

Lecture 01: Introduction to Python

February 6, 2023

1 Introduction to Python

Slides modified from [Pierian Data](#)

2 What is Python and why use it

- **high-level**,
- **interpreted**,
- **general-purpose** programming language that is used for a wide range of applications.
- It is **easy to learn**, and
- **powerful**.

3 Why is it called Python?

When he began implementing Python, Guido van Rossum (left) was also reading the published scripts from “Monty Python’s Flying Circus” (Right), a BBC comedy series from the 1970s. Van Rossum thought he needed a name that was short, unique, and slightly mysterious, so he decided to call the language Python. –General Python FAQ

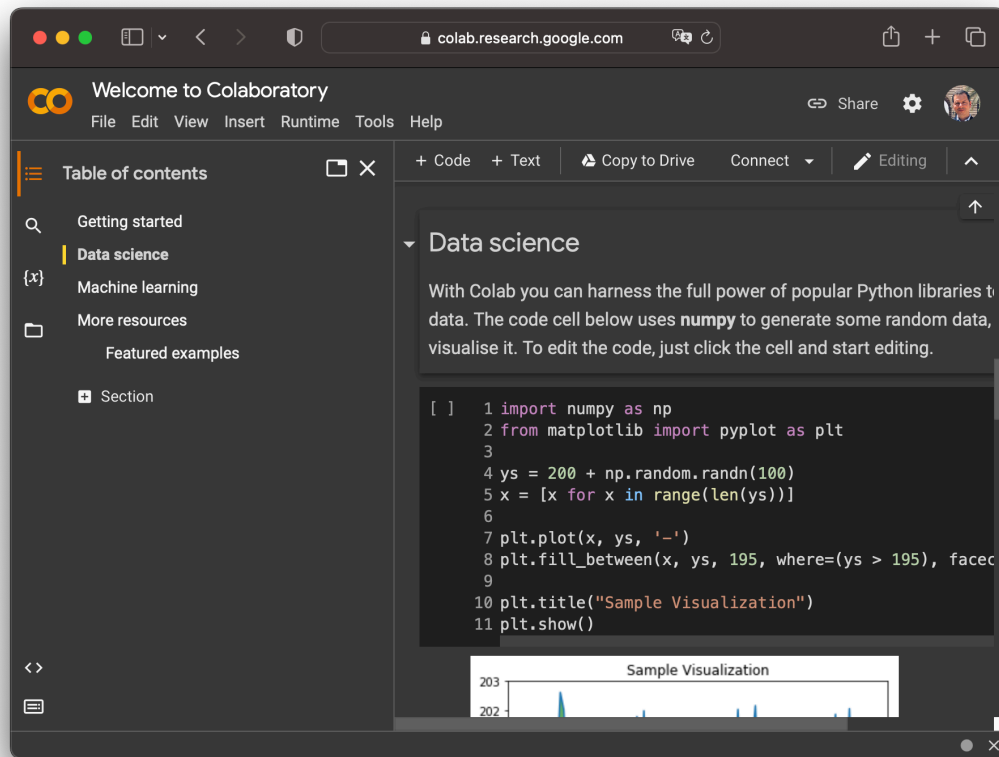
```
[1]: from IPython.display import display, Image
display(Image(filename="images/python.png"))
```



4 Working with Python using Google Colab

Homepage: <https://colab.research.google.com/> (runs online, cloud-computing like)

```
[2]: display(Image(filename="images/colab.png"))
```



5 Working with Python using JupyterLab Desktop

Homepage: <https://github.com/jupyterlab/jupyterlab-desktop> (runs offline, desktop)

```
[3]: display(Image(url="https://raw.githubusercontent.com/jupyterlab/
↳jupyterlab-desktop/master/media/jupyterlab-desktop.png"))
```

```
<IPython.core.display.Image object>
```

6 First Things First

As with any programming course, here is the Hello World! in Python.

```
[4]: print ("Hello World!")
```

```
Hello World!
```

7 Variable

A variable is a named storage location used to hold a value. The value of a variable can be changed and it can be used in expressions and operations

8 Variable Assignment

- names can not start with a number
- names can not contain spaces, use `_` instead
- names can not contain any of these symbols: `'",<>/?|\!@#%^&*~--+`
- according to Style Guide for Python Code ([PEP8](#)), it's considered best practice that names are lowercase with underscores
- avoid using Python built-in keywords like `list` and `str`
- avoid using the single characters `l` (lowercase letter el), `O` (uppercase letter oh) and `I` (uppercase letter eye) as they can be confused with `1` and `0`

9 Dynamic Typing

Python uses *dynamic typing*, meaning you can reassign variables to different data types. This makes Python very flexible in assigning data types; it differs from other languages that are statically typed.

```
[5]: my_cat = 2
      my_cat
```

```
[5]: 2
```

```
[6]: my_cat = ['Basbousa', 'Lucy']
      my_cat
```

```
[6]: ['Basbousa', 'Lucy']
```

10 Pros and Cons of Dynamic Typing

- Pros of Dynamic Typing
 - very easy to work with
 - faster development time
- Cons of Dynamic Typing
 - may result in unexpected bugs!

11 Assigning Variables

Variable assignment follows `name = object`, where a single equals sign `=` is an assignment operator

```
[7]: a = 5
      a
```

```
[7]: 5
```

12 Reassigning Variables

Python lets you reassign variables with a reference to the same object.

```
[8]: a = a + 10  
a
```

```
[8]: 15
```

There's actually a shortcut for this. Python lets you add, subtract, multiply and divide numbers with reassignment using `+=`, `-=`, `*=`, and `/=`.

```
[9]: a += 10  
a
```

```
[9]: 25
```

```
[10]: a *= 2  
a
```

```
[10]: 50
```

13 Determining variable type with `type()`

You can check what type of object is assigned to a variable using Python's built-in `type()` function. Common data types include:

```
[11]: type(a)
```

```
[11]: int
```

14 Numbers

Basically there are two types of numbers: - 2 is interger `int` - 2.0 is floating point `float`

Example	Number Type
1,2,-5,1000	Integers
1.2,-0.5,2e2,3E2	Floating point

```
[12]: type(2)
```

```
[12]: int
```

```
[13]: type(2.0)
```

```
[13]: float
```

15 Basic Arithmetic 1/2

```
[14]: 2+1 # Addition
```

```
[14]: 3
```

```
[15]: 2-1 # Subtraction
```

```
[15]: 1
```

```
[16]: 2*2 # Multiplication
```

```
[16]: 4
```

```
[17]: 3/2 # Division
```

```
[17]: 1.5
```

16 Basic Arithmetic 2/2

```
[18]: 3//2 # Floor division (It returns the result of division rounded down to the  
↪nearest integer)
```

```
[18]: 1
```

```
[19]: 2**3 # Powers
```

```
[19]: 8
```

Question: how to calculate the square root of 16?

17 Order of Operations

```
[20]: 2 + 10 * 10 + 3
```

```
[20]: 105
```

```
[21]: (2+10) * (10+3)
```

```
[21]: 156
```

18 Strings

Strings in Python are **text**, such as names, stored as a sequence or a list of characters. For example, Python understands the string 'AUC' to be a sequence of letters in a specific order. This means we will be able to use indexing to grab particular letters (like the first letter A, or the last letter C).

19 Creating a String

To create a string in Python you need to use either single quotes ' or double quotes ".

```
[22]: 'Hello'
```

```
[22]: 'Hello'
```

```
[23]: 'Hello World!'
```

```
[23]: 'Hello World!'
```

```
[24]: "This is also a string"
```

```
[24]: 'This is also a string'
```

```
[25]: 'I'm using single quotes, but this will create an error'
```

```
Cell In[25], line 1
    'I'm using single quotes, but this will create an error'
    ^
SyntaxError: invalid syntax
```

```
[26]: 'Now I\'m ready to use the single quotes inside a string!' # Using escape_↵
      ↪character
```

```
[26]: "Now I'm ready to use the single quotes inside a string!"
```

```
[27]: "Now I'm ready to use the single quotes inside a string!" # Using double quotes
```

```
[27]: "Now I'm ready to use the single quotes inside a string!"
```

20 Printing a String

Using Jupyter notebook with just a string in a cell will automatically output strings, but the correct way to display strings in your output is by using a **print** function.

```
[28]: 'Hello World'
```

```
[28]: 'Hello World'
```

```
[29]: 'Hello World 1'
      'Hello World 2'
```

```
[29]: 'Hello World 2'
```

```
[30]: print('Hello World 1')
      print('Hello World 2')
```

```
Hello World 1
Hello World 2
```

```
[31]: print('Hello World 1\nHello World 2') # using \n for new line
```

```
Hello World 1
Hello World 2
```

21 String Indexing 1/3

Since strings are a sequence, we can use brackets `[]` after an object to call its index. We should also note that indexing **starts at 0** for Python.

```
[32]: name = 'Emma'
      name
```

```
[32]: 'Emma'
```

```
[33]: name[0]
```

```
[33]: 'E'
```

```
[34]: name[1]
```

```
[34]: 'm'
```

```
[35]: name[-1]
```

```
[35]: 'a'
```

22 String Indexing 2/3

```
[36]: name[:2]
```

```
[36]: 'Em'
```

```
[37]: name[2:]
```

```
[37]: 'ma'
```

```
[38]: name[::-1]
```

```
[38]: 'Emma'
```

23 String Indexing 3/3

```
[39]: name[:2]
```

```
[39]: 'Em'
```

What will be the output of `name[::-1]`

24 String Properties 1/3

Strings are **immutable** i.e., once a string is created, the elements within it can not be changed or replaced.

```
[40]: name
```

```
[40]: 'Emma'
```

```
[41]: name[0] = 'e'
```

```
-----  
TypeError                                Traceback (most recent call last)  
Cell In[41], line 1  
----> 1 name[0] = 'e'  
  
TypeError: 'str' object does not support item assignment
```

25 String Properties 2/3

So if we need to change the value of a string, we need to **reassign**:

```
[42]: name = name + " Stone"  
name
```

```
[42]: 'Emma Stone'
```

```
[43]: display(Image(url="https://api.time.com/wp-content/uploads/2016/12/  
    ↪emma-stone-lalaland.jpg", width = 400) )
```

<IPython.core.display.Image object>

26 String Properties 3/3

```
[44]: name * 5
```

```
[44]: 'Emma StoneEmma StoneEmma StoneEmma StoneEmma Stone'
```


27 Built-in String Method

In Python, we can call objects' methods with a period and then the method name in the following form: `object.method(parameters)`. And here are some built-in methods in strings:

```
[45]: name.upper() # Convert to upper case
```

```
[45]: 'EMMA STONE'
```

```
[46]: name.lower() # Convert to lower case
```

```
[46]: 'emma stone'
```

```
[47]: name.split() # Split by a separator (the default are white spaces)
```

```
[47]: ['Emma', 'Stone']
```

```
[48]: name.replace("m", "M")
```

```
[48]: 'EMMa Stone'
```

```
[49]: display(Image(url="https://miro.medium.com/v2/resize:fit:800/format:webp/
↳0*6GbsTL8b7L2EBvu1.jpg", width = 300))
```

```
<IPython.core.display.Image object>
```

```
[50]: print("Thank you!")
```

```
Thank you!
```