Introduction to Python

"Python is an interpreted, high-level, general-purpose programming language."

-Wikipedia

Python 3

```python
print('hello')              # used to be: print 'hello'

range(10)                   # used to be: xrange(10)

3 / 2 == 1.5                # used to == 1

raise ValueError('x')  # used to be: raise ValueError, 'x'
```

Running Python 3

```
$ python3
Python 3.7.5 (default, Oct 27 2019, 15:43:29)
[GCC 9.2.1 20190909] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Python 3: Interactive Shell

```
>>> print('Hello Class')
Hello Class
>>> 3 + 4
7
>>> exit()
```

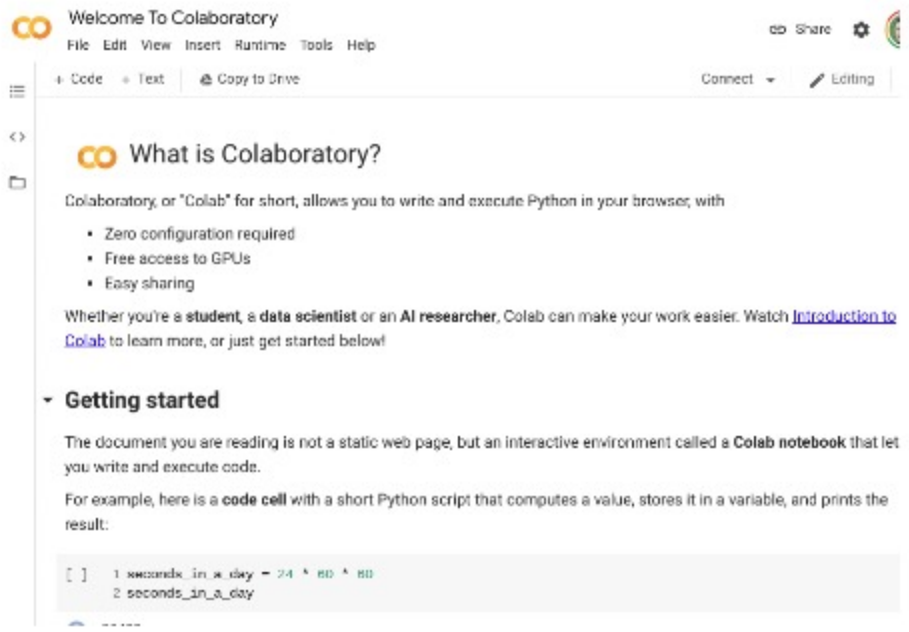
Python 3: Python File

```python
# hello.py

print('Hello Class')
3 + 4
```

```
$ python3 hello.py
Hello Class
```

Python 3: Notebooks



What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

Python Libraries

```python
import re

re.match(...)
```

```python
import tensorflow as tf

tf.keras.Model(...)
```

```python
from sklearn.metrics import mean_squared_error

mean_squared_error(...)
```

White Space Matters

```python
def MyFunction(param):
  print(param)

for i in range(3):
  MyFunction(i)
MyFunction(101)
```

```
0
1
2
101
```

Data Types

```
123     # integer
1.23    # float
"123"   # string
True    # bool
```

Data Types: More About Strings

```
'Guido "BDFL" Van Rossum'    # single-quoted string

'I\'m a pythonista'          # single-quoted string (with escapes)

"I'm a pythonista"            # double-quoted string

'''
Hi. We are about to learn
Python.
'''                          # triple-quoted string

"""
It
    is
        fun!
"""                          # triple-quoted string
```

Data Types: List

```
[]                     # empty list
list()                 # empty list
[1, 2, 3, 4]           # list containing only integers
[1, 2.0, "3", True]    # list containing many types
["a", [1, "b"], 2]     # nested list
```

Data Types: Tuple

```
(,)                               # empty tuple
tuple()                           # empty tuple
(1, 2, 3, 4)                      # tuple containing only integers
1, 2, 3, 4                        # tuple containing only integers
(1, 2.0, "3", True)               # tuple containing many types
("a", (1, "b"), 2, ["x", "y"]) # nested tuple (and list)
```

Data Types: Dictionary

```
{}                             # empty dictionary
dict()                         # empty dictionary
{"a": 1.23, 2: "what"}         # populated dictionary
{"a": {"b": 3}, 2: ["h", 1]}  # nested dictionary (and list)
```

Variables are Dynamically Typed

```
a = 123
print(type(a))

b = 1.23
print(type(b))

c = "123"
print(type(c))
```

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

Variables: Numbers

```
a = 42.12   # float variable
a = a + 1   # add integer to float variable
print(a)

b = 12      # integer variable
b %= 10     # find modulus 10 of b
print (b)

c = a - b   # mixed float and integer math
print(c)
```

```
43.12
2
41.12
```

Variables: Strings

```python
a = "my string"
b = 'your string'

print(a.upper())  # upper case version of string
print(a[1])       # 2nd character of string
print(a[1:4])     # 2nd-4th character of string

print(len(a))     # length of string

c = a + b         # string concatenation
print(c)
```

```
MY STRING
y
y s
9
my stringyour string
```

Variables: Lists

```python
a = ["my", "list", "of", "strings", ["and", "more", "strings"]]

print(a[2])
print(len(a))

a[0] = "My"
print(a[0:3])

print(a[4][1])
```

```
of
5
['My', 'list', 'of']
more
```

Variables: Tuples

```
a = ("my", "tuple", "of", "strings", ("and", "more", "strings"))

print(a[2])
print(len(a))

# a[0] = "My"     # Can't do this!
print(a[0:3])

print(a[4][1])
```

```
of
5
('my', 'tuple', 'of')
more
```

Variables: Dictionaries

```
a = {"x": 12, "y": ["a", "b"], "z": {(2, "a"): "cow"}}

print(a["x"])

a["x"] = 13
print(a["x"])

print(a["y"][1])

print(a["z"][(2, "a")])
```

```
12
13
b
cow
```

Flow Control: if/elif/else

```
a, b, c = 5, 3, 7

if a > b and a > c:
  print(a)
elif b > a and b > c:
  print(b)
else:
  print(c)
```

```
7
```

Flow Control: for - lists and tuples

```
my_list = ["apple", "banana", "coconut"];

for item in my_list:
  print(item)

for i in range(len(my_list)):
  print(i, my_list[i])
```

```
apple
banana
coconut
0 apple
1 banana
2 coconut
```

Flow Control: for - dictionaries

```python
my_dict = {"a": "apple", "b": "banana"}

for key in my_dict:
  print(key, my_dict[key])

for value in my_dict.values():
  print(value)

for key, value in my_dict.items():
  print(key, value)
```

```
b banana
a apple
apple
banana
b banana
a apple
```

Flow Control: for - strings

```python
my_string = "abc"

for c in my_string:
  print(c)

for i in range(len(my_string)):
  print(i, my_string[i])
```

```
a
b
c
0 a
1 b
2 c
```

Flow Control: while

```
count = 0
while count < 3:
  print(count)
  count += 1
```

```
0
1
2
```

None

```
a = None
b = None

if a == b:
  print("None matches")
if a is None:
  print("and a is None)
```

```
None matches
and a is None
```

Your Turn!