Python Basics

Hi Class,

This tutorial will teach you the basics of Python, which is required by the course Labs. It is not necessary to have any prior programming experience.

The tool you are using is Google Colab. It allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to business/data analytics and education.

Colab includes both Text Cells and Code Cells. You can double click a cell to edit it. This cell is a Text Cell. Please go through all the following cells, read the text, and run the code. You can fold/unfold text or execute code by clicking the arrow buton on the left of cell.

Note that the best way to learn is to get your hands dirty! thus, feel free to modify the code and check whether the results make sense to you or not.

If you have any questions, post them on the Canvas Discussion board.

Happy learning!

```
Start coding or generate with AI.
```

Double-click (or enter) to edit

Objectives

1. Execute Python code.
2. Explain Python syntax, variables, data types, and lists.
3. Call Python functions.
4. Load and use the Python library.

What is Python?

Python is a general-purpose programming language, which is also very popular in Data Science. Python has a simple syntax similar to the English language, which allows you to express powerful ideas in fewer lines of code while being very readable.

Why Python?

Python can take your marketing analytics to the next level. It helps marketing professionals get more insights from data, make better-informed data-driven decisions, automate many routine activities, and increase the ROI from the marketing campaigns.

Execute code

Run the following code, and you will print a message on the screen!

`print()` is a function that prints the specified message to the screen. In Python, we often call functions to achieve various goals.

```
print("Hello, World!")
```

```
Hello, World!
```

Python Indentation

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

The number of spaces is up to you as a programmer, but it has to be at least one. You can remove the space in the following cell and see what will happen.

Note that you have to use the same number of spaces in the same block of code.

```
if 5 > 2:
  print("Five is greater than two!")
  print("done")
```

```
Five is greater than two!
done
```

The meaning of the above code is straightforward. We will discuss more the Python Conditions and If statements later.

and if statements later.

Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

```
#This is a comment.
print("Hello, World!")
```

```
Hello, World!
```

Python Variables

In Python, variables are created when you assign a value to it. Variables are containers for storing data values.

```
x = 5
y = "Hello, World!"
```

You can get the data type of a variable with the `type()` function.

```
x = 5
y = "John"
print(type(x)) # integer data type
print(type(y)) # string data type
```

```
<class 'int'>
<class 'str'>
```

String variables can be declared either by using single or double quotes:

```
x = "John"
# is the same as
x = 'John'
```

Note that Variable names are **case-sensitive**. The following code will create two variables:

```
a = 4
A = "Sally"
#A will not overwrite a
```

Python Numbers

The basic numeric types in Python are:

- int
- float

Variables of numeric types are created when you assign a value to them:

```
x = 1    # int
y = 2.8  # float
```

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

```
x = 1
y = 35656222554887711
z = -3255522

print(type(x))
print(type(y))
print(type(z))
```

```
<class 'int'>
<class 'int'>
<class 'int'>
```

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

```
x = 1.10
y = 1.0
z = -35.59

print(type(x))
print(type(y))
print(type(z))
```

```
<class 'float'>
<class 'float'>
<class 'float'>
```

Python String

Strings in python are surrounded by either single quotation marks, or double quotation marks.

`'hello'` is the same as `"hello"`.

As you saw before, you can display a string literal with the `print()` function.

Slicing

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

Note that the index **starts from 0**, and the end index is not included.

```
#Get the characters from position 2 to position 5 (not included):
b = "Hello, World!"
print(b[2:5])
```

```
llo
```

Slice From the Start

By leaving out the start index, the range will start at the first character:

```
b = "Hello, World!"
print(b[:5])
print(b[-2])
```

```
Hello
d
```

Slice To the End

By leaving out the end index, the range will go to the end:

```
# Get the characters from position 2, and all the way to the end:
b = "Hello, World!"
print(b[2:])
```

```
llo, World!
```

Negative Indexing

Use negative indexes to start the slice from the end of the string.

The following code get the characters:

From: "o" in "World!" (position -5)

To, but not included: "d" in "World!" (position -2):

```
b = "Hello, World!"
print(b[-5:-2])
```

```
orl
```

```
b = "Hello, World!"
print(b[-2:-5])
```

String Concatenation

To concatenate, or combine, two strings you can use the + operator.

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

```
HelloWorld
```

Double-click (or enter) to edit

Python Booleans

Boolean Values

In programming you often need to know if an expression is True or False.

You can evaluate any expression in Python, and get one of two answers, True or False.

When you compare two values, the expression is evaluated and Python returns the Boolean answer:

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

```
True
False
False
```

Python Comparison Operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Python List

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

Lists are created using square brackets:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

```
['apple', 'banana', 'cherry']
```

List Items

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

Ordered

When we say that lists are ordered, it means that the items have a defined order, and that order will not change.

If you add new items to a list, the new items will be placed at the end of the list.

Changeable

The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

Allow Duplicates

Since lists are indexed, lists can have items with the same value:

```
# Lists allow duplicate values:
thislist = ["apple", "banana", "cherry", "apple", "cherry"]
print(thislist)
```

```
['apple', 'banana', 'cherry', 'apple', 'cherry']
```

List Length

To determine how many items a list has, use the `len()` function:

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

```
3
```

List Items - Data Types

List items can be of any data type:

```
list1 = ["apple", "banana", "cherry"]
list2 = [1, 5, 7, 9, 3]
list3 = [True, False, False]
```

A list can contain different data types:

```
list1 = ["abc", 34, True, 40, "male"]
```

List append() Method

The `append()` method appends an element to the end of the list.

```
fruits = ['apple', 'banana', 'cherry']
fruits.append("orange")
print(fruits)
```

```
['apple', 'banana', 'cherry', 'orange']
```

Python Conditions and If statements

An "if statement" is written by using the `if` keyword.

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
b is greater than a
```

In this example we use two variables, a and b, which are used as part of the if statement to test whether b is greater than a. As a is 33, and b is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

Elif

The `elif` keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

```
a and b are equal
```

In this example a is equal to b, so the first condition is not true, but the elif condition is true, so we print to screen that "a and b are equal".

Else

The `else` keyword catches anything which isn't caught by the preceding conditions.

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a is greater than b
```

In this example a is greater than b, so the first condition is not true, also the elif condition is not

true, so we go to the else condition and print to screen that "a is greater than b".

You can also have an else without the elif:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

```
b is not greater than a
```

And

The `and` keyword is a logical operator, and is used to combine conditional statements:

```
# Test if a is greater than b, AND if c is greater than a:
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

```
Both conditions are True
```

Or

The `or` keyword is a logical operator, and is used to combine conditional statements:

```
# Test if a is greater than b, OR if a is greater than c:
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

```
At least one of the conditions is True
```

Python For Loops

A for loop is used for iterating over a sequence (sucha as a list).

With the for loop we can execute a set of statements, once for each item in a list.

```
# Print each fruit in a fruit list:
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
  print(fruit)
```

```
apple
banana
cherry
```

Even strings are iterable objects, they contain a sequence of characters:

```
# Loop through the letters in the word "banana":
for x in "banana":
  print(x)
```

```
b
a
n
a
n
a
```

Python's `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

```
# Create a sequence of numbers from 3 to 5, and print each item in the sequence:
x = range(3, 6)
for n in x:
  print(n)
```

```
3
4
5
```

```
x = range(3, 6, 2)
for n in x:
  print(n)
```

```
3
5
```

Python Functions

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result

A function can return data as a result.

Creating a Function

In Python a function is defined using the def keyword:

```
def my_function():
  print("Hello from a function")
```

Calling a Function

To call a function, use the function name followed by parenthesis:

```
def my_function():
  print("Hello from a function")

my_function()
```

```
Hello from a function
```

Arguments

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

```
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

```
Emil Refsnes
Tobias Refsnes
Linus Refsnes
```

Python library

Python Libraries are a set of useful functions that eliminate the need for writing codes from

scratch. There are over 137,000 python libraries present in 2020! Using libraries analytics projects can be done very productively.

A library is often imported by seting an alien name. The function in library is called with the prefix of alien name:

```
# print the current working directory
import os as o
cwd = o.getcwd()
print(cwd)
```

```
     /content
```

```

```