Last Updated: July 2017

EXERCISE 2

Concepts of Scripting, Part 1

Introduction

There are a number of powerful tools available for geospatial analysis that require some knowledge of scripting, such as ArcPy, Google Earth Engine, and the R computing environment. These tools extend the power and functionality of GIS and remote sensing. This exercise provides an introduction to scripting concepts that aren't specific to any programming language or geospatial tool but are necessary to understand to begin writing code in any environment. The exercise is designed to let you work in Python, R, or JavaScript. The numbered instructions provide information about what the commands you enter will do, and the tables give examples about what code needs to be typed in your chosen language.

Objectives

- Become familiar with basic concepts of scripting

Required Software (Choose 1)

- **RStudio**
- **Python v2.x**
- **Google Earth Engine Account**

Prerequisites

- *Introduction to Geospatial Scripting, Exercise 1: Planning a Script*

Table of Contents

Part 1: Variables and Statements

Understanding variables and statements, and knowing how to correctly write them in a programming language, is crucial to learning to use geospatial scripting tools. In this section you may choose between 3 scripting environments, RStudio, Google Earth Engine (GEE) Code Editor, or Python (IDLE). If you need to install or access one of these environments, please refer to the Appendix.

Important: *Notes on Syntax. Before you begin you must be aware of what syntax is in a scripting language. Syntax is the format that programs must be written in. In a natural language you may think of this as sentence structure or grammar. In a scripting language, it may include things like case sensitivity, dot notation, rules for naming variables, and punctuation usage. Just as with natural languages, syntax differs across scripting languages and you will notice these differences throughout all of the exercises in this course. This exercise has many tables to demonstrate different syntax in 3 languages, so make sure when you enter commands in your script that you're using the correct syntax for your chosen language.*

A. Variables

A variable is a reference to a place in the computer's memory, which can contain a value. Different languages have different syntax for naming variables. For example, in JavaScript and R, variables must begin with a letter. In Python they can also begin with an underscore, but none of these languages will let the name of your variable begin with a number. The name, variable1, is acceptable in all three languages, but the name, 1variable, will give an error.

Python, JavaScript, and R are all case sensitive languages. This means that upper and lowercase matter when it comes to naming variables. The variables, "variable1", and, "Variable1", are not considered the same.

1. **Open** IDLE, RStudio, or a Code Editor window. **Save** the script the same way you learned in Exercise 1, part A. Name the saved file Intro_to_GeospatialScripting_ex2. In Python and R, make sure to use the .py or .R extension.
2. To begin, we're going to create the variable, x. In your scripting window create x, then **assign** the variable x a value of 5. For syntax, see the table below.

Language	Environment	Assign Variable Value
Python	IDLE	x=5
R	RStudio	x=5 or x <- 5
JavaScript	Google Earth Engine	var x = ee.Number(5);

Note: *Why does JavaScript use ee.Number? It's because this exercise focuses on using JavaScript in Earth Engine. If you were scripting in JavaScript outside of Earth Engine, you wouldn't use ee.Number. Instead you would just type in your variable's value and it would assume the number type. The "ee" stands for Earth Engine and it means that this number is an Earth Engine object, which you will learn about in the* *next exercise. If you would like to read more about these objects now, see https://developers.google.com/earth-engine/client_server.*

Have you noticed differences in syntax yet? One example are Semicolons (;). These are used in JavaScript to mark the end of statements. In Python and R statements are more commonly ended by a line break (hitting Enter).

B. Statements

A statement is an instruction given to the machine. They are the smallest piece of a code that can perform work on their own. In a natural language, like English, you must speak using a complete sentence. In a scripting language, you need to write using complete statements. The line that you just wrote, creating the variable x and assigning it a value of 5, is now a complete statement.

3. **Run** the Script.
   i. To run the script in Python, **Click** Run, then Run Module, in the Python window. It will prompt you to save the script first.
   ii. To run the script in R, **highlight** the entire script, then **click** Run. You will see the code run in the console at the bottom left.
   iii. To run the script in the Earth Engine Code Editor, **click** Run.

   You may not be able to tell that your computer has done anything, but the instruction to create a variable and assign it a value has been issued.
4. To confirm the variable now exists, enter a print statement on the next line. A print statement is an instruction sent to the computer that tells it to output information. **Type** a print statement using the syntax in the table below, depending on the language you're using.

Language	Environment	Enter	Outputs
Python	IDLE	print x	5
R	RStudio	x	[1] 5
JavaScript	Google Earth Engine	print(x)	5

Your full script should now read:

Language	Environment	Full Script
Python	IDLE	x=5 print x
R	RStudio	x <- 5 x

JavaScript	Google Earth Engine	var x = ee.Number(5);
		print(x)

Note: *In R, you can type print (x), but there is no need. Simply typing x will cause the variable to be printed.*

5. **Run the script.** Notice the output (5) that is listed in the table above.
6. The two lines that you have typed so far are both examples of statements. Keep the variable x for the next section.

C. Variable Types (Strings, Numbers, and Lists)

Variables don't have to be numbers. Variables can take several other forms including strings and lists, which you are going to learn about in this section. Each language has its own variable types that differ slightly. In this exercise you will be learning about some common data types that exist in Python, R, and JavaScript. As you begin to learn more of the specifics of a single language, you will start become more familiar with its variable types. In this exercise you'll learn about some common variable types that are used in all three languages.

1. String variables (often referred to as "Characters" in R) can hold text. To create a variable that stores a string, enclose the variable's value in quotes (""). These variables can hold anything inside the quotation marks including numbers, letters, and other characters. In your scripting window, **create** a new variable, y, and set it equal to "five".

Language	Environment	Enter
Python	IDLE	y="five"
R	RStudio	y<-"five"
JavaScript	Google Earth Engine	var y=ee.String("five");

2. The variable y is now equal to a string of characters, not the number 5. There is another useful command we can give the computer to have it tell us what type of variable we have. It's called a type command. Put a *type of* command in a print statement. **Type** a print statement to get the type of y, then **run the script.** Syntax will vary across languages, see table below.

Language	Environment	Enter	Outputs
Python	IDLE	print(type(y))	<type 'str'>
R	RStudio	typeof(y)	[1] "character"
JavaScript	Google Earth Engine	print(typeof(y.getInfo()));	string

Note: *In Earth Engine, be wary of using the .getInfo command. It is used here to illustrate a data type. But if used inappropriately, it can cause errors, especially on large collections of data. You can read more about this here: https://developers.google.com/earth-engine/client_server*

3. Notice how every environment gives a different output, but each output is telling us the same thing. The variable y is a string (referred to as a character in RStudio).
4. Try giving the same command for the variable x. **Type** a print statement to get the type of x, then **run the** script. Syntax will vary across languages, see table below.

Language	Environment	Enter	Outputs
Python	IDLE	print(type(x))	<type 'int'>
R	RStudio	(typeof(x))	[1] "double"
JavaScript	Google Earth Engine	print(typeof(x.getInfo()));	number

This may be an unexpected result. JavaScript provided number, because we used the getInfo() command to pull the object type out of the Earth Engine object. But what is an int? A double? These are all examples of numeric data types. JavaScript provided number because all numbers in JavaScript are 32-bit floating point numbers, so there is no reason to disambiguate what type of number. A floating point number means that the decimal point in a number can "float", and the bit depth, 32, means that this is a higher precision number. Python is 'int' which is short for integer, which are whole numbers. R provided "double", which is also a floating point number. R also supports integers, but its default storage type is double. For these exercises we won't elaborate further on number types in these languages. If you would like to learn more about number types please see the additional resources section or the glossary.

5. Let's move on to lists. Lists are ordered collections of values, which are useful for many geospatial processing applications that we will explore later on. Let's begin by creating a list of numbers. **Create** the variable num_list and **set it** equal to the list 9,4,7,1. See the table below for syntax.

Language	Environment	Enter
Python	IDLE	num_list = [9,4,7,1]
R	RStudio	num_list = c(9,4,7,1)
JavaScript	Google Earth Engine	var num_list = ee.List([9,4,7,1]);

6. **Add** another print statement for num_list and **Run the script**. You'll see it output a list of numbers.
7. Lists can hold more than just numeric values. They can hold many kinds of values. **Create** a list called string_list using a list of strings. See the table below for syntax.

Language	Environment	Enter
Python	IDLE	string_list = ["scripting", "is", "fun"]
R	RStudio	string_list = c('scripting', 'is', 'fun')
JavaScript	Google Earth Engine	var string_list = ee.List(['scripting', 'is ', 'fun']);

8. **Add** another print statement for string_list and **Run the script**. You'll see it output the list you just created.
9. Lists can also hold mixed value types. **Create** one more list called mix_list and fill it with the values 5 and “five”. See the table below for Syntax

Language	Environment	Enter
Python	IDLE	mix_list = [5, "five"]
R	RStudio	mix_list = c(5, 'five')
JavaScript	Google Earth Engine	var mix_list = ee.List([5, 'five']);

10. **Add** another print statement for mix_list and **Run the script**. You'll see it output the list you just created.

There are many other types of data: dictionaries, matrices, and Booleans to name a few. Not every scripting language uses the same data types, and today we don’t have time to cover all of them, or the differences in each language. As you go through the later exercises you may be introduced to additional data types.

Part 2: Operators

Operators are commands that we can give to the computer in a scripting language. In this exercise you will learn about arithmetic operators, relational operators, and logical operators.

A. Arithmetic Operators (+,-,*,/)

Arithmetic operators are tools used to perform some mathematical process on a variable. They can be very useful in geospatial scripting.

1. To see how operators work, begin by **opening** a new window, or if you still have your IDLE, RStudio, or Code Editor window open, simply delete the code you've written so far.
2. Begin by setting up some variables. First **create** variable x, and **set it** equal to 5. Then **create** variable y and **set it** equal to 7. Use ee.Number(5); in Earth Engine.

Language	Environment	Enter

Python	IDLE	x=5
		y=7

R	RStudio	x<-5
		y<-7

JavaScript	Google Earth Engine	var x = ee.Number(5);
		var y = ee.Number(7);

3. Now we're going to make a new variable and set it equal to the sum of x and y. **Create** the variable z and **set it** equal to x+y.

Language	Environment	Enter
Python	IDLE	z = x+y
R	RStudio	z <- x+y
JavaScript	Google Earth Engine	var z = x.add(y);

4. Now **add** a print statement to print out the variable z, just like you did in part 1. **Run the script.** You will see that z is now set to the sum of x and y.

Other arithmetic operators (-, *, and /) can be used in the exact same way. These can be used to apply equations to your data.

B. Relational Operators (<, >, ==, !=)

Relational operators (also referred to as comparison operators) are used the same way as arithmetic operators, but rather than perform some mathematical function they evaluate the relationship between two variables. This can make it easy to evaluate whether the variables you've calculated are less than or greater than one another. This can be useful when determining two variables relationship to one another, for instance if one is greater than the other.

1. Keep the variables from part A that you already have created.
2. To see how simple relational operators are to use, start with a simple print statement. Tell the computer to print x>y using appropriate syntax provided in the table below. **Run the script.**

Language	Environment	Enter	Outputs
Python	IDLE	print(x>y)	False
R	RStudio	x>y	[1] FALSE
JavaScript	Google Earth Engine	print(x.gt(y))	0

3. The computer will print out false (or 0 in Earth Engine as opposed to 1, which would be true) because this is a false relationship; x is not greater than y. Try one more. Give another print statement. This time, test is x is not equal (!=) to z. **Run the Script**

Language	Environment	Enter	Outputs
Python	IDLE	print(x!=z)	True
R	RStudio	x!=z	[1] TRUE
JavaScript	Google Earth Engine	print(x.neq(z))	1

4. This time the computer prints true (or 1), because it is true that x is not equal to z. Z is still the sum of x and y. Just as there are differences in syntax between the three languages, there are differences in operators as well. To demonstrate this, let’s change the value of our variables. **Set** x equal to the number 2 and **set** y equal to the string “2”.

Language	Environment	Enter

Python	IDLE	x=2
		y="2"

R	RStudio	x<-2
		y<-"2"

JavaScript	Google Earth Engine	var x = ee.Number(2);
		var y = ee.String("2");

5. Now **type** a print statement for x=y. **Run the script.**

Language	Environment	Enter	Outputs
Python	IDLE	print(x==y)	False
R	RStudio	x==y	[1] TRUE
JavaScript	Google Earth Engine	print(x==y);	false

6. The three languages seem to disagree here. This is simply a difference in characteristics of the operators in different languages. Python and JavaScript gives false because the variables are not the same *type*. R returns true, because it recognizes the values are the same, though they are not the same type.

C. Logical Operators (And, Or, and Not)

Logical operators exist to help us make logical decisions in a script. You will find that when writing conditional statements (which you will learn about in the next exercise) that logical operators are exceedingly useful when passing Boolean values and comparing variables.

1. To begin lets set the string variable y back to a number. **Set** y equal to 7. Leave x as 2, and z as the sum of x and y
2. Just as before, **add** a print statement. This time use the AND operator to test if x is less than y AND z. Remember that AND is different from +, which is an arithmetic operator.

Language	Environment	Enter	Outputs
Python	IDLE	print(x<(y and z))	True
R	RStudio	x<y&z	[1] TRUE
JavaScript	Google Earth Engine	print(x.lt(y).and(z));	1

3. The computer returns true because x is a number that is less than y AND is less than z. Had we used the operator "not", the output would have been false. For an additional reference of operators in all three languages, please see the glossary.

Part 3: Additional Resources

A. Variables and Variable Types

1. Python
   i. http://www.tutorialspoint.com/python/python_variable_types.htm
2. R
   i. http://www.tutorialspoint.com/r/r_data_types.htm
3. JavaScript
   i. https://developers.google.com/earth-engine/tutorial_js_01#basic-javascript-data-types

B. Operators

1. Python
   i. http://www.tutorialspoint.com/python/python_basic_operators.htm
2. R
   i. http://www.tutorialspoint.com/r/r_operators.htm
3. JavaScript
   i. https://developers.google.com/earth-engine/getstarted
   ii. https://developers.google.com/earth-engine/image_math
   iii. https://developers.google.com/earth-engine/image_relational

Congratulations: You have completed this exercise. You have now been introduced to some of the basic concepts of scripting. In the next exercise you will build on this knowledge so that you may become familiar with all of the necessary concepts for writing your own script in later exercises.