

# An Introduction to Python

1

This video will introduce you to the Python scripting language and cover a few basic concepts.

# Why Python?

- Relatively easy to learn compared to other languages...
  - code is easy to read
  - many built-in tools
- Many tutorials and books on Python
- Supported by ESRI for use with ArcGIS...
  - all ArcGIS documentation uses Python in examples
  - ESRI scripting workshops teach Python

Python is currently the scripting language most widely used by GIS users. The language is relatively easy to learn and contains many built-in tools. In addition, there is a large and active community of Python users that freely share knowledge and tools that they've created. Another good reason for GIS users to choose Python is that it is the language that ESRI supports for use with ArcGIS.

# The Python language

- It's just English with different syntax and a few new terms...

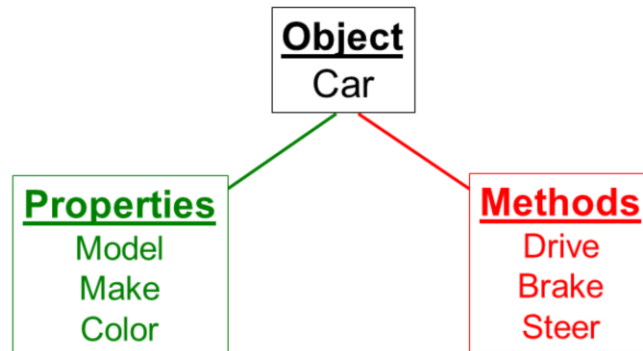
## English

## Python

- Sentence → ▪ **Statement** – one complete instruction
- Noun → ▪ **Object** – any piece of data
- Adjective → ▪ **Property** – describes an object
- Verb → ▪ **Method** – action an object can take

Python is a very concise version of English – it conveys our instructions to the computer without any unnecessary words. We'll spend the next few weeks learning the **syntax** of Python – i.e. how to form “sentences”. But for now, we'll start with a few basic terms and concepts that you'll need to know in order to understand Python textbooks and documentation. In Python, the term **statement** is used to indicate a single complete instruction – in English, we would call this a sentence. An **object** is the Python equivalent of a noun – it is any piece of data (e.g. a number, a word, a file, etc.). Objects have **properties** that describe them and actions, or **methods**, that they can perform.

## Objects, methods, and properties – a real world analogy



4

Let's use a simple real-world analogy to illustrate the concepts of object, properties, and methods. In this analogy, our object is a car. The car can be described by certain properties (e.g. model, make, color) and it can perform certain methods (e.g. drive, brake, steer). The properties and methods of an object depend on the type of object - e.g. a car has different properties and capabilities than a computer.

## Accessing an object's properties

- To get an object's properties...
  - specify the object,
  - followed by a **period**,
  - followed by the **name of the property**



5

In order to get the properties of an object, we create an expression (statement part) using the following steps: 1) specify the object name, 2) followed by a period, 3) followed by the name of the property. When we run this expression, Python will tell us the value of the property.

## Using an object's methods

- To use an object's methods...
  - Same syntax as for accessing properties, but...
  - followed parenthesis including parameter values:
    - not all methods require parameters to be specified.
    - parenthesis are still necessary even if no parameters need to be specified for the method.

Car.Steer(left)                      Car.Brake()  
    ↖    ↑    ↙                      ↖  
object period method parameter(s)

6

If we want an object to perform a method, we start off with the same syntax as for getting an object's property. The difference comes at the end of the expression – for a method, we need to include parenthesis after the method name; inside the parenthesis, we would include any necessary **parameters**. These parameters are the details that Python needs in order to understand what exactly you want the method to do. Not all methods require you to specify a parameter but you still need to include empty parenthesis – this is Python's clue that it should perform a method rather than retrieve a property.

# Variables

- Names given to objects.
  - Refer to objects by their assigned name.
- When statement is run, object is used in place of the variable.
- To assign a variable:
  - Specify the **variable name**,
  - followed by the equal sign ( = ),
  - followed by the definition.

`variable` → field\_name = "Town" ← `object`

7

To make our scripts more efficient, we can give names to our objects - object names are called **variables**. When we “assign a variable”, we’re simply letting Python know what we want to call an object. To assign a variable to an object, simply specify the name you want to give the object followed by an equal sign. The equal sign is followed by the object itself. After you’ve assigned a variable to an object in your script, you can use the variable to refer to that object.

## Rules for naming objects

- Must start with a letter or underscore.
- Can contain letters, digits, or underscores.
- No spaces.
- No reserved words:
  - i.e. `and`, `or`, `del`, `for`, `if`, `print`, `try`, `except`
  - reserved words change color
- No quotes around variable names
  - quotes are for strings (text) only

8

You can use any variable to name your object but you have to follow a few rules. Variables must only contain letters, numbers, or underscores and the first character cannot be a number. There cannot be spaces in the variable and you cannot use reserved words (which have a different font color when typed in Python). Variables often look like strings (i.e. text) because they contain letters and so it may be tempting to put quotes around variables as you would with strings. But remember, there must be NO QUOTES around variables – otherwise Python will interpret the variable as a string instead of the object that the variable was intended to represent.



## Working with variables

- Use object names in a statement instead of the object itself:

```
>>> x = 5
>>> y = 7
>>> x+y
12
```

Assign variables

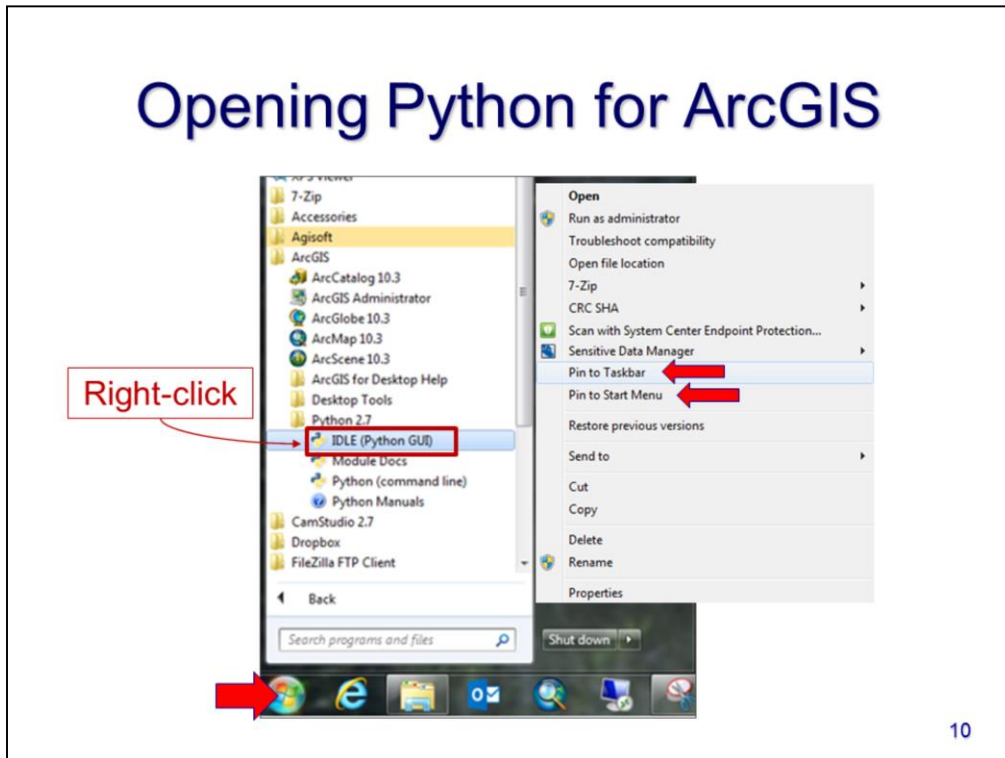
Result

- Case matters.** Be careful about spelling and case when using a variable...
  - copy and paste to ensure correct spelling
- Variables do not exist until they are assigned to an object.

9

Variables are the names that we assign to objects. Whenever Python “sees” a variable, after it is defined, it will substitute the object represented by the variable into the expression. When typing variables in your script, it is important to spell the variable exactly the same every time – including the case. For longer variable names, I suggest copying and pasting to ensure there are no typos. Also keep in mind that you cannot refer to a variable until after you’ve defined it in your script – otherwise Python won’t know what the variable means.

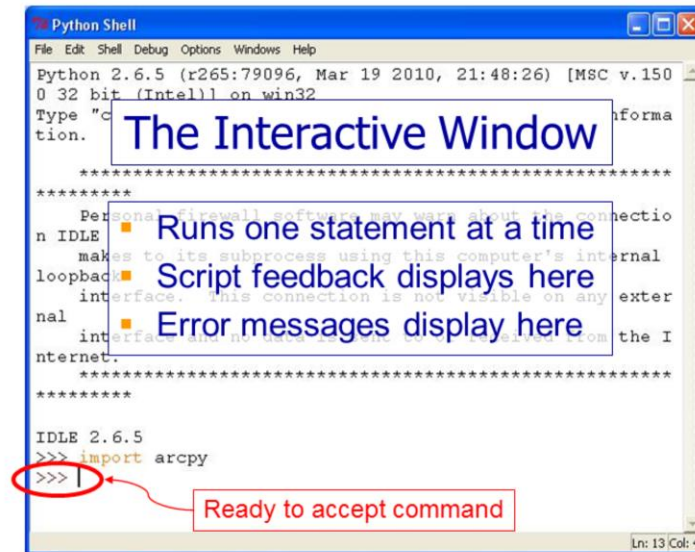
# Opening Python for ArcGIS



10

For this course, it is important to use the version of python that is installed with ArcGIS (i.e. Python v2.7.8) – this will ensure that we don't have to worry about any setup or compatibility issues when using ArcGIS tools in a script. To open the correct version of Python, go to Start > Programs > ArcGIS > Python 2.7 > IDLE (Python GUI). For more convenient access in the future, you can right-click on IDLE, and select either “Pin to Taskbar” or “Pin to Start Menu”.

# The Interface - Python IDLE

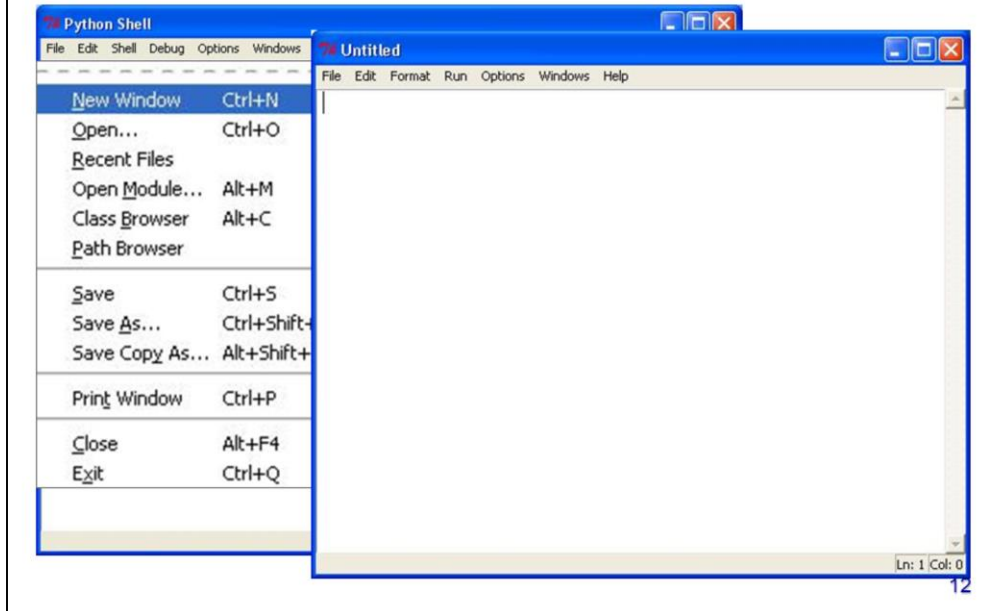


11

In this course, we will use IDLE (Integrated Development and Learning Environment) as our Python interface. IDLE is a basic and reliable interface which comes with the Python installed by ArcGIS.

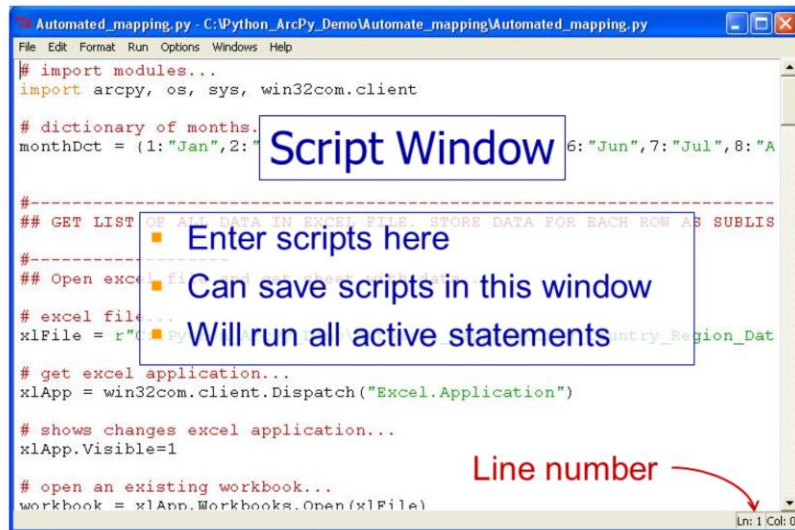
The IDLE interface has two windows that you'll use when developing scripts. The first window is the "Python Shell" which is an interactive window that allows you to run single statements and receive script feedback and error information. This window allows you to test specific statements when developing your script.

# Opening a script window



When you start up IDLE, the first window that will appear is the Python Shell. Through the Shell, you can open up a new script window.

# The Interface - Python IDLE



The screenshot shows a Python IDLE window titled "Automated\_mapping.py". The window contains a Python script with the following code:

```
File Edit Format Run Options Windows Help
# import modules...
import arcpy, os, sys, win32com.client

# dictionary of months.
monthDct = {1: "Jan", 2: "Feb", 3: "Mar", 4: "Apr", 5: "May", 6: "Jun", 7: "Jul", 8: "Aug", 9: "Sep", 10: "Oct", 11: "Nov", 12: "Dec"}

#-----
## GET LIST OF ALL DATA IN EXCEL FILE. STORE DATA FOR EACH ROW AS SUBLISTS
#-----
## Open excel file and get data about sublists
# excel file...
xlFile = r"C:\Python_ArcPy_Demo\Automate_mapping\Automated_mapping\entry_Region_Data.xls"

# get excel application...
xlApp = win32com.client.Dispatch("Excel.Application")

# shows changes excel application...
xlApp.Visible=1

# open an existing workbook...
workbook = xlApp.Workbooks.Open(xlFile)
```

Annotations in the image include:

- A box labeled "Script Window" pointing to the code area.
- A box labeled "Enter scripts here" pointing to the code area.
- A box labeled "Can save scripts in this window" pointing to the code area.
- A box labeled "Will run all active statements" pointing to the code area.
- A red arrow labeled "Line number" pointing to the status bar at the bottom right, which displays "Ln: 1 Col: 0".

13

The script window in IDLE is where you will type your actual script. IDLE color codes certain information in a script to help you more easily recognize certain data types and functions. The colors will depend on the configuration of IDLE on your computer.