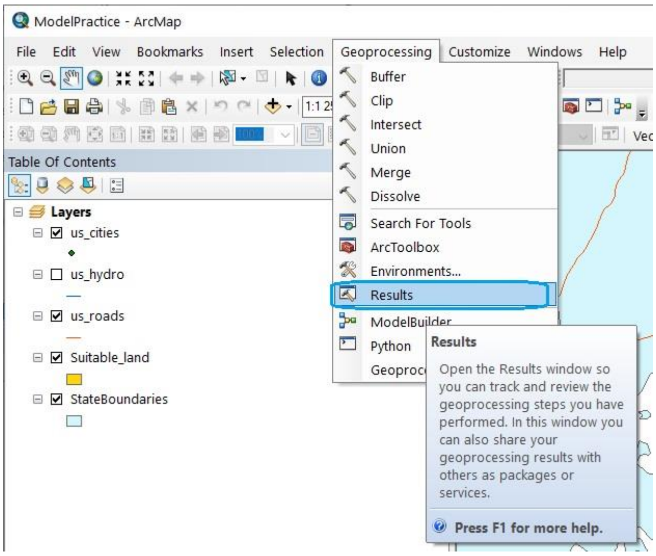


Рис.3.4 Запуск вікна Results в ArcMap

Тепер, коли карта налаштована, наш наступний крок - використати інструмент *Clip*, щоб вирізати клас просторових об'єктів (feature class), використовуючи кордон Providence.

1. Запустіть інструмент *Clip* на виконання. Вікно *ArcToolbox > Analysis Tools > Extract > Clip*.
2. Обріжте RI_Schools по контуру Providence та збережіть результат з іменем Schools в *RhodeIsland.gdb*.
3. У документ карти (map document) буде додано новий шар з ім'ям Schools.
4. У вікні *Results* розкрийте Current Session.
5. Клацніть правою кнопкою миші *Clip*, а потім виберіть *Copy As Python Snippet* (*рис. 3.5*).

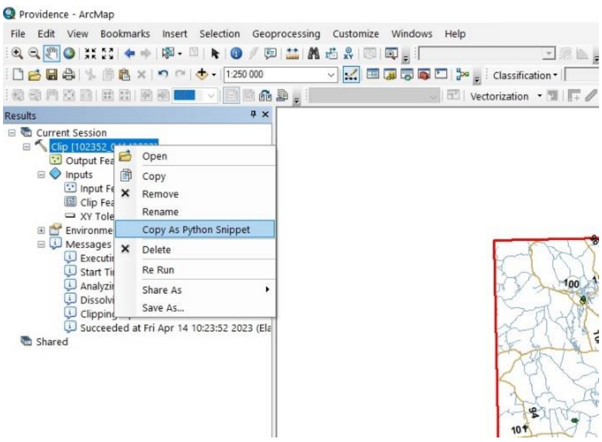


Рис.3.5 Копіювання коду для повторного запуску з вікна Results

6. При необхідності, відкрийте *Вікно Python в ArcMap*.

7. У *вікні Python*, клацніть правою кнопкою миші і виберіть команду *Paste* (Вставити).
8. Синтаксис починається з *Site package (arcpy)*, потім слідує функція (*Clip_analysis*). Параметри функції ті ж самі, що і в інструменту (Input Features, Clip Features, Output Feature Class, та XY Tolerance) (*рис. 3.6*).

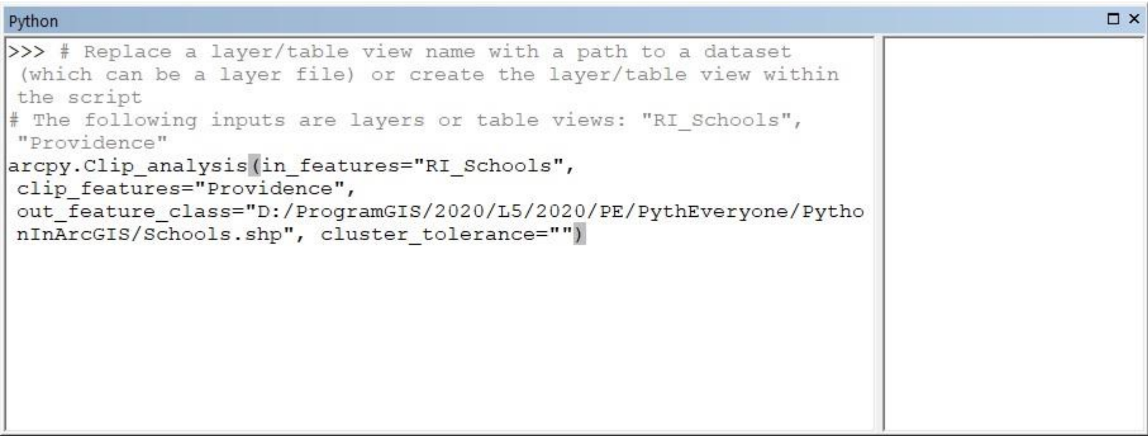


Рис.3.6 Повторний запуск скопійованого коду з вікна Python в ArcMap

Отже, достатньо просто за допомогою вікна *Results* побачити як виконався Ваш код, дослідити повідомлення які видав код та при необхідності виконати його повторно.

3.5. Приклад написання коду Python в ArcGIS

Для того щоб почати знайомство з тим як влаштований *ArcPy* розглянемо декілька прикладів.

- Сценарій просторової прив'язки (Spatial reference script) - Цей скрипт буде працювати з просторовою прив'язкою кожного класу просторових об'єктів (наприклад з заданої бази геоданих). Це дозволить Вам, наприклад, визначити, чи є необхідність перевизначення просторової прив’зки для певного класу об'єктів.
- Сценарій компоновки карт (Map layout script) - Цей скрипт допоможе переформатувати макет (компоновку) карт. Наприклад, на момент старту проекту макет карт включає тільки шар з розташуванням округів на карті («Polk County»). Сценарій переформатує макети карт, включивши в них текст, який також ідентифікує положення на карті («Polk County, OR»).

3.5.1. Сценарій просторової прив'язки (Spatial reference script)

Відкрийте файл *SpatialRef.py* та давайте дослідимо цей сценарій. Зауважимо, що даний приклад коду розрахований на запуск в IDE, наприклад, PyCharm. Також для його вдалого виконання необхідно додати необхідні параметри.

SpatialRef.py

```
# import modules
import arcpy

# set workspace
arcpy.env.workspace = r"geodatabase path"

# set up a describe object for each fc in gdb
fcList = arcpy.ListFeatureClasses()
for fc in fcList:
   desc = arcpy.Describe(fc)
   print (desc.spatialReference.name)
print ("Script completed")
```

Одразу відмічаємо, що для того щоб код який ми пишемо і запускаємо працював необхідно імпортувати необхідний модуль *ArcPy*, тож не будемо у подальшому постійно на цьому наголошувати, просто не забувайте про цей важливий крок.

```
import arcpy
```

Для того щоб визначити робочий простір для коду задаємо необхідний параметр

```
arcpy.env.workspace = r"geodatabase path"
```

Даний параметр допоможе нам при пошуку необхідних файлів та збереженні результатів роботи сценаріїв. Детальніше з даним параметром ми познайомимося пізніше. А зараз, якщо бажаєте виконати сценарій вдало, не забутьте дописати значення для параметра, це може бути, наприклад, база даних або звичайна директорія. Для прикладу - *Oregon_Polk.gdb*. Але шлях до бази необхідно прописати повністю, як до звичайного файлу, наприклад - r"D:\ProgramGIS\2020\L5\2020\PE\PythEveryone\RunningScripts\Oregon_Polk.gdb"

Роздивляючись код далі знаходимо рядок

```
fcList = arcpy.ListFeatureClasses()
```

Результатом його виконання буде *список* доступних просторових об'єктів у визначеному робочому просторі.

Наприклад для нашого прикладу він поверне результат

```
[u'Polk_County', u'Polk_Cities', u'Polk_Streams', u'Polk_GWpoll']
```

Функція *ListFeatureClasses()* повертає список класів просторових об'єктів у поточній робочій області і з нею і тими параметрами, які вона може приймати ми познайомимося пізніше. Але Ви можете почати знайомство і зараз, в документації за даною функцією - *ListFeatureClasses*.

Далі по коду ми організовуємо прохід по циклу за даним списком fcList. Таким чином, змінна fc на кожній ітерації циклу буде отримувати ім’я файлу з отриманого списку. Відповідно, для кожного файлу з бази даних ми будемо робити наступні операції

```
desc = arcpy.Describe(fc)

print (desc.spatialReference.name)
```

Змінна desc приймає результат виконання функції *Describe()*. Як результат це буде достатньо складний об’єкт *Describe*, для якого є відповідний перелік атрибутів та методів, з якими можна познайомитися в документації (*Describe*). Ми це познайомимося з даним об’єктом трохи пізніше. Доступний перелік включає багато описових даних для об’єктів, наприклад, тип даних, поля, індекси та багато інших.

В нашому прикладі нас зацікавила просторова прив’язка для файлів з бази даних. Для її отримання ми для об’єкт *Describe* викликаємо функцію *spatialReference*, яка повертає новий об’єкт *spatialReference*. З ним ми теж познайомимося пізніше. Для цього об’єкта *spatialReference* можна отримати багато властивостей, але нам цікаво отримати ім’я просторової прив’язки name.

Щоб переконатися що весь код виконався вдало ми видаємо повідомлення про це

```
print ("Script completed").
```

3.5.2. Сценарій компоновки карт (Map layout script)

Відкрийте файл *ChangeMapTextEle2.py* та давайте дослідимо цей сценарій. Також зауважимо, що даний приклад коду розрахований на запуск в IDE, наприклад, PyCharm.

ChangeMapTextEle2.py

```
# import modules
import arcpy

# set environment
mxd = arcpy.mapping.MapDocument("mxd path")

# list map layout text elements
eleList = arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT")

# loop through list and find all text elements with "Polk County" title
for ele in eleList:
   if ele.text == "Polk County":
      ele.text = "Polk County, OR"

# refresh the active view
arcpy.RefreshActiveView()

print ("Script completed")
```

Для того щоб визначити робочий простір для коду задаємо необхідний параметр *arcpy.mapping.MapDocument*. Ви вже можете відмітити, що ми використовуємо модуль mapping. Отже ми працюємо з компоновками і представленнями в ArcGIS. В даному прикладі для визначення робочого простору необхідно визначити параметр *MapDocument*. Задаємо йому значення, вказавши шлях до відповідного документа, наприклад – r"D:\ProgramGIS\2020\L5\2020\PE\PythEveryone\RunningScripts\PolkGrdWater.mxd".

Далі код схожий на попередній приклад. Ми так само бажаємо отримати перелік компонент, але для дано прикладу – перелік елементів компоновки, які існують у компоновці карти документа (.mxd).

```
eleList = arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT")
```

В якості аргументів для функції *ListLayoutElements* ми використовуємо ім'я документу та вказуємо які самі елементи на необхідні, в дану прикладі тільки текстові - "TEXT_ELEMENT".

Далі за сценарієм ми знаходимо елемент з вмістом "Polk County" та замінюємо вміст такого елемента на "Polk County, OR".

Оновлюємо активне представлення в ArcMap за допомогою відповідного метода

```
arcpy.RefreshActiveView()
```

Зауважимо, що якщо запустити на виконання даний сценарій, то результат ми не побачимо ніяких змін. Це сталося тому, що ми не зберігали документ. Якщо це Вам необхідно, то збережіть документ додавши до коду рядок

```
mxd.save()
```

Якщо ж є необхідність побачити результат без збереження, то це можна зробити якщо запустити даний код у Вікні Python в ArcMap.

Отже ми познайомилася з прикладами написання простого коду в *ArcPy*. Але для повного розуміння як написати сценарії в *ArcPy* необхідно познайомитися з класами і функціями, які є в доступі в даному модулями, розібратися з параметрами, які вони приймають на вхід та видають в якості результату.

3.6. Функція Describe

Знайомство з класами і функціями *ArcPy* почнемо з функції Describe. Це вкрай важливий компонент *ArcPy* і функція *Describe* (Описати) повертає об'єкт *Describe* з декількома властивостями, наприклад, тип даних, поля, індекси.

Властивості *Describe* організовані в серії груп властивостей. Всі дані незалежно від типу завжди отримують загальні властивості *Describe Object*. Якщо певний набір даних містить властивості хоча б однієї з цих груп, наприклад, якщо описується клас об'єктів бази геоданих (GDB FeatureClass), можна отримати властивості з груп властивостей GDB FeatureClass, FeatureClass, Table і Dataset.

Отже дана функція є по-суті функцією конструктором, яка створює екземпляр класу *Describe*, від якого організовується наслідування залежно від того якого типу об'єкти були задані на вхід функції конструктора.

Познайомитися з даною функцією *Describe* детально ватро в офіційній документації на сайті розробників. Синтаксис функції наступний:

Describe (value, {datatype})

Таблиця 3.2

Властивість	Пояснення	Тип даних
value	елемент даних або об'єкт геообробки для якого потрібний опис	String
datatype	Тип даних. Необхідно вказувати лише у випадку, коли існують конфлікти імен, наприклад, якщо база геоданих містить однойменний набір класів об'єктів (FeatureDataset) та клас просторових об'єктів (FeatureClass). (За замовчуванням — None)	String

Ми ж познайомимося з основними властивостями доступними власне *Describe Object*, об'єкта, який повертається функцією *Describe*.

Функція *Describe (Описати)* повертає такі властивості для всіх об'єктів *Describe* – всі ці властивості носять інформаційний характер і їх не можна змінити (таблиця 3.3).

Таблиця 3.3

Властивість	Пояснення	Тип даних, що повертаються
baseName	Базове ім'я файлу	String
catalogPath	Шлях до робочого простору	String
children	Список підпорядкованих елементів	Describe
childrenExpanded	Показує, чи розгорнуті нащадки	Boolean
dataElementType	Тип елементу	String
dataType	Тип даних	String
extension	Розширення файлу	String
file	Ім'я файлу	String

fullPropsRetrieved	Показує, чи отримані повні властивості	Boolean
metadataRetrieved	Показує, чи отримані метадані	Boolean
name	Визначене користувачем ім'я елемента	String
path	Шлях до файлу	String

Для того аби дослідити як взаємодіють групи властивостей з власне *Describe Object* розглянемо приклад для шарів (layer in ArcMap).

```
import arcpy

# Get the layer as a parameter and describe it.
## The layer could be a layer in ArcMap (like "some_layer")
# Or, it could be a .lyr file (like "C:/data/some.lyr")

layerString = arcpy.GetParameterAsText(0)
desc = arcpy.Describe(layerString)

# Print selected layer and describe object properties
print("Name: {}".format(desc.name))
if hasattr(desc, "layer"):
   print("Layer name: {}".format(desc.layer.name))
   print("Layer data source: {}".format(desc.layer.catalogPath))
   print(".lyr file: {}".format(desc.catalogPath))
else:
   print("Layer name: {}".format(desc.name))
   print("Layer data source: {}".format(desc. catalogPath))

if desc.FIDSet != '':
   print("Number of selected features: {}".format(len(desc.FIDSet.split(";"))))
```

Розглянемо даний код, але не будемо зупинятися на розглянутих раніше моментах. Зауважимо, що ми бажаємо отримати описову інформацію для шару даних, який може бути як динамічним шаром з фрейму ArcMap, або ж збереженим окремо шаром даних (lyr).

Наступним рядком

```
desc = arcpy.Describe(layerString)
```

ми отримуємо об’єкт *Describe Object* і якщо подивитися в таблицю 3.2, то ми можемо отримати будь-яку властивість для даного об’єкта. Наприклад name, або catalogPath. В той же час ми можемо помітити з коду, що частина властивостей, які ми викликаємо відсутня в таблиці. Як так трапилося? Причина в наслідуванні. Для шару даних є властивості які він унаслідував від *Describe Object*, а є і властивості, які характерні для даного типу даних. Зокрема - layer.name, layer.catalogPath.

Це легко зрозуміти якщо звернутися до документації по описових властивостях шарів Layer properties. Ознайомтеся з даною документацією, ми не будемо зупинятися на ній детальніше. Але зауважимо, що до подібного прикладу можна віднести *MosaicLayer*, *FeatureLayer* і *GroupLayer*.

Отже функція Describe є вкрай важливою, адже дозволяє отримати наявну інформацію про властивості об’єктів, причому це стосується не тільки тих, що ми подивилися, а всіх типів об’єктів роботу з якими підтримує ArcGIS, в тому числі і з растрами, і з мережевими шарами даних, і з багатьма іншими типами даних.

3.7. Функція SpatialReference

Окрім функції *Describe* ми вже познайомилися з функцією *SpatialReference*. В прикладі з розділу 3.5.2 було показано, як отримати доступ до просторової прив’язки саме через об’єкт *Describe*.

```
spatial_ref = arcpy.Describe(dataset).spatialReference
```

Однак, функція *SpatialReference* створює саме об’єкт *SpatialReference* із своїми властивостями та методами. Дослідимо функцію *SpatialReference* завдяки її документації. Якщо ми бажаємо створити об’єкт просторової прив’язки *SpatialReference* без використання функції *Describe*, то необхідно скористатися відповідним синтаксисом (таблиця 3.4):

```
SpatialReference ({item}, {vcs})
```

Таблиця 3.4

Властивість	Пояснення	Тип даних
item	Система координат, яка використовуватиметься для створення SpatialReference. Можна налаштувати за допомогою файлу проекції, назви, коду	Variant

	або загальновідомого текстового формату (WKT). Наприклад: • Використанням файлу проекції (.prj) sr = arcpy.SpatialReference("c:/coordsystems/NAD 1983.prj") • Використанням назви системи координат sr = arcpy.SpatialReference("Hawaii Albers Equal Area Conic")	
vcs	Вертикальна система координат (VCS). VCS визначає інформацію про координати Z. Можна встановити за допомогою назви, коду або загальновідомого текстового формату (WKT)	Variant

Кожна частина просторової прив'язки (*SpatialReference*) має безліч властивостей, зокрема, систему координат, яка визначає, які опції картографічної проекції використовуються для визначення горизонтальних координат.

Отже, об'єкт *SpatialReference* можна легко створити, якщо скористатися відповідним методом, наприклад

```
sr = arcpy.SpatialReference(r"c:/coordsystems/NAD 1983.prj")
```

Які саме властивості просторової зв'язки будуть доступні, залежить від системи координат, яка використовується. У списку властивостей, наведеному нижче, властивості, доступні із проекційною системою координат, позначаються цифрою *1*; властивості, доступні тільки з географічною системою координат, позначаються цифрою *2* (таблиця 3.5).

Таблиця 3.5

Властивість	Пояснення	Тип даних
GCS (тільки читання)	Якщо використовується система координат проекції, повертається об'єкт SpatialReference для географічної системи координат, на якій вона заснована. Якщо використовується географічна система координат, повертається той же самий об'єкт SpatialReference.	SpatialReference
alias (читання і запис)	Псевдонім просторової прив'язки.	String

domain (тільки читання)	Екстент домену xy.	String
name (читання і запис)	Ім'я просторової прив'язки.	String
type (читання і запис)	Тип просторової прив'язки. • Geographic — географічна система координат. • Projected — проекційна система координат	String
latitudeOf1st (читання і запис)	Широта першої точки системи координат проекції. [1]	Double
longitude (читання і запис)	Значення довготи цього початкового меридіана. [1]	Double
PCSCode (читання і запис)	Код системи координат проекції.[1]	Integer
PCSName (читання і запис)	Ім'я системи координат проекції. [1]	String
projectionCode (читання і запис)	Код проекції. [1]	Integer
projectionName (читання і запис)	Ім'я проекції. [1]	String
GCSCode (читання і запис)	Код географічної системи координат. [2]	Integer
GCSName (читання і запис)	Ім'я географічної системи координат. [2]	String
datumCode (читання і запис)	Код датума.[2]	Integer
datumName (читання і запис)	Ім'я датума.[2]	String
spheroidCode (читання і запис)	Код сфероїда.[2]	Integer
spheroidName (читання і запис)	Ім'я сфероїда.[2]	String

Також для отриманого об'єкт *SpatialReference* доступні методи (таблиця 3.6).

Таблиця 3.6

Властивість	Пояснення
create ()	Створює об'єкт просторової прив'язки, використовуючи властивості.
createFromFile (prj_file)	Створює об'єкт просторової прив'язки з файлу проекції.
exportToString ()	Експортує об'єкт в його строкове представлення.

loadFromString (string)	Відновлення об'єкта за допомогою його строкового подання. Метод exportToString може використовуватися для створення строкового подання.
setDomain (x_min, x_max, y_min, y_max)	Задає домен XY.
setFalseOriginAndUnits (false_x, false_y, xy_units)	Задає зміщення відносно початку координат для XY і одиниці вимірювання.
setZDomain (z_min, z_max)	Задає домен Z.
setZFalseOriginAndUnits (false_z, z_units)	Задає зміщення відносно початку координат для Z і одиниці вимірювання.

Таким чином Ви маєте можливість як просто дізнатися про просторову прив’язку для існуючих геопросторових об’єктів так і створити об'єкт просторової прив'язки, використовуючи задані властивості, і присвоїти його об’єкту чи файлу.

Однак, пам’ятайте, що дані властивості і методи не роблять перепроектування, а тільки задають необхідні параметри просторової прив’язки, якщо доступний режим запису.

3.8. Створення власного інструменту

Ми вже побачили як можна запускати власні сценарії, як їх отримувати з ModeBuilder, однак інструменти дозволяють інтегрувати ваші сценарії в ArcGIS.

Інструменти мають діалогове вікно інструментів, яке зазвичай містить параметри, які передаються в сценарій. Розробка інструментів є відносно легкою задачею та значною мірою їх використання покращує досвід користування створених сценаріїв. Діалогові вікна інструментів зменшують кількість помилок у користувачів, оскільки параметри можна вказати за допомогою списків, що розкриваються, перевірити комірки з параметрами та інші механізми.

Такий підхід забезпечує суттєвий контроль над введенням даних користувачем, що значно зменшує потребу писати багато перевірок на перехоплення помилок у коді. Створення інструментів також полегшує спільний доступ до сценаріїв.

Отже, давайте створимо власний інструмент. Що для цього нам потрібно? Перш за все нам необхідний сценарій, який ми будемо використовувати для нашого інструменту. Скористаємося кодом, який наведено на рис. 2.15 після експорту моделі *Find Suitable Land With Parameters* в сценарій Python. Нагадаємо, що ми зберегли його в файл find_suitable_land_param.py. Якщо поглянути на код, то ми побачимо, що у ньому є багато рядків у яких зустрічається метод *GetParameterAsText*.

```
# Script arguments
Cities = arcpy.GetParameterAsText(0)

Cities_buffer_distance = arcpy.GetParameterAsText(1)
if Cities_buffer_distance == '#' or not Cities_buffer_distance:
    Cities_buffer_distance = "10 Miles" # provide a default value if unspecified

Roads = arcpy.GetParameterAsText(2)

Roads_buffer_distance = arcpy.GetParameterAsText(3)
if Roads_buffer_distance == '#' or not Roads_buffer_distance:
    Roads_buffer_distance = "10 Miles" # provide a default value if unspecified

Boundaries = arcpy.GetParameterAsText(4)
```

Фактично, метод *GetParameterAsText* отримує вказаний параметр у вигляді текстового рядка по його індексу позиції зі списку параметрів. Будь-яке значення, незалежно від типу даних параметра, буде повернуто як рядок. Це засіб передачі даних до сценарію через інструмент в ArcGIS.

Синтаксис функції є простим (таблиця 3.7):

```
GetParameterAsText (index)
```

Таблиця 3.7

Властивість	Пояснення	Тип даних
index	Чисельна позиція параметра у списку параметрів.	Integer

Отже index вказаний у методі *GetParameterAsText* визначає який з параметрів сценарію буде отримувати значення першим, а який наступним. Залишається дізнатися як передати значення у відповідний метод.

Насправді це доволі легко зробити, якщо створити власний інструмент у ArcGIS. Для цього треба зробити декілька простих кроків:

1. У вікні *Catalog (каталогу)* знайдіть *Toolboxes Lesson 1.tbx*, який Ви створили раніше і який має бути в розділі *Toolboxes* > *Lesson1*. Або просто створіть новий *Toolbox.*
2. Додайте інструмент до спеціальної панелі інструментів за допомогою майстра додавання сценарію *Add* > *Script* (*рис. 3.7*).

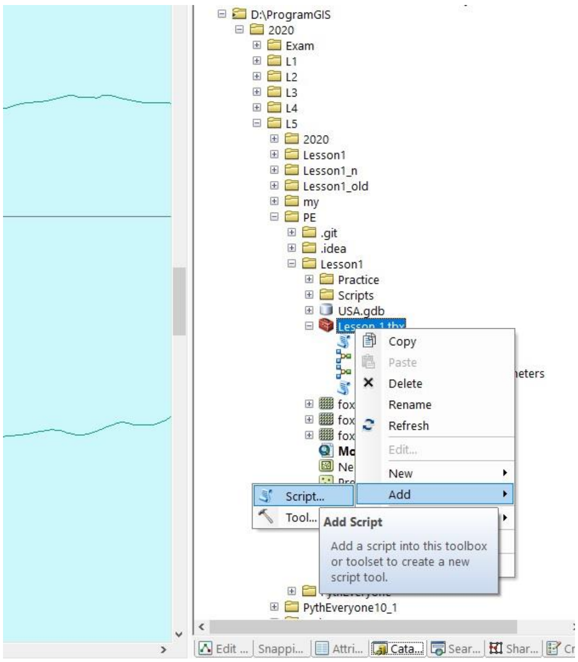


Рис.3.7 Додавання сценарію Python в Toolbox ArcMap

3. Додайте сценарій за допомогою задавання вхідних і вихідних змінних, щоб він був бездоганно інтегрований у структуру геообробки

Отже, як тільки ми запустили майстер додавання сценарію *Add* > *Script* відкриється вікно панелі майстра, як показано на *рис. 3.8*.

Перша панель майстра використовується для задавання *імені інструменту*, його *підпису*, *опису* і *таблиці стилів* наступним чином:

• Назва інструменту використовується, коли Ви забажаєте запустити інструмент з Python. Назва не може містити пробілів чи нижніх підкреслень.

• Підпис інструменту є відображуваною назвою в ArcToolbox. Підпис може містити пробіли

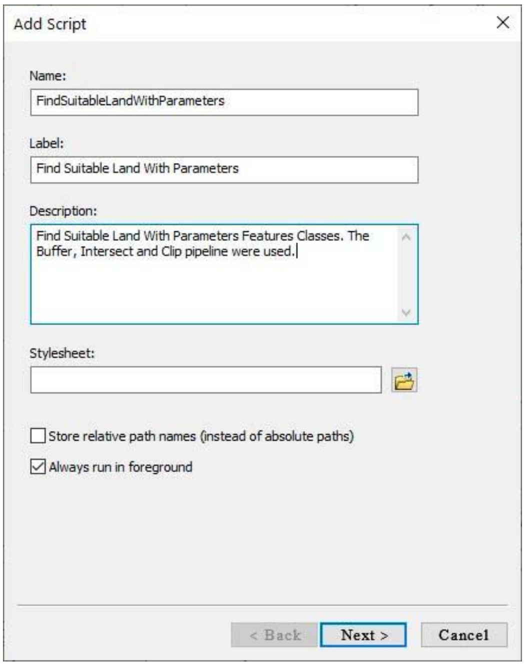


Рис.3.8 Перше вікно панелі майстра Додавання сценарію Python в Toolbox ArcMap

- Опис є необов'язковим полем і використовується для надання власного опису інструменту. Текст автоматично використовується для надання вмісту панелі довідки в діалоговому вікні інструмента.

Друга панель майстра використовується для приєднання сценарію (*рис. 3.9*):

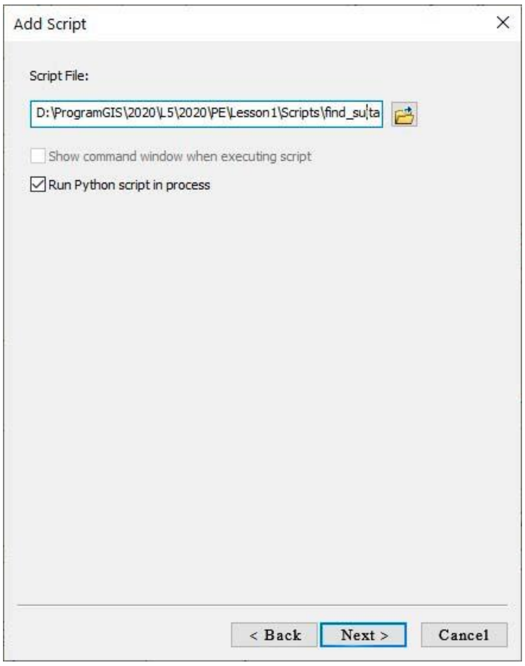


Рис.3.9 Задання шляху до файлу сценарію в панелі майстра Додавання сценарію Python в Toolbox ArcMap

• Отже, необхідно вказати повний шлях до файлу сценарію Python, який потрібно запустити. Ви можете переглядати шлях до наявного файлу або просто вставте шлях до файлу. Якщо Ви наведете шлях до файлу сценарію, якого не існує, то буде запропоновано створити новий (порожній) файл сценарію. Крім того, поле файлу сценарію можна залишити порожнім і додати пізніше.

- Прапорець *Show command window when executing script*, за замовчуванням знято. Коли цей прапорець вибрано, під час виконання інструменту з'являється додаткове вікно, у якому відображаються повідомлення, які не є частиною звичайних повідомлень геообробки, а записуються у стандартний вихід для Python.

- Прапорець *Run Python script in process*, встановлений за замовчуванням. Сценарії Python виконуються швидше, якщо вони виконуються *in process*, тому зазвичай потрібно вибрати цей параметр. Даний режим вимагає щоб модулі Python у вашому сценарії були розроблені для виконання в процесі, це стосується стандартних модулів, таких як os, math і string. Нестандартні модулі сторонніх розробників можуть бути не розроблені для даного режиму, що може призвести до проблем із продуктивністю. Отже, якщо Ви використовуєте сторонні модулі у своєму сценарії, що, здається, призводить до неочікуваних проблем, Ви можете спробувати запустити сценарій в режимі *Show command window when executing script*.

Третя панель майстра додавання сценарію використовується для визначення параметрів інструменту. За замовчуванням параметри не вказані, але більшості інструментів потрібен принаймні один вхідний параметр і один вихідний параметр. Верхня половина панелі дозволяє створювати параметри, а нижня половина дозволяє встановлювати властивості для кожного з цих параметрів (рис. 3.10).

Встановимо параметри для інструменту трохи згодом. Поки ж можна завершити роботу майстра додавання сценарію, для чого натисніть кнопку *Finish* (Готово).

Виконання цього майстра додасть інструмент до вашої спеціальної панелі інструментів.

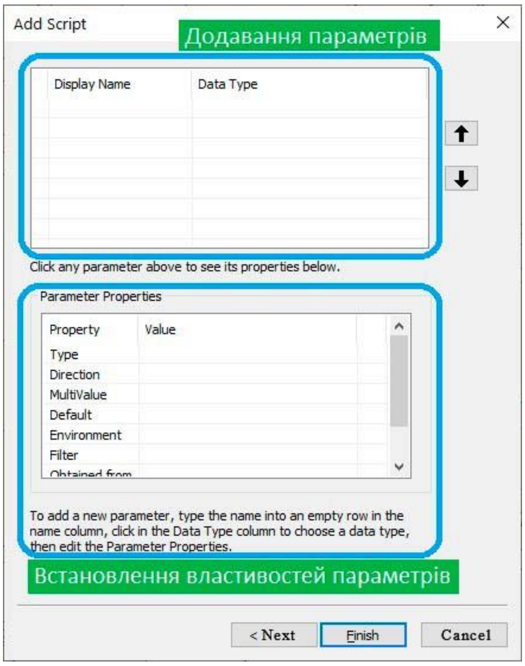


Рис.3.10 Задання параметрів та властивостей сценарію в панелі майстра Додавання сценарію Python в Toolbox ArcMap

Фактично, власний інструмент сценарію готовий, залишилося налаштувати параметри. Для цього оберіть інструмент сценарію в розділі *Toolboxes > Lesson1 > FindSuitableLandWithParameters*. Натисніть правою кнопкою «миші» на ньому і у контекстному меню оберіть *Properties* (Властивості). У вікні властивостей оберіть вкладку *Parameters* (Параметри) в якому налаштуємо параметри сценарію (рис. 3.10).

Ви могли помітити, що функція *GetParameterAsText* зустрічається в нашому коді п’ять разів з індексами від 0 до 4. Відповідно наш інструмент сценарію вимагає налаштування п’яти параметрів. Необхідно додати їх по-черзі і налаштувати.

Треба відмітити, що функція *GetParameterAsText* отримує параметри як текстовий рядок, навіть якщо параметри в діалоговому вікні інструмента мають інший тип даних. Числові значення, логічні значення та інші типи даних перетворюються на рядки, і додається додатковий код для правильної інтерпретації цих рядків.

Призначимо імена і типи даних для наших параметрів. Для цього скористаємося нашими знаннями при створені сценарію (рис. 3.11).

- Cities = arcpy.GetParameterAsText(0) – отже ім’я призначимо *Cities*, а тип даних доберемо *Feature Layer*. Чому *Feature Layer*? Для даного сценарію ми плануємо використовувати шари з фреймів ArcMap, за інших обставин можна обрати потрібний тип параметра.

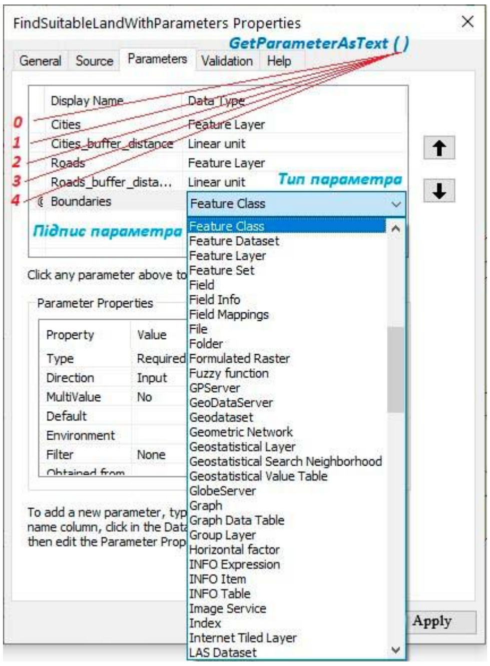


Рис.3.11 Задання підписів параметрам та типу даних для сценарію в панелі майстра Додавання сценарію Python в Toolbox ArcMap

- Cities_buffer_distance = arcpy.GetParameterAsText(1) – отже ім'я призначимо *Cities_buffer_distance*, а тип даних доберемо *Linear unit*. Чому *Linear unit*? Для даного сценарію ми плануємо використовувати відстані для побудови буферів навколо об'єктів, а це числове значення. Для цього можна було використати будь-який числовий тип, наприклад *Double*, але в даному випадку, нам необхідно вказати і в чому вимірюється відстань, тож *Linear unit* більш зручний для цього.
- Roads = arcpy.GetParameterAsText(2) – отже ім'я призначимо *Roads*, а тип даних доберемо *Feature Layer*.
- Roads_buffer_distance = arcpy.GetParameterAsText(3) – отже ім'я призначимо *Roads_buffer_distance*, а тип даних доберемо *Linear unit*.
- Boundaries = arcpy.GetParameterAsText(4) – отже ім'я призначимо *Boundaries*, а тип даних доберемо *Feature Class*. Чому *Feature Class*? Насправді, якщо підходити послідовно до нашого сценарію, то варто було б призначити тип *Feature Layer*, так як ми раніше використовувати шари з фреймів ArcMap. Але, для того щоб Ви не переживали щодо можливості помилитися ми обрали *Feature Class*, не бійтеся експериментувати. До чого призведе вказання типу даних як *Feature Class*? Нам просто доведеться обирати для інструменту Clip не шар а файл, який зберігається в базі даних, або в директорії.

Звісно, що для того аби визначати необхідний тип даних для параметрів, необхідно знати які типи даних підтримує ArcGIS. Тож за необхідності користуйтеся офіційною документацією.

Після призначення типу даних параметрам діалогове вікно інструмента використовує цей тип даних для перевірки значення параметра. Наприклад, якщо Ви введете шлях до елемента іншого типу даних, діалогове вікно інструменту згенерує помилку. Цей вбудований механізм перевірки помилок запобігає використанню користувачами неправильних параметрів для запуску інструменту. Коли інструмент запускається, діалогове вікно вже перевірило параметр, наприклад *Cities*, як шар ознак, і для його перевірки в сценарії не потрібен додатковий код.

Кожен параметр має ряд властивостей, як показано в нижній частині діалогового вікна (рис. 3.12). Коли кожен параметр створюється, для його властивостей встановлюються значення за замовчуванням на основі типу даних параметра. Деякі з ключових параметрів обговоримо нижче. Повний опис усіх параметрів можна знайти в довідці ArcGIS у розділі Налаштування параметрів інструменту сценарію.

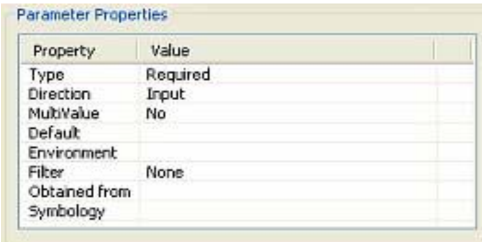


Рис.3.12 Задання властивостей параметрам для сценарію в панелі майстра Додавання сценарію Python в Toolbox ArcMap

- Type (тип) – Існує три варіанти типу: *Required* (обов’язковий), *Optional* (необов’язковий) і *Derived* (похідний). *Required* означає, що для запуску інструменту потрібно вказати значення параметра. *Optional* означає, що значення необов’язкове для запуску сценарію. Як правило, це означає, що вказано значення за умовчанням. *Derived* параметр використовується лише для вихідних (*output*) параметрів і не відображаються в діалоговому вікні інструмента.
- Direction (напрямок) – властивість визначає, чи є параметр *input* (вхідним) для інструменту або *output* (вихідним) для інструменту. Для *Derived* параметрів напрямок параметра буде автоматично встановлено *output*. Кожен інструмент повинен мати *output* параметри. Це робить можливим використання інструменту в ModelBuilder. Хоча технічно сценарій може працювати без *output* параметрів.

- MultiValue (множинне значення) – деякі параметри інструментів складаються зі списку значень, а не з одного значення. Якщо для властивості *MultiValue* встановлено значення *No* (ні), можна використовувати лише одне значення. Якщо для властивості MultiValue встановлено значення Yes (так), то можна використовувати список значень. *MultiValue* параметри досить поширені для вбудованих інструментів геообробки. Наприклад, інструмент *Union*, *Intersect* використовують список вхідних класів об'єктів. *MultiValue* параметр передається до сценарію як рядок із роздільниками за окремими елементами списку, що розділені крапкою з комою (;).
- Default (за замовчуванням) – значення параметра за замовчуванням - це вміст параметра, коли відкрито діалогове вікно інструмента сценарію. Якщо значення за замовчуванням не вказано, значення параметра буде пустим у діалоговому вікні інструмента. Якщо вказано значення за замовчуванням, то властивість Environment буде вимкнено.
- Environment (середовище) – значення для параметра також можна вказати за допомогою властивості Environment. Клацнувши правою кнопкою «миші» клітинку поруч із написом Environment, Ви можете вибрати назву параметра середовища. Якщо цю властивість встановлено, значення за замовчуванням отримується з налаштувань середовища інфраструктури геообробки.
- Filter (фільтр) – властивість дозволяє обмежити значення типів набору даних, які потрібно ввести для параметра. Існує декілька типів фільтрів, і тип залежить від типу даних параметра. Наприклад: *Value List* (список значень), *Range* (діапазон), *Feature Class* (клас просторових об'єктів), *File* (файл), *Field* (поле) та *Workspace* (робоча область).
- Obtained from (отримано з) – у багатьох випадках параметр інструменту тісно пов'язаний з іншими. Наприклад для інструмента *Delete Field* властивість *Obtained from* параметра *Delete Field* можна встановити на вхідну таблицю.
- Symbology (символіка) – за замовчуванням *output* (результат роботи) інструмента геообробки додається до змісту ArcMap. Symbology шару, доданого таким чином, відповідає тим самим правилам, що й у разі додавання даних за допомогою параметра *Add Data* (Додати дані) в ArcMap - іншими словами, налаштування символіки відсутнє. Властивість Symbology можна встановити для файлу спеціального шару (.lyr).

Для нашого інструменту ми залишили всі властивості за замовчуванням. Зокрема властивість Direction встановили у значення *input*, тобто всі параметри будуть вхідними. Тож відмітимо недолік нашого інструменту – він не має вихідних (*output*) параметрів. Варто було б додати. Ви можете зробити, якщо в сценарій додати такий параметр, наприклад для змінної Suitable_land. Це логічно мати вихідний параметр для збереження результатів. Зробіть це самостійно і будьте критичними до свого коду.

Результат запуску готового коду буде схожим як на рис. 3.13.

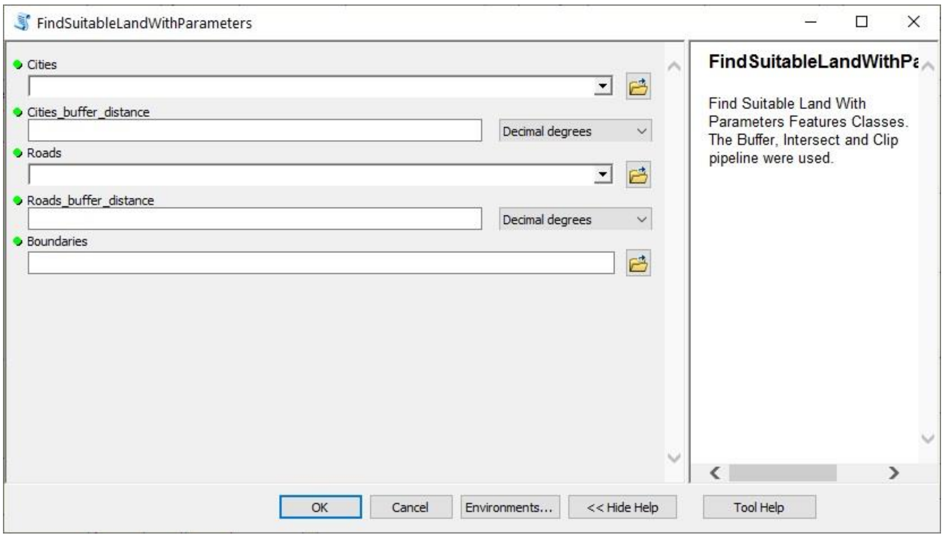


Рис.3.13 Готовий інструмент сценарію для сценарію Python в Toolbox ArcMap

3.9. Робота з повідомленнями інструменту

Під час запуску інструментів пишуться повідомлення про успішне або невдале виконання інструменту. Комунікація між інструментами та користувачами здійснюється за допомогою повідомлень. Типова інформація в цих повідомленнях складається з наступного:

- Точний час початку та завершення роботи інструменту
- Значення параметрів, які використовуються для запуску інструменту
- Інформація про хід роботи інструменту (*information messages* - інформаційні повідомлення)
- Попередження про можливі проблеми під час роботи інструменту (*warning messages* - попереджувальні повідомлення)
- Будь-які помилки, які перешкоджали запуску інструменту (*error messages* - повідомлення про помилки)

Усі повідомлення мають властивість серйозності повідомлення. Ця властивість є цілим числом зі значенням 0 (*інформація*), 1 (*попередження*) або 2 (*помилка*).

Повідомлення з останнього запуску інструменту зберігаються ArcPy і можуть бути отримані за допомогою функції *GetMessages*. Ця функція повертає один рядок, що містить усі повідомлення від останнього запущеного інструменту. Повідомлення можна відфільтрувати, вказавши аргумент серйозності. Детально про функцію *GetMessages* варто дослідити в документації (таблиця 3.8).

GetMessage (index)

Таблиця 3.8

Властивість	Пояснення	Тип даних
index	Отримане повідомлення.	Integer

Кількість повідомлень від останнього запущеного інструменту можна отримати за допомогою функції *GetMessageCount*. Ця функція особливо корисна для отримання лише останнього повідомлення. Оскільки Ви зазвичай не знаєте наперед, скільки повідомлень могло бути отримано в результаті запуску інструменту, Ви можете використовувати кількість повідомлень, щоб отримати останнє повідомлення. Код для отримання підрахунку повідомлень - *arcpy.GetMessageCount()*.

Також корисно і додавати власні інформаційні повідомлення або генерувати виключення в ArcMap. Для цього можна використати методи *AddMessage* та *AddError*. Почитайте про них у офіційній документації.

При запуску інструменту ArcPy доступна інформація про те, з якого додатка він був викликаний. Один з головних ефектів полягає в тому, що Ви можете записувати повідомлення в Python, і ваші повідомлення будуть автоматично з'являтися в діалоговому вікні інструмента, в історії та у вікні Python.

Наприклад:

```
# This script runs the Buffer tool. The user supplies the input and output paths, and the buffer distance.
import arcpy
arcpy.env.overwriteOutput = True
```

```
try:
   # Get the input parameters for the Buffer tool
   inPath = arcpy.GetParameterAsText(0)
   outPath = arcpy.GetParameterAsText(1)
   bufferDistance = arcpy.GetParameterAsText(2)

   # Run the Buffer tool
   arcpy.Buffer_analysis(inPath, outPath, bufferDistance)

   # Report a success message
   arcpy.AddMessage("All done!")
except:
   # Report an error messages
   arcpy.AddError("Could not complete the buffer")

   # Report any error messages that the Buffer tool might have generated
   arcpy.AddMessage(arcpy.GetMessages())
```

Даний сценарій будує буфер навколо вхідних об'єктів. Але, після того як ми приєднаємо інструмент сценарію до нашого коду, ми отримаємо більш якісний інструмент сценарію. І ми ще більше покращимо взаємодію користувача з таким інструментом, якщо будемо виводити повідомлення під його роботи. Це ми і зробили шляхом додавання *AddMessage* та *AddError* до нашого сценарію.

Звернемо увагу ще на один рядок коду

```
arcpy.env.overwriteOutput = True
```

У скриптах поведінка, встановлена за замовчуванням для всіх інструментів, забороняє перезапис вихідних даних, які вже існують. Цю поведінку можна змінити, встановивши для властивості *overwriteOutput* значення True. Спроба виконати перезапис, коли для властивості *overwriteOutput* вибрано значення *False*, завершиться невдачею. Тож врахуйте цей важливий рядок коду при написанні сценаріїв, він може бути корисним.

Питання для самоконтролю:

1. В якому середовищі розробки можна використовувати модуль ArcPy?
2. Чи можна змінити властивості, які є доступними власне Describe Object?

3. Через яку з груп властивостей Describe можна отримати доступ до об’єкту Spatial reference?
4. Якою функцією можна створити об’єкт просторової прив’язки?
5. Що трапиться, якщо Ви наведете шлях до файлу сценарію, якого не існує під час створення власного інструменту сценарію?
6. Як можна передати параметри у функцію GetParameterAsText? На Вашу думку це можна зробити через командний рядок?
7. У чому відмінності між функціями GetMessages та AddMessage?
8. Якщо під час запуску інструменту Ви побили повідомлення з кодом 2, що це означає?

4. ОСНОВИ РОБОТИ З АТРИБУТИВНИМИ ДАНИМИ В ARCPY ВИВЧЕННЯ ПРОСТОРОВИХ ДАНИХ. SEARCHCURSOR

У попередньому розділі ми використовували функцію *Describe* для дослідження просторових даних, включаючи перевірку існування наборів даних і опис наборів даних у робочому просторі.
Однак, окрім такого важливо інструменту є багато інших функцій з якими нам варто ознайомитися для роботи з просторовими даними, і зокрема з атрибутикою даних. Наприклад, функції списку можна використовувати не лише для переліку наборів даних, але й для переліку таких елементів, як робочі області, поля та таблиці.

Списки часто зустрічаються в сценаріях, оскільки вони дають змогу перебирати велику кількість елементів. Вбудовані функції Python можна використовувати для роботи зі списками.

Крім того робота з інструментами вибірки даних може значно посилити роботу ваших сценаріїв.

4.1. Лістинг даних

Пакетна обробка є однією з основних причин розробки сценаріїв геообробки. У попередніх розділах Ви бачили кілька способів використання пакетної обробки. Однак існують обмеження, і сценарії забезпечують більш потужну та гнучку структуру для пакетної обробки.

Одним із ключових завдань у пакетній обробці є інвентаризація доступних даних, щоб сценарій міг перебирати дані під час обробки. ArcPy містить низку функцій для інвентаризації даних. Ці функції повертають *список* Python, який складається зі списку значень. Список може містити будь-який тип даних, але рядки є найпоширенішими під час роботи зі списками ArcPy.

Принцип використання списків полягає в тому, що сценарій може прокручувати значення в списку для роботи з кожним значенням окремо. Для роботи зі списком у Python зазвичай потрібен оператор циклу for. Функції списків ArcPy включають ListFields, ListIndexes, ListDatasets, ListFeatureClasses, ListFiles, ListRasters, ListTables, ListWorkspaces і ListVersions (таблиця 4.1).

Таблиця 4.1

Метод	Пояснення
ListFields(dataset, wild_card, field_type)	Повертає список полів, знайдених у вхідному значенні
ListIndexes(dataset, wild_card)	Повертає список атрибутивних індексів, знайдених у вхідному значенні
ListDatasets(wild_card, feature_type)	Повертає набори даних з поточної робочої області
ListFeatureClasses(wild_card, feature_type, feature_dataset)	Повертає класи просторових об'єктів з поточної робочої області
ListFiles(wild_card)	Повертає файли з поточної робочої області
ListRasters(wild_card, raster_type)	Повертає список растрів, знайдених в поточній робочій області
ListTables(wild_card, table_type)	Повертає список таблиць, знайдених в поточній робочій області
ListWorkspaces(wild_card, workspace_type)	Повертає список робочих областей, знайдених в поточній робочій області
ListVersions(sde_workspace)	Повертає список версій, використання яких дозволено підключеному користувачу

Параметри цих функцій схожі (подивіться детальніше про використання списків в офіційній документації). Деякі функції вимагають вхідний набір даних, оскільки елементи, що нас цікавлять розташовані в наборі даних або об’єкті. Для інших функцій набір даних не потрібен, оскільки елементи, що нас цікавлять розташовані в поточній робочій області. Усі функції мають символ підстановки (*), який визначає фільтр імені. Цей фільтр використовується для перерахування лише елементів, які відповідають певному критерію.

Тобто у всіх функціях є параметр групового символу, який використовується для обмеження імен об'єктів або наборів даних у списку. Груповий символ (*wild_card*) задає ім'я фільтра, який повинен відповідати всьому вмісту нового створеного списку. Наприклад, може бути необхідно перерахувати всі класи просторових об'єктів робочої області, які починаються з букв G.

```
fcs = arcpy.ListFeatureClasses("G*")
```

4.2. Використання параметрів середовища

Параметри середовища - це, по суті, приховані параметри, які впливають на роботу інструменту. Ви вже бачили, як налаштувати середовища в Python за допомогою класу *env*. У цьому розділі розглянемо ці параметри трохи докладніше, оскільки середовища є фундаментальними для керування робочими процесами *геообробки*.

Параметри середовища представлені як властивості класу *env*. Ці властивості можна використовувати для отримання поточних значень або їх встановлення. Кожна властивість має *name* (назву) та *label* (мітку). *Label* відображаються в діалоговому вікні *Environment settings* (Параметри середовища) в ArcGIS, але Python працює лише з іменами.

Синтаксис доступу до властивостей із класу середовища такий

arcpy.env.<environmentName>

Інструменти, які застосовують параметр середовища workspace (поточна робоча область), використовують робочу область, визначену як місце розташування за замовчуванням для вхідних і вихідних даних інструменту геообробки. У ArcMap параметри середовища тимчасової робочої області і поточної робочої області (workspace) синхронізуються за замовчуванням для бази геоданих, яка використовується за замовчуванням – *Defaulf.gdb*. Не варто залишати значення workspace за замовчуванням.

Синтаксис вказання поточної робочої області (workspace) (таблиця 4.2):

arcpy.env.workspace = path

Таблиця 4.2

Метод	Пояснення
path	Місцезнаходження за замовчуванням для вхідних і вихідних даних інструментів геообработки

пакетної обробки. Однак існують обмеження, і сценарії забезпечують більш потужну та гнучку структуру для пакетної обробки.

4.3. Копіювання даних через CopyFeatures

Інколи для роботи потрібно скопіювати дані. Ця операція дозволить використовувати дані для різних проектів без ризику їх втрати при перенесенні проекту.

Для копіювання даних є інструмент *CopyFeatures_management*, який копіює об'єкти із вхідного класу просторових об'єктів або шару до нового класу об'єктів

Також, цей інструмент може використовуватися для конвертації даних, оскільки він може зчитувати безліч векторних форматів (усі, які можна додати на карту) і записувати їх у шейп-файл або базу геоданих.

Дослідимо синтаксис функції *CopyFeatures_management* (таблиця 4.3) та застосуємо її на прикладі.

```
arcpy.management.CopyFeatures(in_features, out_feature_class,
{config_keyword}, {spatial_grid_1}, {spatial_grid_2}, {spatial_grid_3})
```

Таблиця 4.3

Властивість	Пояснення	Тип даних
in_features	Просторові об'єкти, які будуть копіюватися.	Feature Layer; Raster Catalog Layer
out_feature_class	Клас просторових об'єктів, що створюються, в який будуть скопійовані об'єкти. Якщо вихідний клас об'єктів вже існує, і опція перезапису включена, спочатку буде видалений існуючий клас. Якщо вихідний клас об'єктів вже існує, і опція перезапису вимкнена, операція не зможе бути виконана.	Feature Class
config_keyword (додатково)	Якщо вихідні дані записуються в файлову або ArcSDE базу геоданих, можете використовувати ключове слово конфігурації бази геоданих.	String
spatial_grid_1 (додатково)	Параметри Просторова сітка 1, 2 і 3 (Spatial Grid 1, 2, and 3) застосовуються тільки до файлових баз геоданих і до деяких класів просторових об'єктів в базах геоданих робочої групи і корпоративних. Якщо Ви не знайомі з установкою розмірів сітки, залиште ці опції як 0,0,0, і ArcGIS вирахує оптимальні для вас розміри. Щоб дізнатися більше про цей параметр, зверніться до документації інструменту Додати просторовий індекс (Add Spatial Index).	Double

Зауважимо, що якщо вхідні дані є шаром (layer), де обрані об'єкти, будуть копіюватися лише вибрані об'єкти. Якщо вхідні дані є класом об'єктів бази геоданих або шейп-файлом, будуть скопійовані всі об'єкти.

```
# Copies all feature classes from one folder to another
import arcpy

try:
   arcpy.env.workspace = r"D:\ProgramGIS\2020\L5\2020\PE\Lesson1"

   # List the feature classes in the Lesson 1 folder
   fcList = arcpy.ListFeatureClasses()

   # Loop through the list and copy the feature classes to the Lesson 2
PracticeData folder
   for featureClass in fcList:
      arcpy.CopyFeatures_management(featureClass, "
D:\\ProgramGIS\\2020\\L6\\2020\\PE\\Lesson2\\PracticeData\\" + featureClass)

except:
   print ("Script failed to complete")
   print (arcpy.GetMessages(2))
```

Метою даного сценарію є переніс всіх об’єктів, які належать *FeatureClasses* з директорії Lesson 1 до нової директорії Lesson2\PracticeData.

В даному прикладі нам вже повинно бути все знайоме, ми використали arcpy.env.workspace - для того аби задати поточну робочу область, ListFeatureClasses - для того щоб отримати список доступних просторових об'єктів з поточної робочої області. Функцію *CopyFeatures_management* використали власне для копіювання, задавши при цьому їй два обов’язкових параметра.

4.4. Створення копії шаблону просторових об'єктів

Інша задача, яка може зустрітися у розробника це створення порожнього класу просторових об'єктів у розрахованій на багато користувачів, файловій або персональній базі геоданих.

Подібна задача вирішується за допомогою функції *CreateFeatureclass*. Поглянемо на дану функцію з іншого боку.

Уявіть, що Ви працюєте з даними опадів *Nebraska* з директорії Lesson 2, і що Ви хочете створити копію .shp-файлу *Precip2008Readings.shp* протягом наступних 4 років після 2008 року (наприклад, *Precip2009Readings*, *Precip2010Readings* і так далі). По-суті, Ви хочете, скопіювати атрибутивну схему з *Precip2008Readings.shp*, але не самі дані точок. Діні за вимірами для інших років будуть додані пізніше.

Інструментом для автоматизації такого роду операції є створення *CreateFeatureclass* в панелі інструментів *DataManagement*. Подивіться цей інструмент *CreateFeatureclass* в довідковій системі і продивіться його синтаксис і приклади сценаріїв. Зверніть увагу на додатковий параметр шаблону, який дозволяє визначити клас об'єктів, чию атрибутивну схему Ви хочете скопіювати. Також зверніть увагу, що Esri використовує дещо особливий виклик для цього інструменту, і Ви повинні будете викликати arcpy.CreateFeatureclass_management() з використанням нижнього регістра «с» в «class», якщо викоритовується версія ArcGIS 10.6 (таблиця 4.4). У версії ArcGIS 10.8 вже синтаксис arcpy.management.CreateFeatureclass()

CreateFeatureclass_management (out_path, out_name, {geometry_type}, {template}, {has_m}, {has_z}, {spatial_reference}, {config_keyword}, {spatial_grid_1}, {spatial_grid_2}, {spatial_grid_3})

Таблиця 4.4

Властивість	Пояснення	Тип даних
out_path	ArcSDE, файлова або персональна база геоданих, або папка, в якій буде створено вихідний клас об'єктів. Робоча область повинна існувати.	Workspace; Feature Dataset
out_name	Ім'я створюваного класу просторових об'єктів.	String
geometry_type (додатково)	Тип геометрії класу просторових об'єктів. • ТОЧКА (POINT) • MULTIPOINT • ПОЛІГОН (POLYGON) • ПОЛІЛІНІЯ (POLYLINE)	String
template [template,...] (додатково)	Клас просторових об'єктів, використовуваний як шаблон для визначення схеми атрибутів класу просторових об'єктів.	Feature Layer

Отже використаємо *CreateFeatureclass* для поставленої задачі.

```
import arcpy

try:
   arcpy.env.workspace =  r"D:\ProgramGIS\2020\L6\2020\PE\Lesson2"

   template = "Precip2008Readings.shp"

   for year in range(2009,2013):
      newfile = "Precip" + str(year) + "Readings.shp"
      arcpy.CreateFeatureclass_management(arcpy.env.workspace, newfile,
"POINT", template, "DISABLED", "DISABLED", template)

except:
   print (arcpy.GetMessages())
```

Зверніть увагу на значення template для атрибутів *template* та *spatial_reference*. Завдяки їх ми скопіювали атрибутивну схему з Precip2008Readings.shp та його просторову прив’язку.

4.5. Доступ до полів даних

Перш ніж ми надто заглибимося у доступ до векторних даних, буде корисно швидко переглянути, як векторні дані зберігаються у ArcGIS. Векторні об'єкти в класах просторових об'єктів ArcGIS (пам'ятайте, включаючи шейп-файли) зберігаються з використанням атрибутивної таблиці. У таблиці є *rows* (рядки/об’єкти) та *columns* (стовпці/поля/атрибути).

Поля в таблиці зберігають інформацію про геометрію та атрибути об'єктів.

У таблиці є два поля, які ви не можете видалити. Одне з полів (зазвичай називається Shape) містить інформацію про геометрію об'єктів, а друге містить унікальний ID *rows* (рядка/об’єкта). Поле геометрії включає координати кожної вершини в об'єкті і дозволяє рисувати об'єкт на екрані.

Геометрія зберігається у двійковому форматі; якби Ви побачили його надрукованим на екрані, це не мало б для вас жодного сенсу. Однак, Ви можете читати і працювати з геометрією, використовуючи об'єкти, що поставляються разом з ArcPy.

Щодо другого поля, що включене до кожного класу просторових об'єктів, – це поле ідентифікатора об'єкта (OBJECTID або FID). Воно містить унікальний номер або ідентифікатор кожного запису, який використовується ArcGIS для відстеження об'єктів. Ідентифікатор об'єкта допомагає уникнути плутанини під час роботи з даними. Іноді записи мають однакові атрибути, однак, поле OBJECTID ніколи не може мати однакове значення для двох записів.

Інші поля містять атрибутивну інформацію, що описує об'єкт. Ці атрибути зазвичай зберігаються у вигляді чисел чи тексту, хоча можуть бути і рисунки і інші складніші типи.

Розглянемо приклад – ми хочемо отримати список полів у класі об’єктів *Cities* з бази геоданих під назвою USA.gdb.

```
# Reads the fields in a feature class

import arcpy

featureClass =
r"D:\ProgramGIS\2020\L6\2020\PE\Lesson2\PracticeData\USA.gdb\Cities"
fieldList = arcpy.ListFields(featureClass)

# Loop through each field in the list and print the name
for field in fieldList:
   print (field.name)
```

В даному сценарії, ми не вказуємо назви конкретних полів, які ми хотіли прочитати і отримати в доступ, хоча пам’ятайте – це можливо, якщо використати *wild_card*. Отже ми отримали список імен полів Python за допомогою arcpy.ListFields().

Для даного прикладу результат буде таким:

```
OBJECTID
Shape
UIDENT
POPCLASS
NAME
CAPITAL
STATEABB
COUNTRY
```

Але звернемо увагу на рядок коду print (field.name).

ArcPy розглядає поле як об'єкт, тому *Field* (поле) має властивості, які його описують. Ось чому ви можете надрукувати field.name.

У довідковому розділі *Field* (поле) перелічено всі властивості, які можна прочитати з поля (таблиця 4.5). Синтаксис *Field* (об'єкта поле) є максимально простим:

```
Field ()
```

Таблиця 4.5

Властивість	Пояснення	Тип даних
aliasName	Псевдонім поля.	String
baseName	Невідповідне ім'я поля.	String
defaultValue	Значення за замовчуванням для поля.	Variant
length	Довжина поля.	Integer
name	Ім'я поля.	String
type	Тип поля. • Blob –Blob • Date –Date • Double –Double • Geometry –Geometry • GlobalID -Global ID • Guid –Guid • Integer -Integer (Long Integer) • OID -Object ID • Raster –Raster • Single -Single (Float) • SmallInteger -Small Integer (Short Integer) • String -String (Text) Детальніше про типи даних полів ArcGIS	String

Об'єкт поля (поле - Field) представляє стовпець в таблиці. Поле має безліч властивостей, з яких найбільш очевидними є name(ім'я) і type(тип).

Саме назви конкретних полів ми і отримали через властивість name(ім'я).

Доступ до властивостей об'єкта *Field* (поле) можуть бути отриманий через функцію *ListFields* і *Describe* (*Describe.fields*).

Властивості поля доступні для читання і запису, тобто Ви можете дізнатися, якими є властивості поля, і можете змінити ці властивості в сценарії за допомогою об’єкта *Field*. Але, змінюючи властивості поля, змінюється тільки об'єкт *Field* (поле), дійсне поле в таблиці або класі просторових об'єктів залишається без змін.

Наприклад, якщо ви хочете змінити масштаб і точність поля, вам доведеться програмно додати нове поле.

4.6. Додавання та зміна полів даних

Переглянути доступні поля є важливою задачею, але якщо ми бажаємо змінити властивості дійсного поля в таблиці доведеться створити нове поле. І ми розглянемо це пізніше. Е і інша важлива задача - обчислити значення в заданому полі класу об'єктів, векторного шару або растра.

Уявімо, що є задача змінити значення існуючого поля. Одним з шляхів є використання функції *CalculateField*. Саме вона доступна Вам, коли відкривається атрибутивна таблиця шару в ArcGIS.

Синтаксис функції *CalculateField* є наступним (таблиця 4.6):

CalculateField_management (in_table, field, expression, {expression_type}, {code_block})

Таблиця 4.6

Властивість	Пояснення	Тип даних
in_table	Вхідна таблиця, яка містить поля, які будуть оновлені при обчисленні.	Raster Catalog Layer; Mosaic Layer; Raster Layer; Table View
field	Поля, які будуть оновлені при обчисленні.	Field
expression	Простий вираз обчислення, що використовується для створення значення, яке буде внесено в обрані рядки.	SQL Expression
expression_type	Задає тип виразу, що використовується. • VB – вираз буде записано в стандартному форматі VB. Використовується за замовчуванням.	String

	• PYTHON - виражена буде записано в стандартний формат Python. Використовуються ті ж методи і параметри геопроцесора, що і при створенні геопроцесора версії 9.2. • PYTHON_9.3 -вираз буде записано в стандартний формат Python. Використовуються ті ж методи і параметри геопроцесора, що і при створенні геопроцесора версії 9.3. Для успішної роботи з інструментом у 64-розрядних додатках, в тому числі ArcGIS Pro, ArcGIS Desktop, вирази необхідно перетворити в формат Python.	
code_block	Дозволяє введення блоку коду в складні вирази.	String

Подивимося на приклад:

```
import arcpy

fc =
r"D:\ProgramGIS\2020\L6\2020\PE\Lesson2\PracticeData\USA.gdb\CitiesCopy"

# Get a list of field objects
#
fields = arcpy.ListFields(fc, 'CAPITAL')

for field in fields:
    # Check the field name, perform a calculation when finding the field 'Flag'
    #
    if field.name == 'CAPITAL':
        # Set the NEW value for the field and exit loop
        #
        arcpy.CalculateField_management(fc, 'CAPITAL', '1')
        break
```

Попередньо створіть копію класу просторових об’єктів *Cities* в базі даних USA.gdb, та назвіть її *CitiesCopy*. В результаті виконання цього сценарію ми для GDB FeatureClass (класу просторових об’єктів бази даних) *CitiesCopy* для поля 'CAPITAL' змінили всі значення на '1'. Не дуже вдале застосування функції, але такий був наш SQL-вираз. Варто використовувати більш складні вирази, які враховують значення що зберігаються в атрибутивній таблиці. Ми розглянемо як написати більш осмислений SQL-вираз, але трохи пізніше.

Якщо ж ми бажаємо змінити властивості дійсного поля в таблиці доведеться створити нове поле. Це можна зробити за допомогою функції *AddField*.

AddField додає нове поле в таблицю або таблицю класу просторових об'єктів, векторний шар і / або растри з атрибутивними таблицями.

Синтаксис функції *AddField* наступний (таблиця 4.7):

AddField_management (in_table, field_name, field_type, {field_precision}, {field_scale}, {field_length}, {field_alias}, {field_is_nullable}, {field_is_required}, {field_domain})

Таблиця 4.7

Властивість	Пояснення	Тип даних
in_table	Вхідна таблиця, до якої буде додано вказане поле. Поле буде додано до існуючої вхідної таблиці і не створить нової вихідної таблиці. Поля можуть бути додані в класи просторових об'єктів файлових баз геоданих, шейп-файлів, автономних таблиць, растрів з атрибутивними таблицями і / або шарів.	Table View; Raster Layer; Raster Catalog Layer; Mosaic Layer
field_name	Ім'я поля, що додається до вхідної таблиці.	String
field_type	Тип нового поля. • TEXT – Будь-який рядок символів. • FLOAT - Дробові числа між -3.4E38 і 1.2E38. • DOUBLE - Дробові числа між -2.2E308 і 1.8E308. • SHORT - Цілі числа між -32 768 і 32 767. • LONG - Цілі числа між -2 147 483 648 і 2 147 483 647. • DATE - Дата і / або час. • BLOB - Довгий ряд двійкових чисел. Вам буде потрібно використовувати користувацький завантажувач або вьювер, або додаток третіх осіб для завантаження об'єктів в поля типу BLOB або перегляду його вмісту.	String

	• RASTER - Растрове зображення. Можна зберігати всі формати наборів растрових даних, що підтримуються в ArcGIS, але рекомендується для цього поля використовувати тільки маленькі зображення. • GUID-глобальними унікальний ідентифікатор.	
field_precision	Число цифр, що зберігаються в полі. Враховуються всі цифри, незалежно від їх розташування відносно розділювача-коми. Якщо вхідні таблиця є файловою або персональної базою геоданих, значення розрядності поля буде ігноруватися.	Long

Спробуємо вирішити задачу зміни поля з використанням нового поля і функції *CalculateField*.

```
import arcpy

arcpy.CopyFeatures_management(r"D:\ProgramGIS\2020\L6\2020\PE\Lesson2\PracticeData\USA.gdb\CitiesCopy",
r"D:\ProgramGIS\2020\L6\2020\PE\Lesson2\PracticeData\USA.gdb\CitiesCopy2")

fc =
r"D:\ProgramGIS\2020\L6\2020\PE\Lesson2\PracticeData\USA.gdb\CitiesCopy2"

# set local variables
fieldname = "capital_range"
fieldprecision = 3
fieldalias = "range_cap"

# get a list of field objects
fields = arcpy.ListFields(fc)
expression = "getrange(float(!capital!))"
codeblock = """def getrange(capital):
    if capital < 0:
        return 0
    return 1"""

if fieldname not in fields:
```

```
arcpy.AddField_management(fc, fieldname, "short", fieldprecision, field_alias = fieldalias, field_is_nullable = "nullable")
arcpy.CalculateField_management(fc, fieldname, expression, "python_9.3", codeblock)
```

В даному сценарії ми спробували позбутися недоліків попереднього сценарію. Для цього ми створили копію класу просторових об’єктів *Cities* в базі даних USA.gdb, та назвіть її *CitiesCopy2*. Для цього використали функцію *CopyFeatures_management*. Далі переглянули список доступних полів *ListFields* для *CitiesCopy2*. Підготували SQL-вираз у змінній expression – це виклик функції на мові Python, саму ж функцію ми описали у змінній codeblock. Запис подібних виразів є достатньо заплутаним, але якщо пам’ятати про функції Python eval() та exec() то все стане простіше. Фактично ми записали функцію Python у вигляді рядка, не забувайте про відступи для конструкцій, а потім відправили такий рядок на виконання.

Після підготовки було додано поле, яке ми назвали "capital_range" та задали для нього всі параметри, наприклад його тип - "short".

І тільки після всіх цих кроків запускаємо на виконання функцію *CalculateField*.

Однак є і інший спосіб отримати доступ до об’єктів і полів атрибутивної таблиці – за допомогою Cursor (курсор).

Розглянемо цей клас ArcPy послідовно, так як він значно складніший, але і гнучкіший, за *CalculateField*.

4.7. Курсори. SearchCursor

У версії 10.1 ArcMap компанія Esri випустила новий модуль доступу до даних, який запропонував більш високу продуктивність разом із більш надійною поведінкою, коли траплялися помилки з Cursor (курсор). Модуль містить різні класи курсорів для різних операцій, які розробник може забажати виконати над записами таблиці - один для вибору або читання існуючих записів - *SearchCursor*; один для внесення змін до існуючих записів - *UpdateCursor*; і один для додавання нових записів - *InsertCursor*.

Ми обговоримо курсори редагування і оновлення пізніше, зосередивши наше обговорення зараз на курсорі, який використовується для читання табличних даних, *SearchCursor* (курсорі пошуку).

У попередньому розділі Ви бачили, як можна використовувати функції списку для перебору набору значень у списку, включаючи класи об’єктів, таблиці та поля. Аналогічний підхід із використанням курсору використовується для роботи з рядками в таблиці.

Cursor (курсор) - це термін технології бази даних для доступу до набору записів у таблиці. Концептуально це схоже на те, як працюють функції списку, де курсор використовується для керування списком об’єктів. Курсор можна використовувати для оновлення набору об’єктів у таблиці або для вставки нових об’єктів у таблицю. Об’єкти в таблиці також називаються *rows* (рядками), запам’ятаємо це. В ArcGIS курсори можна використовувати для читання та запису геометрії об’єктів, рядок за рядком.

Як і всі класи геообробки, клас *SearchCursor* задокументовано в довідковій системі. (Переконайтеся, що під час пошуку в довідковій системі ви вибрали клас *da.SearchCursor*, який можна знайти в модулі доступу до даних. Старіший клас, до 10.1, доступний через модуль arcpy і відображається в довідковій системі як функція ArcPy SearchCursor).

Синтаксис старішого класу *SearchCursor* виглядає так (таблиця 4.8):

SearchCursor (dataset, {where_clause}, {spatial_reference}, {fields}, {sort_fields})

Таблиця 4.8

Властивість	Пояснення	Тип даних
dataset	Клас просторових об'єктів, шейп-файл або таблиця з рядками для пошуку.	String
where_clause	Додатковий вираз, що обмежує рядки, які повертаються в курсорі. Для отримання більш докладної інформації по SQL виразах запиту where див. в Про побудову SQL-виразів.	String
spatial_reference	Якщо задано, об'єкти будуть проектуватися «на льоту» з використанням наданої **spatial_reference.**	SpatialReference
fields	Поля, які будуть включені в курсор. За замовчуванням включаються всі поля.	String
sort_fields	Поля, які використовуються для сортування рядків в курсорі. Порядок по зростанню і зменшенню для кожного поля позначається, відповідно, літерами **A** і **D**.	String

Синтаксис новішого класу *da.SearchCursor* виглядає так (таблиця 4.9):

SearchCursor (in_table, field_names, {where_clause}, {spatial_reference}, {explode_to_points}, {sql_clause})

Таблиця 4.9

Властивість	Пояснення	Тип даних
in_table	Вхідний клас об'єктів, шар, таблиця або табличні представлення.	String
field_names [field_names,...]	Список (або кортеж) імен полів. Для одного поля можна використовувати рядок замість списку рядків. Доступ до додаткової інформації можна отримати за допомогою токенів (таких як OID@) замість імен полів, наприклад: • SHAPE@XY - Кортеж x, y координат центроїду об'єкта. • SHAPE@X - Значення подвійної точності координати об'єкта. • SHAPE@Y - Значення подвійної точності координати y об'єкта. • SHAPE@JSON - Рядок Esri JSON, що представляє геометрію. • SHAPE@WKB - Стандартне двійкове (well-known binary, WKB) подання для геометрії OGC. Він забезпечує портативне представлення значення геометрії як безперервного потоку байтів. • SHAPE@ - Об'єкт geometry (геометрія) для просторового об'єкта. • OID@ - Значення поля ObjectID.	String
where_clause	Повертається додатковий вираз, який обмежує запис. Докладніше про умови WHERE та вирази SQL див. в Побудова виразу запиту. (За замовчуванням - None)	String
spatial_reference	Просторова прив'язка до класу об'єктів. Її можна вказати з об'єктом SpatialReference або рядковим еквівалентом. (За замовчуванням - None)	SpatialReference
explode_to_points	Розбиває об'єкт окремі точки чи вершини. Якщо для explode_to_points встановлено значення True, то мультиточковий об'єкт із	Boolean

	п'ятьма точками, наприклад, представляється п'ятьма рядками. (За замовчуванням - False)	
sql_clause	Додаткова пара префікса та постфіксу SQL, об'єднана у списку чи кортежі. SQL-префікс підтримує None, DISTINCT та TOP. SQL-постфікс (суфікс) підтримує None, ORDER BY та GROUPBY. Префікс SQL знаходиться в першій позиції і буде вставлений між ключовим словом SELECT та SELECT COLUMN LIST. Префікс виразу SQL найчастіше використовується для таких виразів, як DISTINCT та ALL. Постфікс SQL розташовується в другій позиції і додається до інструкції SELECT після вираження where. Постфікс виразу SQL найчастіше використовується таких виразів, як ORDER BY.	tuple

Функція *SearchCursor* встановлює курсор "тільки для читання" на клас об'єктів або таблицю. *SearchCursor* можна використовувати для циклічної обробки *Row* об'єктів і вилучення значень полів. Пошук можна обмежити умовою *where* або полем і, додатково, впорядкувати.

Курсори arcpy.da (*arcpy.da.SearchCursor*, *arcpy.da.UpdateCursor* і *arcpy.da.InsertCursor*) – є бажаними для застосування, вони були розроблені в ArcGIS 10.1 для істотного збільшення продуктивності в порівнянні з попереднім набором функцій курсора (*arcpy.SearchCursor*, *arcpy.UpdateCursor* і *arcpy.InsertCursor*). Старі версії курсорів поставляються тільки для підтримки сумісності з попередніми версіями.

SearchCursor можуть працювати циклічно за допомогою циклу for або while, з використанням методу курсора next для переходу до наступного рядка. При використанні методу next для курсора з метою вибору всіх рядків таблиці з кількістю рядків n, скрипт повинен виконати кількість викликів n до методу next. Виклик next після останнього рядка в отриманому наборі результатів повертає None.

Кожен рядок, отриманий із таблиці за допомогою курсору, повертається як список значень полів у тому самому порядку, який визначено аргументом *field_names*.

Значення, що повертається - Об'єкт *Cursor*, який може обробляти об'єкти *Row*.

Cursor (Курсор) - це об'єкт доступу до даних, які можуть бути використані як для ітерації по набору рядків в таблиці, так і для вставки нових рядків у таблицю. Як правило, курсори використовуються для зчитування і поновлення атрибутів.

Методи класу *Cursor* (Курсор) виглядають так (таблиця 4.10):

Таблиця 4.10

Метод	**Пояснення**
deleteRow (row)	Видаляє рядок в таблиці. Рядок, що відповідає поточному положенню курсору, буде видалений.
insertRow (row)	Вставляє новий рядок в таблицю.
newRow ()	Створює порожній строковий об'єкт.
next ()	Повертає наступний об'єкт за поточним індексом.
reset ()	Встановлює поточний індекс перерахування (за методом next) назад на перший елемент.
updateRow (row)	Метод updateRow може використовуватися для поновлення рядка в поточному положенні курсору оновлення

Об'єкт *Row* представляє рядок таблиці. Об'єкт *Row* повертається з *InsertCursor*, *SearchCursor* або *UpdateCursor*. Об'єкт *Row* динамічно підтримує імена полів з джерела даних як властивості читання/запису. Імена полів, які не підтримуються безпосередньо як властивість, такі як повні імена полів, що містять точки, можуть бути доступні з використанням методів *setValue* і *getValue*.

Методи класу *Row* виглядають так (таблиця 4.11):

Таблиця 4.11

Метод	**Пояснення**
getValue (field_name)	Отримує значення поля.
isNull (field_name)	Значення поля - null.
setNull (field_name)	Задає значення поля як null.
setValue (field_name, object)	Встановлює значення поля.

Нижче наведено приклад використання курсору пошуку для перебору рядків у таблиці. У наведеному нижче прикладі коду функція *SearchCursor* отримує рядки в таблиці. Цикл for використовується для перебору рядків у таблиці та друку значень для даного поля.

```
import arcpy

fc = r"D:\ProgramGIS\2020\L6\2020\PE\Lesson2\PracticeData
\Pennsylvania.gdb\counties"
populationField = "POP1990"

average = 0
totalPopulation = 0
recordsCounted = 0

# cursor = arcpy.da.SearchCursor(featureClass, (populationField,))

with arcpy.da.SearchCursor(fc, (populationField,)) as cursor:
   for row in cursor:
      totalPopulation += row[0]
      recordsCounted += 1

average = totalPopulation / recordsCounted
print ("Average population for a county is " + str(average))
```

Наведений сценарій для знаходження середнього значення по полю "POP1990" для класу просторових об’єктів *counties* в базі даних Pennsylvania.gdb.

Для доступу до значень поля "POP1990" (рис. 4.1), яке зберігається для кожного об’єкта *counties* було використано *da.SearchCursor*. В якості параметрів було використано два обов’язкових параметра - вхідний клас об'єктів fc, та кортеж полів для доступу (хоча для доступу беремо лише одне поле). row в даному сценарії це об’єкт *Row*, що повернувся з *SearchCursor*. Так як ми задавили для доступу лише одне поле, то в списку полів для *Row* буде лише одне поле, тож і звертаємося до нього за індексом 0 - row[0].

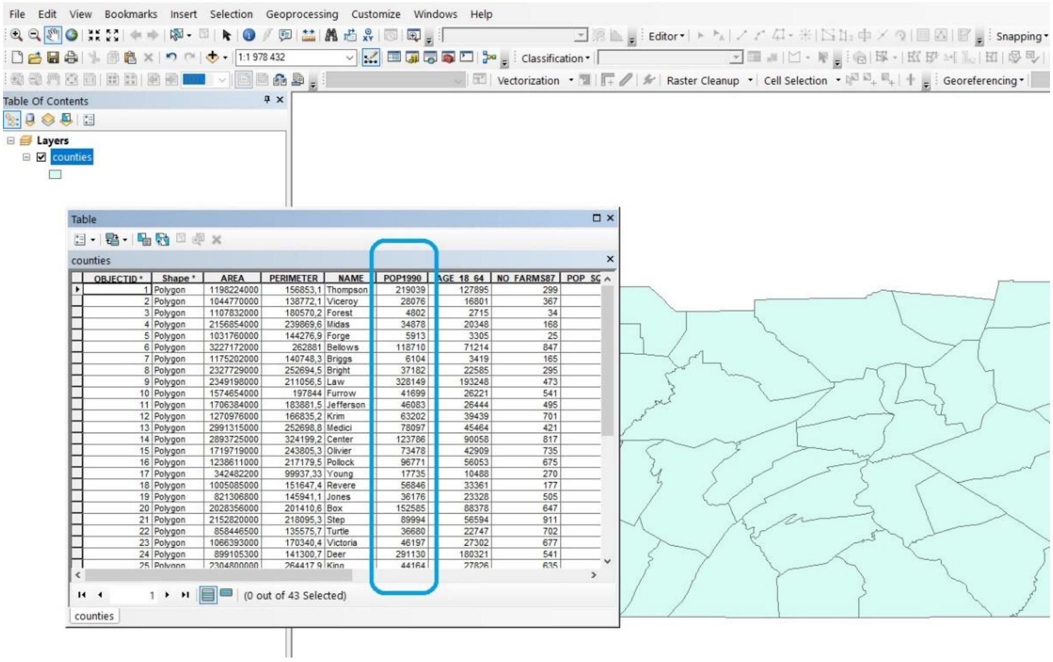


Рис.4.1 Значення поля "POP1990" для сценарію знаходження середнього значення по полю

Одним із найпоширеніших етапів обробки в геообробці є застосування запиту за допомогою мови структурованих запитів (SQL). SQL використовується для визначення одного або кількох критеріїв на основі атрибутів, операторів і обчислень. Наприклад, SQL використовується в діалоговому вікні *Select By Attributes* в ArcMap, а також у багатьох різних інструментах ArcToolbox, включаючи інструмент *Select*.

Додатковий параметр *where_clause* класу *da.SearchCursor* складається з виразу SQL. Синтаксис SQL дещо відрізняється залежно від набору даних, який запитується, і ті самі правила синтаксису застосовуються, коли SQL використовується в параметрах курсору пошуку *da.SearchCursor*.

У наступному коді показано приклад використання виразу SQL:

```
# Finds the average population in a counties dataset

import arcpy

featureClass = r"D:\ProgramGIS\2020\L6\2020\PE\Lesson2\PracticeData
\Pennsylvania.gdb\counties"
populationField = "POP1990"

with arcpy.da.SearchCursor(featureClass, (populationField,) , '"POP1990" >
100000') as cursor:
```

```
for row in cursor:
    print (str(row[0]))
```

Наш виразу SQL - '"POP1990" > 100000'. Тобто нас цікавили не всі значення, я тільки ті які в шуканому полі "POP1990" були б більше за 100000.

Розглянемо як записати подібний запит SQL.

SELECT * FROM становить першу частину виведення SQL і виводиться автоматично.

Виконання запиту використовує загальну форму наступного виду: Select * From <шар або набір даних> Where (наприклад, частина виклику SQL, яка йде після SELECT * FROM <ім’я_шару> WHERE).

Таким чином наш вираз "POP1990" > 100000 є частиною, яке йде після WHERE. Зверніть увагу що виразу SQL який ми таким чином формуємо повинен бути рядкового типу.

Є і інші особливості такого виразу SQL. Це загальна форма для виведення запиту SQL в ArcGIS:

<Ім’я_поля> <Оператор> <Значення або рядок>

Для складних виразів використовується наступна форма:

<Ім’я_поля> <Оператор> <Значення або рядок> <З’єднувач> <Ім’я_поля> <Оператор> <Значення або рядок> ...

Додатково можна використовувати круглі дужки () для визначення порядку операцій в складному запиті.

Роздільники імені поля залежать від СКБД. При запиті до будь-яких файлових даних Ви можете записувати імена полів в подвійні лапки: "POP1990", що ми і зробили в нашому запиті.

Строкові значення завжди записують у виразах в одинарних лапках. наприклад:

STATE_NAME = 'California'

Таким чином загальний вираз запиту до рядкового поля буде, наприклад:

"PARCEL" = 'A2003KSW'

4.8. Отримання записів за допомогою просторового запиту SelectLayer

При застосуванні SQL-виразу для *SearchCursor* виконується фактично вибірка за атрибутом (атрибутивний запит), а не просторовий запит. Наприклад, Ви можете легко знайти за допомогою *SearchCursor* всі округи з назвою *«Lincoln»*, використовуючи вираз SQL, але знайти всі округ, які торкаються або включають річку Міссісіпі Ви не зможете – це вимагає іншого підходу. Для того, щоб отримати набір записів на основі просторового критерію, необхідно використовувати інструмент *Select Layer By Location*.

Припустимо, що Ви хочете вибрати всі штати, чиї кордони торкаються Вайомінгу. У більшості випадків Вам не потрібно буде створити абсолютно новий *feature class* для отримання цих конкретних штатів; Вам, ймовірно, потрібно тільки зберегти ці знайдені записи в пам'яті комп'ютера протягом короткого проміжку часу, в той час як Ви оновлюєте деякі атрибути.

Розглянемо такий приклад. Використання *SearchCursor* для даних за штатом Вайомінг (Wyoming) і знаходження всіх штатів що межують з ним займе «чотири» кроки:

1. Використовуємо *MakeFeatureLayer* для створення *feature layer* (шару просторових об’єктів) всіх штатів США. Назвемо його *AllStatesLayer*.
2. Використовуємо *MakeFeatureLayer* для створення другого *feature layer* тільки для Вайомінгу. (Щоб отримати тільки Вайомінг, Ви повинні застосовувати SQL-вираз при створенні *feature layer*). Назвемо його *SelectionStateLayer*.
3. Використовуємо *SelectLayerByLocation*, щоб обмежити вибір у шарі *AllStatesLayer* (шар, створений на кроці 1) - тільки до тих штатів, які торкаються цільового шару *SelectionStateLayer*.
4. Використовуємо *SearchCursor* для *AllStatesLayer*. Курсор буде включати тільки Вайомінг і штати, які його торкаються, що ї є результатом застосування вибірки до *AllStatesLayer*. Пам'ятайте, що *feature layer* - це просто набір записів, що зберігаються в пам'яті. Навіть якщо Ви назвали його *AllStatesLayer*, він не буде включати всі штати після застосування виділення.

```
# Selects all states whose boundaries touch a user-supplied state
import arcpy
# Get the US States layer, state, and state name field
usaLayer = r"
D:\ProgramGIS\2020\L6\2020\PE\Lesson2\PracticeData\USA.gdb\Boundaries"
state = "Wyoming"
```

```
nameField = "NAME"
try:
   # Make a feature layer with all the US States
   arcpy.MakeFeatureLayer_management(usaLayer, "AllStatesLayer")
   # Make a feature layer containing only the state of interest
   arcpy.MakeFeatureLayer_management(usaLayer, "SelectionStateLayer", '"' +
str(nameField) + '" =' + "'" + str(state) + "'")
   # Apply a selection to the US States layer

arcpy.SelectLayerByLocation_management("AllStatesLayer","BOUNDARY_TOUCH
ES","SelectionStateLayer")
   # Open a search cursor on the US States layer
   with arcpy.da.SearchCursor("AllStatesLayer", (nameField,)) as cursor:
      for row in cursor:
         # Print the name of all the states in the selection
         print (row[0])
except:
   print (arcpy.GetMessages())
# Clean up feature layers
arcpy.Delete_management("AllStatesLayer")
arcpy.Delete_management("SelectionStateLayer")
```

В даному сценарії використовувалася функція *MakeFeatureLayer*, яка створює векторний шар з вхідного класу просторових об'єктів або файлу шару. Якщо шар не зберегти на диск, або не зберегти документ карти (.mxd), то цей тимчасовий шар не буде збережений після завершення сеансу роботи. Ми створили два шари "AllStatesLayer" – без використання SQL-виразу, "SelectionStateLayer" –з використанням SQL-виразу.

Синтаксис *MakeFeatureLayer* є наступним (таблиця 4.12):

arcpy.management.MakeFeatureLayer(in_features, out_layer, {where_clause}, {workspace}, {field_info})

Таблиця 4.12

Властивість	Пояснення	Тип даних
in_features	Вхідний клас об'єктів або шар, з якого створюється новий шар. Класи складних об'єктів, такі як класи анотацій або об'єктів-розмірів, не можуть бути	Feature Layer

	використані в якості вхідних даних для цього інструменту.	
out_layer	Ім'я векторного шару, що створюється. Новий створений шар може використовуватися в якості вхідних даних для інструменту геообробки, який допускає векторний шар в якості вхідних даних.	Feature Layer
where_clause	SQL-вираз, що використовується для вибору піднабору просторових об'єктів. Детальніше про синтаксис SQL див. в Довідка SQL за виразами запитів.	SQL Expression
workspace	Вхідна робоча область використовується для перевірки імен полів.	Workspace; Feature Dataset
field_info	Може використовуватися для перегляду та зміни імен полів, а також, щоб приховати піднабір полів в вихідному шарі.	Field Info

Тимчасовий векторний шар можна зберегти як файл шару за допомогою інструменту *SaveToLayerFile_management* або зберегти в новий клас просторових об'єктів, використовуючи інструмент *CopyFeatures_management*.

Для кроку 3 використовуємо функцію *SelectLayerByLocation*.

SelectLayerByLocation вибирає об'єкти в шарі на основі просторових відносин з об'єктами в іншому шарі. Кожен об'єкт у вхідному векторному шарі (Input Feature Layer) порівнюється з об'єктами в шарі допоміжні об'єкти вибірки (Selecting Features) або класом об'єктів; якщо вхідний об'єкт відповідає значенню в параметрі Relationship (відносини), він вибирається. Вхідними даними повинен бути векторний шар; це не може бути клас просторових об'єктів.

Синтаксис *SelectLayerByLocation* є наступним (таблиця 4.13):

SelectLayerByLocation_management (in_layer, {overlap_type}, {select_features}, {search_distance}, {selection_type}, {invert_spatial_relationship})

Таблиця 4.13

Властивість	Пояснення	Тип даних
in_layer	Шар, що містить об'єкти, які будуть оцінюватися в порівнянні з допоміжними об'єктами вибірки (select_features). До	Feature Layer; Mosaic Layer; Raster Catalog Layer

	цього шару буде застосовуватися вибірка. Вхідними даними не може бути шлях до класу просторових об'єктів на диску.	
overlap_type	Оцінюване просторове відношення. • INTERSECT. • INTERSECT_3D. • WITHIN_A_DISTANCE. • WITHIN_A_DISTANCE_3D. • WITHIN_A_DISTANCE_GEODESIC. • CONTAINS. • COMPLETELY_CONTAINS. • CONTAINS_CLEMENTINI. • WITHIN. • COMPLETELY_WITHIN. • WITHIN_CLEMENTINI. • ARE_IDENTICAL_TO. • BOUNDARY_TOUCHES. • SHARE_A_LINE_SEGMENT_WITH. • CROSSED_BY_THE_OUTLINE_OF. • HAVE_THEIR_CENTER_IN.	String

Питання для самоконтролю:

1. Чи є можливість в ArcPy отримати список доступних в робочій області растрів?
2. Що означає параметр середовища workspace?
3. Яким методом можна скопіювати атрибутивну схему з файла, але не самі дані?
4. Як можна отримати поля атрибутивної таблиці, назва яких починається з «S»?
5. Як отримати поле геометрії для атрибутивної таблиці, якщо використовувати SearchCursor?
6. Що можна записати в поле expression для функції CalculateField_management?
7. Які переваги Ви могли б навести для використання методу MakeFeatureLayer?
8. Сформулюйте SQL-вираз для SearchCursor, якщо буде необхідно знайти співпадіння за текстовим значенням, наприклад "Wyoming"?

5. ІНСТРУМЕНТИ ОНОВЛЕННЯ ТА ДОДАВАННЯ ДАНИХ (UPDATE/INSERTCURSOR). ГЕОМЕТРІЯ ДАНИХ.

Так само, як Ви використовуєте курсори для читання даних векторних атрибутів, також можна використовувати курсори для запису даних. Для запису даних передбачено два типи курсорів:

- Update cursor (Оновити курсор) – цей курсор редагує значення в існуючих записах або видаляє записи
- Insert cursor (Вставити курсор) - цей курсор додає нові записи

У наступних розділах ви дізнаєтесь про обидва ці курсори та отримаєте поради щодо їх використання.

5.1. Оновлення атрибутивних даних. UpdateCursor

UpdateCursor встановлює доступ для читання і запису для об’єктів, що повертається з класу просторових об'єктів або таблиці. Повертається ітератор списків. Порядок значень в списку збігається з порядком полів, зазначених аргументом *field_names*.

Синтаксис *UpdateCursor* є наступним (таблиця 5.1):

UpdateCursor (in_table, field_names, {where_clause}, {spatial_reference}, {explode_to_points}, {sql_clause})

Таблиця 5.1

Властивість	Пояснення	Тип даних
in_table	Вхідний клас об'єктів, шар, таблиця або табличне представлення.	String
field_names [field_names,...]	Список (або кортеж) імен полів. Для одного поля можна використовувати рядок замість списку рядків. Використовуйте символ зірочки (*) замість списку полів, якщо Ви хочете отримати доступ до всіх полів з вхідних таблиці (растрові поля і поля BLOB виключені). Растрові поля не підтримуються.	String

	Доступ до додаткової інформації можна отримати, використовуючи маркери (такі як OID @) замість імен полів: • SHAPE@XY - Кортеж x, y координат центроїда об'єкта. • SHAPE@TRUECENTROID - Кортеж x, y координат істинного центроїда об'єкта. • SHAPE@X - Значення подвійної точності координати x об'єкта. • SHAPE@Y - Значення подвійної точності координати y об'єкта. • SHAPE@ - Об'єктив geometry (геометрія) для просторового об'єкта. • OID @ - Значення поля ObjectID.	
where_clause	Додатковий вираз, що обмежує рядки, які повертаються в курсорі. Детальніше про синтаксис SQL див. в [Довідка SQL за виразами запитів]().	String
sql_clause	Додаткова пара префікса та постфіксу SQL, об'єднана у списку чи кортежі. SQL-префікс підтримує None, DISTINCT та TOP. SQL-постфікс (суфікс) підтримує None, ORDER BY та GROUPBY. Префікс SQL знаходиться в першій позиції і буде вставлений між ключовим словом SELECT та SELECT COLUMN LIST. Префікс виразу SQL найчастіше використовується для таких виразів, як DISTINCT та ALL. Постфікс SQL розташовується в другій позиції і додається до інструкції SELECT після вираження where. Постфікс виразу SQL найчастіше використовується таких виразів, як ORDER BY.	tuple

UpdateCursor є доступними і методи класу (таблиця 5.2):

Таблиця 5.2

Метод	**Пояснення**
deleteRow (row)	Видаляє рядок в таблиці. Рядок, що відповідає поточному положенню курсору, буде видалений.
next ()	Повертає наступний об'єкт за поточним індексом.

reset ()	Встановлює поточний індекс перерахування (за методом next) назад на перший елемент.
updateRow (row)	Метод updateRow може використовуватися для поновлення рядка в поточному положенні курсору оновлення

А властивість, що дозволяє оперувати полями атрибутивної таблиці є одна (таблиця 5.3)

Таблиця 5.3

Властивість	Пояснення	Тип даних
fields	Список (або кортеж) імен полів, використовуваних курсором. У кортеж будуть включені всі поля (і маркери), зазначені аргументом field_names. Якщо аргумент field_names матиме значення "*", то в якості полів будуть включатися всі поля, використовувані курсором. Коли використовується "*", значення геометрії будуть виводитися в кортежі координат x і y (еквівалент токені SHAPE@XY).	tuple

Розглянемо приклад використання *UpdateCursor*. Припустимо, у вас є база геоданих *Washington.gdb* з двома класами об'єктів:

- *CityBoundaries* - містить полігональні кордони міст в штаті Вашингтон площею понад 10 квадратних кілометрів.

- *ParkAndRide* - Містить точкові об'єкти, що представляють паркінги і стоянки, де водії можуть залишати свої автомобілі.

Ми бажаємо дізнатися, в яких містах є паркінги і стоянки, і який відсоток міст має хоча б одну стоянку чи паркінг.

Клас просторових об'єктів (*feature classes*) *CityBoundaries* має поле *HasParkAndRide*, яке за замовчуванням встановлено в значення «False». Наше завдання - визначити це поле як «True» для кожного міста, що містить принаймні один паркінг чи стоянку в межах його кордонів.

Також, наш сценарій повинен розрахувати відсоток міст, що мають паркінг чи стоянку, і роздрукувати цей показник для користувача.

Але врахуйте, що це припущення, що кожна точка в наборі даних *ParkAndRide* представляє один дійсний паркінг чи стоянку (ігноруємо значення в поле TYPE).

```
# This script determines the percentage of cities in the state with park and ride
facilities
import arcpy
arcpy.env.overwriteOutput = True
cityBoundaries=r"
D:\ProgramGIS\2020\L7\2020\PE\Lesson3\\Lesson3PracticeExerciseA\Washingto
n.gdb\CityBoundaries"
parkAndRide=r"
D:\ProgramGIS\2020\L7\2020\PE\Lesson3\\Lesson3PracticeExerciseA\Washingto
n.gdb\ParkAndRide"
parkAndRideField = "HasParkAndRide"    # Name of column with Park & Ride
information
citiesWithParkAndRide = 0     # Used for counting cities with Park & Ride

try:
   # Make a feature layer of all the park and ride facilities
   arcpy.MakeFeatureLayer_management(parkAndRide, "ParkAndRideLayer")
   # Make a feature layer of all the cities polygons
   arcpy.MakeFeatureLayer_management(cityBoundaries, "CitiesLayer")
except:
   print ("Could not create feature layers")

try:
   # Narrow down the cities layer to only the cities that contain a park and ride
   arcpy.SelectLayerByLocation_management("CitiesLayer", "CONTAINS",
"ParkAndRideLayer")
   # Create an update cursor and loop through the selected records
   with arcpy.da.UpdateCursor("CitiesLayer", (parkAndRideField,)) as cursor:
      for row in cursor:
         # Set the park and ride field to TRUE and keep a tally
         row[0] = "True"
         cursor.updateRow(row)
         citiesWithParkAndRide +=1

# Delete the feature layers even if there is an exception (error) raised
finally:
   arcpy.Delete_management("ParkAndRideLayer")
```

```
    arcpy.Delete_management("CitiesLayer")

# Count the total number of cities (this tool saves you a loop)
numCitiesCount = arcpy.GetCount_management(cityBoundaries)
numCities = int(numCitiesCount.getOutput(0))

# Calculate the percentage and print it for the user
percentCitiesWithParkAndRide = ((1.0 * citiesWithParkAndRide) / numCities) *
100
print (str(percentCitiesWithParkAndRide) + " percent of cities have a park and
ride.")
```

В даному сценарії ми використали функцію *MakeFeatureLayer* для перетворення даних у шари, адже це тимчасові шари тільки на час виконання скрипта, це розумний підхід. Тож в *UpdateCursor* ми використовували саме шари.

Поле для оновлення було використане тільки одне *parkAndRideField*, тож і доступ до нього ми отримали із сприску доступних полів через row[0].

Власне оновлення поля робиться методом *updateRow(row)*.

Звернемо увагу на метод GetCount_management, він повертає загальну кількість рядків у таблиці. Варто ознайомитися з цим методом в документації.

5.2. Додавання нових записів. InsertCursor

InsertCursor створює курсор для запису в класі об'єктів або таблиці. *InsertCursor* можна використовувати для додавання нових рядків.

При використанні *InsertCursor* в класі точкових об'єктів створення *PointGeometry* і установка токена SHAPE@ є досить витратною операцією. Для більш швидкого і ефективного доступу можна визначити точковий об'єкт за допомогою таких токенов, як SHAPE@XY, SHAPE@Z і SHAPE@M.

Синтаксис *InsertCursor* є наступним (таблиця 5.4):

```
InsertCursor (in_table, field_names)
```

Таблиця 5.4

Властивість	Пояснення	Тип даних
in_table	Вхідний клас об'єктів, шар, таблиця або табличне представлення.	String
field_names [field_names,...]	Список (або кортеж) імен полів. Для одного поля можна використовувати рядок замість списку рядків. Використовуйте символ зірочки (*) замість списку полів, якщо Ви хочете отримати доступ до всіх полів з вхідних таблиці (растрові поля і поля BLOB виключені). Растрові поля не підтримуються. Доступ до додаткової інформації можна отримати, використовуючи маркери (такі як OID @) замість імен полів: • SHAPE@XY - Кортеж x, y координат центроїда об'єкта. • SHAPE@TRUECENTROID - Кортеж x, y координат істинного центроїда об'єкта. • SHAPE@X - Значення подвійної точності координати x об'єкта. • SHAPE@Y - Значення подвійної точності координати y об'єкта. • SHAPE@ - Об'єктив geometry (геометрія) для просторового об'єкта. • OID @ - Значення поля ObjectID.	String

InsertCursor є доступними і методи класу (таблиця 5.5):

Таблиця 5.5

Метод	**Пояснення**
insertRow (row)	Вставляє рядок в таблицю.

А властивість, що дозволяє оперувати полями атрибутивної таблиці є одна (таблиця 5.6)

Таблиця 5.6

Властивість	Пояснення	Тип даних
fields	Список (або кортеж) імен полів, використовуваних курсором. У кортеж будуть включені всі поля (і маркери),	tuple

	зазначені аргументом field_names. Якщо аргумент field_names матиме значення "*", то в якості полів будуть включатися всі поля, використовувані курсором. Коли використовується "*", значення геометрії будуть виводитися в кортежі координат x і y (еквівалент токені SHAPE@XY).	

Робота з *InsertCursor* є досить простою, якщо мова йде про додавання нових записів. Наприклад, якщо ми забажаємо додати до класу просторових об’єктів *CitiesCopy2* в базі даних USA.gdb новий рядок, то це буде виглядати так:

```
#add new field in feature class
import arcpy

fc = r"
D:\ProgramGIS\2020\L6\2020\PE\Lesson2\PracticeData\USA.gdb\CitiesCopy2"
newField = ["NAME","CAPITAL"]

with arcpy.da.InsertCursor(fc, newField) as cursor:
	cursor.insertRow(["Silenthill",3])
	del cursor
print ("All done!")
```

Сценарій є простим і звертаємо увагу, що в класі просторових об’єктів *CitiesCopy2* ми створили запис, але додали значення тільки в два поля "NAME" та "CAPITAL". Поле геометрії залишилося пустим, тож і в фреймі ArcMap ми не побачимо такого точкового об’єкту. Отже виникає необхідність розібратися з геометрією об’єктів.

5.2. Робота з геометрією

Кожен об’єкт у класі об’єктів містить набір точок, які визначають вершини об’єкта. Доступ до цих точок можна отримати за допомогою геометричних об’єктів, таких як *Point*, *Polyline*, *PointGeometry* та *MultiPoint*, який повертає масив об’єктів *Point*.

Доступ до повних геометричних об’єктів *Geometry* займає дещо багато часу. У результаті сценарії, які працюють з об’єктами повної геометрії великих наборів даних, можуть стати дуже повільними. Якщо вам потрібні лише певні властивості геометрії, ви можете використовувати токени геометрії як ярлики для доступу до властивостей геометрії. Наприклад, SHAPE@XY поверне кортеж координат (x, y), що представляють центроїд об’єкта, а SHAPE@LENGTH поверне довжину об’єкта типу double. З іншого боку, SHAPE@ поверне повний геометричний об’єкт *Geometry*.

Корисно почати роботу з геометрією, якщо подивитися на роботу SearchCursor, якщо спробуємо отримати поле ID та геометрії:

```
# Finds the geometry of point in the cities
import arcpy

featureClass = r"
D:\ProgramGIS\2020\L6\2020\PE\Lesson2\PracticeData\USA.gdb\CitiesCopy2"
searchedField = ["OID@","SHAPE@XY"]

# Loop through each row and keep all geometry for it

with arcpy.da.SearchCursor(featureClass, searchedField) as cursor:
   for row in cursor:
      print "{0} : X={1} Y={2}".format(row[0],row[1][0],row[1][1])
   print "Coodrinate system: " +
arcpy.Describe(featureClass).spatialReference.name
   print "Datum name: " +
arcpy.Describe(featureClass).spatialReference.datumName
```

Як результат ми отримаємо для кожного об’єкта ID та координати точок. Так як за запитом було два параметри, то до ID ми отримаємо доступ за row[0], а до геометрії за row[1]. Відповідно якщо нам цікаво отримати x - row[1][0], y - row[1][1].

Ми побачило, що для даного прикладу поле геометрії повернуло нам кортежі координат (x, y). Чи можна скористатися цим знанням аби записати в поле геометрії координати для прикладу, коли ми додавали нові записи, але не додали саме геометріє для об’єктів? Так, подивіться на наступний сценарій:

```
# Insert the geometry of point in the cities
import arcpy
arcpy.env.workspace = r"
D:\ProgramGIS\2020\L6\2020\PE\Lesson2\PracticeData\USA.gdb "
featureClassList = arcpy.ListFeatureClasses()
featureClass = " CitiesCopy3"
if featureClass not in featureClassList:
```

```
    arcpy.CreateFeatureclass_management(arcpy.env.workspace, featureClass,
"POINT", "Cities", "DISABLED", "DISABLED", "Cities")
    print "Temp was created"
insertedField = ["SHAPE@XY"]
# Define array for points
xy = [(-97.7624204537, 28.3709213515),
      (-80.1164167102, 26.3299831143),
      (-81.8008128002, 26.6600164123),
      (-80.1358864047, 26.1503917519),
      (-80.3203965408, 25.9088593326),
      (-81.7737622983, 26.1454599642),
      (-80.2244915167, 25.7804603874),
      (-97.4270261477, 27.757033849)]
# Loop through each row and insert all geometry for it
with arcpy.da.InsertCursor(featureClass,insertedField) as cursor:
    for row in xy:
        cursor.insertRow([row])
print "All done!"
```

Однак, для того аби робити додавання геометрії більш коректним, варто розібратися, що собою представляє геометрія в ArcMap.

Об'єкти *Geometry* визначають просторове положення і пов'язану з ним геометричну форму. У багатьох робочих процесах геообробки може знадобитися запустити певну операцію, використовуючи інформацію про координати і геометрію, при цьому не бажаючи проходити процес створення нового (тимчасового) класу просторових об'єктів, відображення класу просторових об'єктів за допомогою курсорів, використання класу просторових об'єктів, і подальшого видалення тимчасового класу просторових об'єктів.

Щоб спростити геообробку, замість цього в якості вхідних і вихідних даних можна використовувати об'єкти геометрії. Об'єкти геометрії можуть створюватися спочатку, використовуючи класи *Geometry*, *Multipoint*, *PointGeometry*, *Polygon*, або *Polyline*.

Синтаксис Geometry є наступним (таблиця 5.7):

```
Geometry  (geometry, inputs, {spatial_reference}, {has_z}, {has_m})
```

Таблиця 5.7

Параметр	Пояснення	Тип даних
geometry	Тип геометрії: точка, полігон, полілінія або мультиточка.	String
inputs	Координати, що використовуються для створення об'єкта. Тип даних може бути Point або Array об'єкти.	Object
spatial_reference	Просторова прив'язка нової геометрії. (Значення за замовчуванням - None)	SpatialReference
has_z	Статус Z: True для геометрії, якщо Z використовуються, і False - якщо немає. (Значення за замовчуванням - False)	Boolean
has_m	Статус M: True для геометрії, якщо M використовуються, і False - якщо немає. (Значення за замовчуванням - False)	Boolean

Всі властивості *Geometry* є тільки для читання (таблиця 5.8)

Таблиця 5.8

Властивість	Пояснення	Тип даних
JSON	Повертає представлення геометрії Esri JSON у вигляді рядка. Рядок, що повертається може бути конвертований в словник за допомогою функції Python json.loads.	String
WKB	Повертає WKB- представлення для геометрії OGC. Він забезпечує портативне представлення значення геометрії у вигляді безперервного потоку байтів.	Bytearray
WKT	Повертає WKT- представлення для геометрії OGC. Він забезпечує портативне представлення значення геометрії у вигляді текстового рядка. Будь-які справжні криві в геометрії будуть ущільнені в апроксимуючі криві в рядки WKT.	String
area	Площа об'єкта полігону. Пусто для всіх інших типів об'єктів.	Double
centroid	Повертається справжній стан центроїда, якщо він розташований всередині об'єкта, інакше повертається точка напису. Повертає точковий об'єкт.	Point
extent	Екстент геометрії.	Extent
firstPoint	Перша точка координат геометрії.	Point

hullRectangle	Рядок пар координат прямокутника, що утворює опуклу оболонку, розділені пробілами.	String
isMultipart	True, якщо кількість частин для цієї геометрії більше однієї.	Boolean
labelPoint	Точка розташування напису. labelPoint завжди розташований всередині об'єкта або на ньому.	Point
lastPoint	Остання координата об'єкта.	Point
length	Довжина лінійного об'єкта. Нуль для точкових і багатоточкових об'єктів.	Double
length3D	3D-довжина лінійного об'єкта. Нуль для точкових і багатоточкових об'єктів.	Double
partCount	Кількість частин просторового об'єкта.	Integer
pointCount	Загальна кількість точок об'єкта.	Integer
spatialReference	Просторова прив'язка геометрії.	SpatialReference
trueCentroid	Центр тяжіння просторового об'єкта.	Point
type	Тип геометрії: полігон, полілінія, точка, мультиточки, мультіпатч, розмір або анотація.	String

Зверніть увагу на властивість *geometry*, саме вона визначає геометрію для об’єкта і я кою вона буде *Point*, *Polyline*, *PointGeometry* чи *MultiPoint*.

Отже для того аби додати геометрію для точкових об’єктів треба створити об’єкт геометрії *Point*. У минулому прикладі ми такого не робили, ми створили просто список з кортежів координат (x, y). Наскільки це важливо створити саме об’єкт геометрії *Point*? Якщо необхідно створювати більш складну геометрію ніж точкову, то це є необхідним кроком.

Об'єкт *Point* часто використовується з курсором. Точкові об'єкти замість масиву точкових об'єктів повертають єдиний об'єкт *Point*. Всі інші типи об'єктів - *Polyline*, *PointGeometry* чи *MultiPoint* - повертають масив точкових об'єктів.

Point не є класом геометрії, але зазвичай використовується для побудови геометрії. У наведеному нижче прикладі, *Point* використовується для створення об'єкта *PointGeometry*.

```
point = arcpy.Point(25282, 43770)
```

```
ptGeometry = arcpy.PointGeometry(point)
```

Розглянемо що собою представляє об’єкт геометрії *Point* (таблиця 5.9):

```
Point ({X}, {Y}, {Z}, {M}, {ID})
```

Таблиця 5.9

Параметр	Пояснення	Тип даних
X	X - координата точки. (Значення за замовчуванням - 0.0)	Double
Y	Y - координата точки. (Значення за замовчуванням - 0.0)	Double
Z	Z - координата точки. (Значення за замовчуванням - None)	Double
M	M - значення точки. (Значення за замовчуванням - None)	Double
ID	ID форми точки. (Значення за замовчуванням - 0)	Integer

Всі властивості *Point* є для читання і запису (таблиця 5.10)

Таблиця 5.10

Властивість	Пояснення	Тип даних
ID	Ціле число, яке використовується для унікальної ідентифікації точки	Integer
M	Значення вимірювання в точці.	Double
X	Горизонтальна координата точки.	Double
Y	Вертикальна координата точки.	Double
Z	Значення висоти точки.	Double

Подивимося на прикладі як працювати з *Point*. Створимо власний модуль practiceModule1.py

```
# This module is saved as practiceModule1.py
# The function below creates points from a list of coordinates
# Example list: [[-113,23][-120,36][-116,-2]]

def createPoints(coordinateList, featureClass):

    # Import arcpy and create an insert cursor
    import arcpy

    with arcpy.da.InsertCursor(featureClass, ("SHAPE@",)) as rowInserter:

        # Loop through each coordinate in the list and make a point
        for coordinate in coordinateList:
            point = arcpy.Point(coordinate[0],coordinate[1])
            rowInserter.insertRow((point,))
```

Метод *createPoints(coordinateList, featureClass)* буде приймати на вхід два параметри – список координат (x,y) та назву класу просторових об'єктів до якого необхідно додати геометрію.

Далі ж використаємо розроблений модуль:

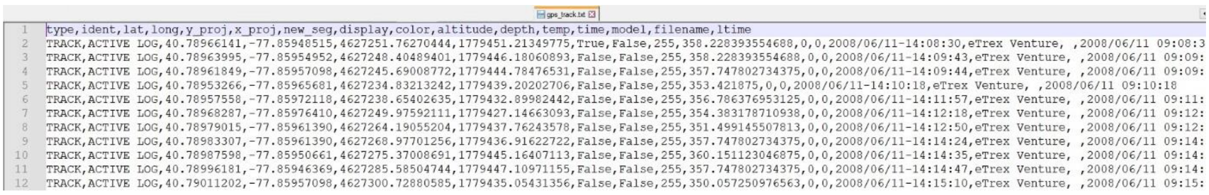
```
# This script is saved as addPointGeometry.py
# Import the module containing a function we want to call
import practiceModule1

# Define point list and shapefile to edit
myWorldLocations = [[-123.9,47.0],[-118.2,34.1],[-112.7,40.2],[-63.2,-38.7]]
myWorldFeatureClass = r"
D:\ProgramGIS\2020\L6\2020\PE\Lesson2\PracticeData\USA.gdb\CitiesCopy2"

# Call the createPoints function from practiceModule1
practiceModule1.createPoints(myWorldLocations, myWorldFeatureClass)
```

Використання більш складної геометрії схоже на те, як створювалися точкові об'єкти. Але відмінність у тому, що об'єкти *Polyline*, *PointGeometry* та *MultiPoint* складаються з об'єктів *Point*.

Подивимося на подібне використання на прикладі - зчитаємо текстовий файл, зібраний із пристрою GPS. Рядки у файлі представляють показання, отримані з пристрою GPS під час руху користувача по шляху. Розглянемо один спосіб аналізу координат із кожного прочитаного запису (рис. 5.1).



```
type,ident,lat,long,y_proj,x_proj,new_seg,display,color,altitude,depth,temp,time,model,filename,ltime
TRACK,ACTIVE LOG,40.78966141,-77.85948515,4627251.76270444,1779451.21349775,True,False,255,358.228393554688,0,0,2008/06/11-14:08:30,eTrex Venture, ,2008/06/11 09:08:3
TRACK,ACTIVE LOG,40.78963995,-77.85954952,4627248.40489401,1779446.18060893,False,False,255,358.228393554688,0,0,2008/06/11-14:09:43,eTrex Venture, ,2008/06/11 09:09:
TRACK,ACTIVE LOG,40.78961849,-77.85957098,4627245.69008772,1779444.78476531,False,False,255,357.747802734375,0,0,2008/06/11-14:09:44,eTrex Venture, ,2008/06/11 09:09:
TRACK,ACTIVE LOG,40.78953266,-77.85965681,4627234.83213242,1779439.20202706,False,False,255,353.421875,0,0,2008/06/11-14:10:18,eTrex Venture, ,2008/06/11 09:10:18
TRACK,ACTIVE LOG,40.78957558,-77.85972118,4627238.65402635,1779432.89982442,False,False,255,356.786376953125,0,0,2008/06/11-14:11:57,eTrex Venture, ,2008/06/11 09:11:
TRACK,ACTIVE LOG,40.78968287,-77.85976410,4627249.97592111,1779427.14663093,False,False,255,354.383178710938,0,0,2008/06/11-14:12:18,eTrex Venture, ,2008/06/11 09:12:
TRACK,ACTIVE LOG,40.78979015,-77.85961390,4627264.19055204,1779437.76243578,False,False,255,351.499145507813,0,0,2008/06/11-14:12:50,eTrex Venture, ,2008/06/11 09:12:
TRACK,ACTIVE LOG,40.78983307,-77.85961390,4627268.97701256,1779436.91622722,False,False,255,357.747802734375,0,0,2008/06/11-14:14:24,eTrex Venture, ,2008/06/11 09:14:
TRACK,ACTIVE LOG,40.78987598,-77.85950661,4627275.37008691,1779445.16407113,False,False,255,360.151123046875,0,0,2008/06/11-14:14:35,eTrex Venture, ,2008/06/11 09:14:
TRACK,ACTIVE LOG,40.78996181,-77.85946369,4627285.58504744,1779447.10971155,False,False,255,357.747802734375,0,0,2008/06/11-14:14:47,eTrex Venture, ,2008/06/11 09:14:
TRACK,ACTIVE LOG,40.79011202,-77.85957098,4627300.72880585,1779435.05431356,False,False,255,350.057250976563,0,0,2008/06/11-14:15:10,eTrex Venture, ,2008/06/11 09:15:
```

Рис.5.1 Приклад csv файлу отриманого для GPS треку, який будемо аналізувати

```
# This script reads a GPS track in CSV format and writes geometries from the list
of coordinate pairs
import csv
import arcpy
import os

# Set up input and output variables for the script
gpsTrack = open(r"D:\ProgramGIS\2020\L7\2020\PE\Lesson3\gps_track.txt")
```

```
polylineFC = r"D:\ProgramGIS\2020\L7\2020\PE\Lesson3\tracklines.shp"
if not os.path.isfile(polylineFC):
    from arcpy import env
    env.workspace = r"D:\ProgramGIS\2020\L7\2020\PE\Lesson3"
    arcpy.CreateFeatureclass_management(env.workspace, "tracklines.shp",
"POLYLINE", "", "DISABLED", "DISABLED", r"
D:\ProgramGIS\2020\L7\2020\PE\Lesson3\counties.shp")
spatialRef = arcpy.Describe(polylineFC).spatialReference

# Set up CSV reader and process the header
csvReader = csv.reader(gpsTrack)
header = csvReader.next()
latIndex = header.index("lat")
lonIndex = header.index("long")

# Create an empty array object
vertexArray = arcpy.Array()

# Loop through the lines in the file and get each coordinate
for row in csvReader:
    lat = float(row[latIndex])
    lon = float(row[lonIndex])
    # Make a point from the coordinate and add it to the array
    vertex = arcpy.Point(lon,lat)
    vertexArray.add(vertex)

# Write the array to the feature class as a polyline feature
with arcpy.da.InsertCursor(polylineFC, ("SHAPE@",)) as cursor:
    polyline = arcpy.Polyline(vertexArray, spatialRef)
    cursor.insertRow((polyline,))

print ("All done!")
```

Ви можете побачити, що ми зчитали масив координат з csv-файлу і використали їх для створення точкових об’єктів - *Point(lon,lat)*, і тільки після цього створили об’єкт *Polyline* з точкових об’єктів *Point* - *Polyline(vertexArray, spatialRef)*. Звернемо увагу і геопросторову прив’язку для створеної *Polylin*, не забувайте про неї.

Результат роботи сценарію можна побачити на рис. 5.2.

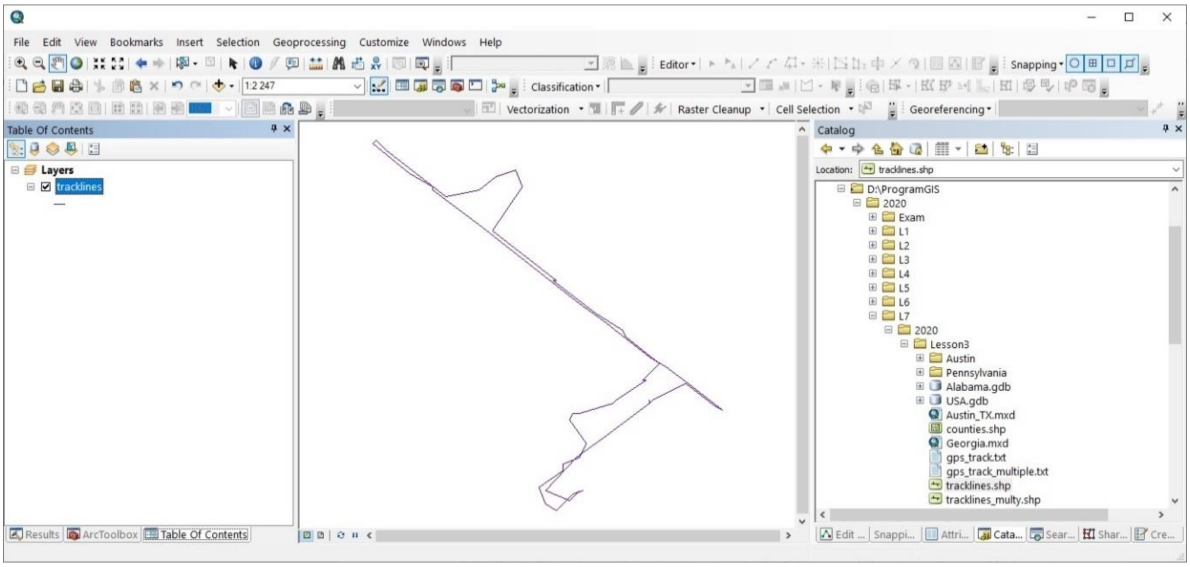


Рис.5.2 Створення Polyline за отриманими з GPS треку координатами

Питання для самоконтролю:

1. З якою метою в ArcPy використовуються токени, наприклад SHAPE@ або SHAPE@XY?
2. Які методи підтримує UpdateCursor, чи є серед них метод insertRow?
3. Що трапиться, якщо під час додавання нового рядка не вказати геометрію об’єкту?
4. Чим відрізняється робота з Point, Polyline, PointGeometry та MultiPoint при створення геометрії об’єкту?
5. Чи обов’язково при створенні точкової геометрії використовувати клас Point?
6. Яким саме методом InsertCursor додає нові рядки до атрибутивної таблиці?
7. Коли варто використовувати доступ до повних геометричних об’єктів Geometry?

6. РОБОТА З РАСТРАМИ В ARCPY

Растри являють собою унікальний тип просторових даних, і ряд інструментів геообробки розроблено спеціально для використання переваг структури растрових даних. Коротко познайомимося, як використовувати звичайні функції ArcPy для лістінгу та опису растрів.

ArcPy також містить модуль *Spatial Analyst*, який називається *arcpy.sa*, який повністю інтегрує алгебру карт у середовище Python, що робить створення сценаріїв набагато ефективнішим. Розглянемо оператори алгебри карт, а також функції та класи модуля *arcpy.sa*.

6.1. Лістинг растрів

Функція *ListRasters* повертає список Python растрів у робочій області.

Синтаксис функції *ListRasters* такий (таблиця 6.1):

ListRasters ({wild_card}, {raster_type})

Таблиця 6.1

Параметр	Пояснення	Тип даних
wild_card	Параметр wild_card дозволяє обмежувати список результатів, що повертаються. Якщо wild_card не заданий, будуть повернуті всі значення.	String
raster_type	Тип растра для обмеження результатів, що повертаються аргументом з груповим символом. Наступні типи растрів є коректними: • bmp - Формат бітової графіки для наборів растрових даних. • GIF - Формат графічного обміну для наборів растрових даних. • IMG - Формат растрових даних ERDAS IMAGINE. • JP2 - Формат растрових даних JPEG 2000. • JPG - Формат растрових даних Joint Photographics Experts Group. • PNG - Формат растрових даних Portable Network Graphics. • TIF - Tagged Image File для набору растрових даних. • GRID - Формат даних Grid. • All - Повертаються всі підтримувані типи растрів. Це значення використовується за замовчуванням. (Значення за замовчуванням - All)	String

Наступний код ілюструє використання функції *ListRasters* для друку списку растрів у робочій області:

```
import arcpy
from arcpy import env
env.workspace = r"D:\ProgramGIS\2020\L7\2020\PE\Lesson4"
rasterlist = arcpy.ListRasters( )
for raster in rasterlist:
   print (raster)
```

Результат виглядатиме приблизно так:

```
boole_ras.tif
Elevation
foxlake
foxlake_recl
landcover.tif
tm.img
```

Ім'я кожного растру друкується у вікні Python разом із необов'язковим розширенням файлу.

Наприклад, це .img для формату ERDAS IMAGINE, .tif для формат TIFF, .jpg для формату JPEG тощо. Для формату Esri GRID (база даних глобальної інформації про ресурси) або для растрів, що зберігаються в базі геоданих, розширення файлів не додаються. Тому, якщо розширення файлу відсутнє, переконайтеся, що ви працюєте з GRID або з набором растрових даних у базі геоданих.

6.2. Опис растрів

Растри можна описати за допомогою загальної функції *Describe*, як уже обговорювалося в **розділі 3**. Функція *Describe* повертає властивості для зазначеного елемента даних. Ці властивості є динамічними, що означає, що наявні властивості залежать від типу даних, що описуються. Наприклад, коли функція *Describe* використовується для растрів, загальний набір властивостей присутній на додаток до конкретних властивостей, які є унікальними для конкретного растрового елемента.

Згадаємо, що виділяють наступні три різні елементи растрових даних:

1. Растровий набір даних (*Raster dataset*) - модель просторових даних растра, яка зберігається на диску або у базі геоданих. Растрові набори

даних можуть зберігатися в багатьох форматах, включаючи TIFF, JPEG, IMAGINE, Esri GRID і MrSID. *Raster dataset* можуть бути однодіапазонними або багатодіапазонними.

2. Растровий канал (*Raster band*) - один шар в *Raster dataset*, що представляє значення даних для певного каналу в електромагнітному спектрі або інші значення отримані шляхом маніпуляції з вихідними каналами зображення. Багато типів супутникових зображень, наприклад, містять кілька каналів.

3. Растровий каталог (*Raster catalog*) - колекція *Raster datasets*, визначених у таблиці будь-якого формату, в якій записи визначають окремі набори даних растра, які включені в каталог. Растрові каталоги можуть використовуватися для відображення сусідніх або перекритих наборів растрових даних без необхідності об'єднання їх в мозаїку в один великий файл.

Властивості для кожного з цих елементів відповідно змінюються. Наприклад, формат (TIFF, JPEG, і інші) є властивістю *Raster dataset*, а розмір комірки (cell size) є властивістю *Raster band*. Загальна властивість dataType може бути використана для визначення типу елементу даних. Всі властивості, однак, доступні через *Describe*.

```
import arcpy
from arcpy import env
env.workspace = r"D:\ProgramGIS\2020\L7\2020\PE\Lesson4"
rasterlist = arcpy.ListRasters( )
for raster in rasterlist:
   desc = arcpy.Describe(raster)
   print (raster + " is <<<" + desc.dataType + ">>> raster type")
```

Подивимося на деякі властивості. Зокрема властивості *Raster dataset* (таблиця 6.2)

Таблиця 6.2

Властивість	Пояснення	Тип даних
bandCount	Кількість каналів в наборі растрових даних.	Integer
compressionType	Тип стиснення: CCITT Group 3, CCITT Group 4, CCITT (1D), DEFLATE, компресія/декомпресія, JPEG, JPEG2000, LZW, LZ77, None, PACKBITS, RLE, Векторне квантування (VECTOR QUANTIZATION), Хвильове перетворення	String

	Wavelet (MG2), Хвильове перетворення Wavelet (MG4), JPEG YCbCr	
format	Формат растру: AAIGrid, BAG, BIL, bmp, BSQ, Рівень DTED 1, Рівень DTED 2, FGDBR, FLT, GIF, GRID, GRID Stack 7.x, HDF5Image, Зображення IMAGINE, JFIF, JP2, MrSID, NITF, PNG, RPF, SDR, TIFF, TIL, USGSDEM	String
permanent	Показує постійний статус растра: False, якщо растр є тимчасовим, і True, якщо растр є постійним.	Boolean
sensorType	Тип датчика, який використовується для отримання зображення.	String

Властивості *Raster band* (таблиця 6.3)

Таблиця 6.3

Властивість	Пояснення	Тип даних
height	Число рядків	Integer
isInteger	Показує, чи є в каналі растра тип "ціле число"	Boolean
meanCellHeight	Розмір комірки в напрямку y	Double
meanCellWidth	Розмір комірки в напрямку x	Double
noDataValue	Значення Nodata каналу растра	String
pixelType	Тип пікселя • U1 - 1 bit • U2 - 2 біта • U4 - 4 біта • U8 - 8-бітові цілі числа без знака • S8 - 8-бітові цілі числа • U16 - 16-бітові цілі числа без знака • S16 - 16-бітові цілі числа • U32 - 32-бітові цілі числа без знака • S32 - 32-бітові цілі числа • F32 - З плаваючою комою одинарної точності • F64 - З плаваючою комою подвійної точності	String
primaryField	Індекс поля	Integer
tableType	Назви класів в таблиці • Значення - Значення в таблиці використовуються тільки в якості значень, а не для індексації. • Індекс - Значення в таблиці використовуються, як індекси в таблиці растра. • Некоректно - Некоректні значення.	String
width	Число стовбців	Integer

Інший приклад - для багатоканальних растрів необхідно вказати конкретний канал. Без вказівки конкретного каналу такі властивості, як розмір комірки, висота, ширина і тип пікселя будуть недоступні. Посилання на конкретні канали вказуються на зразок Band_1, Band_2 і так далі. Наступний код ілюструє, як отримати доступ до властивостей каналу в багатоканальному наборі даних:

```
import arcpy
from arcpy import env

env.workspace = r"D:\ProgramGIS\2020\L7\2020\PE\Lesson4"
rasterband = "tm.img/Layer_1"
desc = arcpy.Describe(rasterband)
x = desc.meanCellHeight
y = desc.meanCellWidth
spatialref = desc.spatialReference
units = spatialref.angularUnitName
pixeltype = desc.pixelType

print ("\n-------Rasterband block-------")

print ("The raster resolution of Band 1 is " + str(x) + " by " + str(y) + " " + units + "
in " + pixeltype + " type of pixel.")
```

6.3. Робота з растрами. Raster

ArcPy також містить клас *Raster*, який використовується для посилання на растрові дані. Растровий об'єкт може бути створений двома способами:

- шляхом посилання на існуючий растр на диску;
- за допомогою алгебри карт.

Синтаксис для класу *Raster* (таблиця 6.4):

```
Raster (inRaster)
```

Таблиця 6.4

Параметр	Пояснення	Тип даних
InRaster	Вхідний набір растрових даних.	Raster

Властивості для класу *Raster* наведено в таблиці 6.5, всі тільки для читання.

Таблиця 6.5

Властивість	Пояснення	Тип даних
bandCount	Кількість каналів в зазначеному наборі растрових даних.	Integer
catalogPath	Повний шлях і ім'я набору растрових даних.	String
compressionType	Тип стиснення. Доступні наступні типи: • LZ77 • JPEG • JPEG 2000 • PACKBITS • LZW • RLE • CCITT GROUP 3 • CCITT GROUP 4 • CCITT (1D) • Нема	String
extent	Екстент зазначеного набору растрових даних.	Extent
format	Формат растра • BIL – файл Esri Band Interleaved by Line • BIP – файл Esri Band Interleaved by Pixel • BMP – формат бітової растрової графіки • BSQ – файл Esri Band Sequential • DAT – файл ENVI DAT • GIF – графічний обмінний формат для наборів растрових даних • Grid – формат набору растрових даних Esri Grid • IMAGINE Image – формат растрових даних ERDAS IMAGINE • JP2000 – формат растрових даних JPEG 2000 • JPEG – формат растрових даних Joint Photographic Experts Group • PNG – формат растрових даних Portable Network Graphic	String

	• TIFF – формат файлу зображення для набору растрових даних	
hasRAT	Позначає, чи є пов'язана атрибутивна таблиця: True - атрибутивна таблиця присутня, False - відсутня.	Boolean
height	Число рядків.	Integer
isInteger	Цілочисльний стан: True - набір растрових даних є цілочисельним.	Boolean
isTemporary	Стан зазначеного набору растрових даних: True - набір растрових даних є тимчасовим, False - постійним.	Boolean
maximum	Максимальне значення в зазначеному наборі растрових даних.	Double
mean	Середнє значення в зазначеному наборі растрових даних.	Double
meanCellHeight	Розмір комірки в напрямку y.	Double
meanCellWidth	Розмір комірки в напрямку x.	Double
minimum	Мінімальне значення в зазначеному наборі растрових даних.	Double
name	Ім'я зазначеного набору растрових даних.	String
noDataValue)	Значення NoData в зазначеному наборі растрових даних.	Double
path	Повний шлях і ім'я набору растрових даних.	String
pixelType	Тип пікселя • U1 - 1 bit • U2 - 2 біта • U4 - 4 біта • U8 - 8-бітові цілі числа без знака • S8 - 8-бітові цілі числа • U16 - 16-бітові цілі числа без знака • S16 - 16-бітові цілі числа • U32 - 32-бітові цілі числа без знака • S32 - 32-бітові цілі числа • F32 - З плаваючою комою одинарної точності • F64 - З плаваючою комою подвійної точності	String
spatialReference	Просторова прив'язка зазначеного набору растрових даних.	SpatialReference
standardDeviation	Стандартне відхилення значень осередків в набір растрових даних.	Double
uncompressedSize	Розмір зазначеного набору растрових даних на диску.	Double
width	Число стовбців.	Integer

Для класу *Raster* важливим є метод save ({name}), який дозволяє зберігати об'єкт *Raster* в зображення на диску (таблиці 6.6).

Таблиця 6.6

Параметр	Пояснення	Тип даних
name	Ім'я, яке призначається набору растрових даних на диску.	String

Як результат створення об'єкту *Raster*, наприклад з зображення на диску, отримаємо можливість застосувати до такого об'єкту Алгебру карт.

6.4. Робота з растрами. Алгебра карт

Алгебра карт (*Map Algebra*) - це проста і потужна алгебра, за допомогою якої можна виконувати всі інструменти *Spatial Analyst*, оператори і функції для виконання географічного аналізу.

Алгебра карт доступна в модулі *Spatial Analyst*. Оскільки *Map Algebra* інтегрована в Python, Ви можете використовувати всю функціональність Python і ArcPy і їх розширень (модулів, класів, функцій і властивостей).

Для роботи з *Spatial Analyst* перевірте наявність ліцензії:

```
# Check out the Spatial Analyst extension

arcpy.CheckOutExtension("Spatial")
```

Наступний код показує, як створити растровий об'єкт за допомогою посилання на растра з диску:

```
import arcpy
myraster = arcpy.Raster(r"D:\ProgramGIS\2020\L7\2020\PE\Lesson4\elevation")
```

або допомогою *Map Algebra*

```
import arcpy
outraster = arcpy.sa.Slope(r"D:\ProgramGIS\2020\L7\2020\PE\Lesson4\elevation
")
```

В обох випадках отриманий об'єкт *Raster* може бути використаний в операторах Python і в додаткових виразах *Map Algebra*. Об'єкти *Raster* мають багато властивостей, які багато в чому аналогічні тим, які вже обговорювалися раніше в *Describe*, в тому числі bandCount, compressionType, format, height, width, meanCellHeight, meanCellWidth, pixelType, spatialReference, і інші.

Для написання виразів в *Map Algebra* необхідно зрозуміти, що є можливість роботи з об’єктом *Raster* і через *Spatial Analyst*, тоді треба дивитися документацію за відповідною функцією в модулі, або через використання *операторів* які підтримує Алгебра карт.

Як ми побачили з документації об'єкти *Raster* мають тільки один метод: *save()*. Об'єкт *Raster* (змінні і пов'язаний набір даних) повертається з виразу алгебри карт як тимчасовий, за замовчуванням. Це означає, що змінна і контрольний набір даних видаляються, коли змінна виходить з області видимості - наприклад, коли ArcGIS або автономний скрипт закриваються. Метод *save()* може бути викликаний аби зробити растровий об'єкт постійним.

```
import arcpy
from arcpy import env
env.workspace = r"D:\ProgramGIS\2020\L7\2020\PE\Lesson4\USA.gdb"
arcpy.CheckOutExtension("Spatial")
outraster = arcpy.sa.Slope("elevation")
outraster.save("Slope")
print ("All done!")
```

В цьому прикладі ми використали для роботи з об'єктом *Raster* модуль *Spatial Analyst*.

Розглянемо наступний приклад, який приймає мінімальне і максимальне значення висот в якості параметрів, потім робить деяку алгебру карт з цими значеннями. Ми використаємо *оператори* які підтримує *Алгебра карт*. Написаний нами вираз ізолює області, де висота більше, ніж мінімальний параметр і менше максимального параметра. Комірки, які задовольняють виразу дають значення 1, а ті комірки, що ні - 0.

```
# This script takes a DEM, a minimum elevation, and a maximum elevation. It
outputs a new raster showing only areas that fall between the min and the max
import arcpy
from arcpy import env
from arcpy.sa import *
arcpy.CheckOutExtension("Spatial")
```

```
arcpy.env.overwriteOutput = True
env.workspace = r"D:\ProgramGIS\2020\L7\2020\PE\Lesson4"

# Get parameters of min and max elevations
inMin = arcpy.GetParameterAsText(0)
inMax = arcpy.GetParameterAsText(1)

# Perform the map algebra and make a temporary raster
inDem = Raster("foxlake")
tempRaster = (inDem > int(inMin)) & (inDem < int(inMax))

# Set up remap table and call Reclassify, leaving all values not 1 as NODATA
remap = RemapValue([[1,1]])
remappedRaster = Reclassify(tempRaster, "Value", remap, "NODATA")

# Save the reclassified raster to disk
remappedRaster.save("foxlake_recl")
arcpy.CheckInExtension("Spatial")
print ("All done!")
```

В даному сценарії вираз для роботи з *Raster* виглядає як (inDem > int(inMin)) & (inDem < int(inMax)).

Рекомендуємо дослідити які ще вирази можна використати в офіційній документації по *операторах* які підтримує *Алгебра карт*

6.5. Робота з растрами. Отримати значення комірки

Часто при роботі з растрами виникає задача отримати значення, яке зберігається в конкретному пікселі, зокрема коли працюємо з цифровими моделями рельєфу. Для цього є метод, який є достатньо простим для використання – *GetCellValue*.

GetCellValue отримує значення пікселя по його координатам x, y. Цей інструмент використовується, якщо вам потрібно значення пікселя для моделі геообробки. У ArcMap, ArcScene або ArcGlobe Ви можете використовувати інструмент Ідентифікувати (*Identify*) замість нього.

Синтаксис для функції *GetCellValue* (таблиця 6.7):

```
arcpy.management.GetCellValue(in_raster, location_point, {band_index})
```

Таблиця 6.7

Параметр	Пояснення	Тип даних
in_raster	Растр, до якого Ви хочете зробити запит.	Mosaic Dataset; Mosaic Layer; Raster Layer
location_point	Координати x і y розташування пікселя.	Point
band_index [band_index,...] (Додатковий)	Задайте канали, до яких Ви хочете створити запит. Залиште порожнім для запиту до всіх каналів мультиспектрального набору.	Value Table

Розглянемо приклад на використання *GetCellValue* – дізнаємося з DEM за координатами точкових об’єктів Feature Class значення висот, які зберігаються в растрі.

```
# This script extract value from a DEM by Feature Class.
import arcpy
from arcpy import env
env.workspace = r"D:\ProgramGIS\2020\L7\2020\PE\Lesson4"
inDem = "foxlake"
featureClass = "Temp.shp"
# Enter for loop for each feature
with arcpy.da.SearchCursor(featureClass, ["SHAPE@XY",]) as cursor:
    for row in cursor:
        x, y = row[0]
        result = arcpy.GetCellValue_management(inDem,str(x) + " " + str(y))
        print ("Pixel value: " + result.getOutput(0))
print ("All done!")
```

6.6. Робота з растрами. Numpy

Слід згадати ще дві растрові функції:

- *RasterToNumPyArray* - конвертує растр в масив *NumPy*.
- *NumPyArrayToRaster* - конвертує масив *NumPy* в растр.

Це звичайні функції ArcPy, а не функції модуля *arcpy.sa*. Ці дві функції дозволяють конвертіцію між растрами та масивами *NumPy*. Масив NumPy призначений для роботи з дуже великими масивами. Сам *NumPy* - це пакет, який використовується для наукових обчислень за допомогою Python. Крім усього іншого, він забезпечує роботу з потужними об'єктами N-вимірного масиву. Цей тип об'єктів дає можливість переміщувати дані між базами даних. Наприклад, пакет *SciPy* містить численні алгоритми, які можуть бути корисними для конкретного застосування, такі як перетворення Фур'є, моделі максимальної ентропії та обробка багатовимірних зображень.

Замість того, щоб намагатися створити інструмент в ArcGIS, який виконує ці спеціалізовані функції, ви можете написати інструмент сценарію, який перетворює растр на масив *NumPy*, а потім викликає спеціалізовані функції з пакету *SciPy*.

Синтаксис для функції *RasterToNumPyArray* (таблиця 6.8):

RasterToNumPyArray (in_raster, {lower_left_corner}, {ncols}, {nrows}, {nodata_to_value})

Таблиця 6.8

Параметр	Пояснення	Тип даних
in_raster	Вхідний растр, що конвертується в масив NumPy.	Raster
lower_left_corner	Нижній лівий кут в in_raster, з якого витягується блок обробки для конвертації в масив. Значення x і y в одиницях карти. Якщо значення не задано, то буде використана початкова точка вхідного растру. (Значення за замовчуванням - None)	Point
ncols	Число стовпців з lower_left_corner в in_raster для конвертації в масив NumPy. Якщо значення не задано, то буде використано кількість стовпців вхідного растру. (Значення за замовчуванням - None)	Integer
nrows	Число рядків з lower_left_corner в in_raster для конвертації в масив NumPy. Якщо значення не задано, то буде використано кількість рядків вхідного растру. (Значення за замовчуванням - None)	Integer
nodata_to_value	Значення для присвоєння значень NoData з in_raster в отриманий масив NumPy. Якщо значення не задано, то буде використано значення NoData з in_raster. (Значення за замовчуванням - None)	Variant

Синтаксис для функції [NumPyArrayToRaster]() (таблиця 6.9):

NumPyArrayToRaster (in_array, {lower_left_corner}, {x_cell_size}, {y_cell_size}, {value_to_nodata})

Таблиця 6.9

Параметр	Пояснення	Тип даних
in_array	Масив NumPy, конвертований в растр. Потрібно дво- або тривимірний масив NumPy.	NumPyArray
lower_left_corner	Об'єкт Point (Точка), що визначає нижній лівий кут вихідного растра в одиницях карти. За замовчуванням, нижній лівий кут встановлюється з координатами (0.0, 0.0). (Значення за замовчуванням - None)	Point
x_cell_size	Розмір комірки в напрямку x, заданий в одиницях карти. Вхідними даними можуть бути задане значення розміру комірки (тип: Double) або вхідний растр. Коли набір даних заданий, розмір комірки цього набору даних по осі x використовується в якості розміру комірки по осі x вихідного растру. Якщо задано тільки значення x_cell_size, без y_cell_size, то буде створена квадратна комірка відповідного розміру. Якщо ні значення x_cell_size, ні значення y_cell_size не задані, то буде використано значення за замовчуванням, рівне 1. (Значення за замовчуванням - 1.0)	Double
y_cell_size	Розмір комірки в напрямку y, заданий в одиницях карти. Вхідними даними можуть бути задане значення розміру комірки (тип: Double) або вхідний растр. Коли набір даних заданий, розмір комірки цього набору даних по осі x використовується в якості розміру комірки по осі y вихідного растру. Якщо задано тільки значення y_cell_size, без x_cell_size, то буде створена квадратна комірка відповідного розміру. Якщо ні значення x_cell_size, ні значення y_cell_size не задані, то для розмірів x і y осередку буде використано значення за замовчуванням, рівне 1,0. (Значення за замовчуванням - 1.0)	Double
value_to_nodata	Значення в масиві NumPy, якому в вихідному растрі буде присвоєно значення NoData. Якщо для value_to_nodata значення не вказано, в отриманому растрі значень NoData не буде. (Значення за замовчуванням - None)	Double

Загальний сценарій виглядатиме так:

```
import arcpy
import numpy
from arcpy import env
from arcpy.sa import *
arcpy.CheckOutExtension("Spatial")
arcpy.env.overwriteOutput = True
env.workspace = r"D:\ProgramGIS\2020\L7\2020\PE\Lesson4"

# Get input Raster properties
inRas = arcpy.Raster('foxlake')
lowerLeft = arcpy.Point(inRas.extent.XMin,inRas.extent.YMin)
cellSize = inRas.meanCellWidth

# Convert Raster to numpy array
arr = arcpy.RasterToNumPyArray(inRas,nodata_to_value=0)

# Calculate percentage of the row for each cell value
arrSum = arr.sum(1)
arrSum.shape = (arr.shape[0],1)
arrPerc = (arr)/arrSum

#Convert Array to raster (keep the origin and cellsize the same as the input)
newRaster = arcpy.NumPyArrayToRaster(arrPerc, lowerLeft,cellSize,
value_to_nodata=0)
newRaster.save("foxlake_per")
arcpy.CheckInExtension("Spatial")
print ("All done!")
```

Це гарна ідея використати *NumPy* для роботи з растрами, адже при роботі із великими растрами і необхідності використання додаткових функцій, яки нема в ArcGis, Ви значно покращите ефективність роботи сценаріїв.

Питання для самоконтролю:

1. Які параметри приймає функція функції ListRasters, чи можна використати фільтри за іменами растрів при її використанні?
2. Які властивості модна отримати для растра з розширенням .tiff?
3. Чим відрізняється Raster від растру, що зберігається на диску?

4. До яких об’єктів можна застосувати Алгебру карт?
5. Який результат повернеться після використання Алгебри карт, що це буде за тип даних і які значення будуть у ньому зберігатися?
6. З якими типами растрів працює функція GetCellValue?
7. Чи є можливість задати просторову прив’язку для растрів, які є результатом виконання функції NumPyArrayToRaster?

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ArcPy Essentials (2023). *Learning Plan from Esri Academy*. Режим доступу: https://www.esri.com/training/catalog/5e7a48e6a662e60f85592a97/arcpy-essentials/

Lutz, M. (2013). Programming Python, 5th Edition. *O'Reilly Media, Inc.*, - 1643 p.

Python for Everyone (2023). *Web Course from Esri Academy*. Режим доступу: https://www.esri.com/training/catalog/57630436851d31e02a43f13c/python-for-everyone/

Sterling, Q., Dutton, J.A., Detwiler, J., Hardisty, F., O'Brien, J. (2023). ГІС Programming and Software Development. Режим доступу: https://www.e-education.psu.edu/geog485/node/91

Zandbergen, Paul A. (2013). Python scripting for ArcGIS. *Esri Press. N.Y.*, - 360 p.

Документація по ArcMap (2023). Режим доступу: https://desktop.arcgis.com/en/arcmap/latest/analyze/main/what-is-geoprocessing.htm

Документація по ArcPy (2023). Режим доступу: https://desktop.arcgis.com/en/arcmap/latest/analyze/arcpy/what-is-arcpy-.htm

Документація по Python 2.7.18 (2020). Режим доступу: https://docs.python.org/2.7/