EXERCISE 2
Variables and Operators

Introduction

This exercise introduces ways to store data and values as variables. It then goes over several operations you can perform on the variable once it has been created. Throughout this exercise you should get comfortable with your environment, making comments, and print statements. The numbered instructions provide information about what the commands do, and the tables give examples about what code needs to be typed in your chosen language.

Objectives

- Understand how data is stored as a variable
- Introduce different variable types
- Preform basic operations on a variable

USDA Non-Discrimination Statement

In accordance with Federal civil rights law and U.S. Department of Agriculture (USDA) civil rights regulations and policies, the USDA, its Agencies, offices, and employees, and institutions participating in or administering USDA programs are prohibited from discriminating based on race, color, national origin, religion, sex, gender identity (including gender expression), sexual orientation, disability, age, marital status, family/parental status, income derived from a public assistance program, political beliefs, or reprisal or retaliation for prior civil rights activity, in any program or activity conducted or funded by USDA (not all bases apply to all programs). Remedies and complaint filing deadlines vary by program or incident.

Persons with disabilities who require alternative means of communication for program information (e.g., Braille, large print, audiotape, American Sign Language, etc.) should contact the responsible Agency or USDA's TARGET Center at (202) 720-2600 (voice and TTY) or contact USDA through the Federal Relay Service at (800) 877-8339. Additionally, program information may be made available in languages other than English.

To file a program discrimination complaint, complete the USDA Program Discrimination Complaint Form, AD-3027, found online at How to File a Program Discrimination Complaint and at any USDA office or write a letter addressed to USDA and provide in the letter all of the information requested in the form. To request a copy of the complaint form, call (866) 632-9992. Submit your completed form or letter to USDA by: (1) mail: U.S. Department of Agriculture, Office of the Assistant Secretary for Civil Rights, 1400 Independence Avenue, SW, Washington, D.C. 20250-9410; (2) fax: (202) 690-7442; or (3) email: program.intake@usda.gov.

USDA is an equal opportunity provider, employer, and lender.

Table of Contents

Part 1: Variables and Statements

A. Variables

A variable stores 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, Google Earth Engine, and R are 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. To begin, we're going to create the variable **x** then assign **x** a value of 5

Language	Environment	Code
Python	IDLE or ArcPro	`x = 5`
R	RStudio	`x <- 5`
JavaScript	GEE	`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. The "ee" stands for Earth Engine and it means that this number is an Earth Engine object. If you would like to read more about these objects in GEE click here.

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).

2. Run the script
   i. **Python 2**: Click **Run**, then Run Module, in the Python window. It will prompt you to save the script first.
   ii. **Python 3**: Click **Run** or press **cntrl+Enter**
   i. **R**: highlight the entire script, then click **Run**. You will see the code run in the console at the bottom left.
   ii. **JavaScript**: 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.

B. Print Statement

A statement is an instruction given to the machine. 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.

1. To confirm the variable now exists, enter a print statement. A **print()** statement is an instruction sent to the computer that tells it to output information

Language	Environment	Code	Outputs
Python 2	IDLE	`print x`	`5`
Python 3	ArcPro	`print(x)`	`5`
R	RStudio	`x`	`[1] 5`
JavaScript	GEE	`print(x)`	`5`

2. Your full script should now read:

Language	Environment	Full Script

Python 2	IDLE	`# Header`
		`x=5`
		`print x`

Python 3	ArcPro	`# Header`
		`x=5`
		`print(x)`

R	RStudio	`# Header`
		`x <- 5`
		`x`

JavaScript	GEE	`// Header`
		`var x = ee.Number(5);`
		`print(x)`

Run the script and notice the output (5) that is listed in the table above.

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

Variables don't have to be numbers. Variables can take several other forms including strings and lists. In this exercise you'll learn about some common variable types that are used in all three languages.

1. Strings are variables that hold text or characters. 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 the variable **y** and set it equal to **"five"**

Language	Environment	Enter
Python	IDLE or ArcPro	`y="five"`
R	RStudio	`y<-"five"`
JavaScript	GEE	`var y=ee.String("five");`

   The variable y is now equal to a string of characters, not the number 5.

2. How do we know the variable type? We can use the **type** command. Put a **type** command in a print statement to see the output!

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

   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).

3. Try giving the same command for the variable x

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

   This may be an unexpected result. JavaScript provided number, 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. If you would like to learn more about number types please see the additional resources section or the glossary.

4. Lists are ordered collections of values. 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**

Language	Environment	Enter
Python	IDLE or ArcPro	`num_list = [9,4,7,1]`
R	RStudio	`num_list = c(9,4,7,1)`
JavaScript	GEE	`var num_list = ee.List([9,4,7,1]);`

   Print out **num_list** using a print statement to see the list of numbers
5. You can store different variable types in a list. Create one more list called **mix_list** and fill it with the number **5** and the string **"five"**

Language	Environment	Enter
Python	IDLE or ArcPro	`mix_list = [5, "five"]`
R	RStudio	`mix_list = c(5, 'five')`
JavaScript	GEE	`var mix_list = ee.List([5, 'five']);`

   Add another print statement for **mix_list** and **Run** the script to see the output

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: Operations

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

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

1. Begin by setting the variable **y** equal to **7**, we already set **y = "five"** so what happens if we set **y** equal to **7** by adding a new line of code?

Language	Environment	Enter
Python	IDLE or ArcPro	`y=7`
R	RStudio	`y<-7`
JavaScript	GEE	`var y = ee.Number(7);`

   Print the variable **y** to see the result! We overwrote the variable **y** so it is now equal to **7**

2. Create the variable **z** and set it equal to **x+y**

Language	Environment	Enter
Python	IDLE and ArcPro	`z = x+y`
R	RStudio	`z <- x+y`
JavaScript	GEE	`var z = x.add(y);`

3. **Print** the variable **z** and **Run** the script

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

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 or ArcPro	`print(x>y)`	`False`
R	RStudio	`x>y`	`[1] FALSE`
JavaScript	Google Earth Engine	`print(x.gt(y))`	`0`

1. 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 or ArcPro	`print(x!=z)`	`True`
R	RStudio	`x!=z`	`[1] TRUE`
JavaScript	Google Earth Engine	`print(x.neq(z))`	`1`

2. 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 or ArcPro	`x=2` `y="2"`
R	RStudio	`x<-2` `y<-"2"`
JavaScript	Google Earth Engine	`var x = ee.Number(2);` `var y = ee.String("2");`

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

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

4. 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 let's 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 or ArcPro	`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**. What would the output be if we use the operator OR? Check it out for yourself!
4. Save your script and continue to Exercise 3

Part 3: Additional Resources

A. Variables and Variable Types

1. Python
   - Python Variable Types
   - Software Carpentry: Python Variables
2. R
   - R Data Types
3. Google Earth Engine/JavaScript
   - Beginner's Cookbook
   - Google Earth Engine Tutorials

B. Operators

1. Python
   - Python Operators
2. R
   - R Operators
3. JavaScript
   - Getting Started with Earth Engine
   - Image Math in Earth Engine
   - Image Relational

Congratulations! You have completed this exercise. You should have a basic understanding of how variables work in scripting!