

Python Tutorial Series 2019

Basics 1: Python Overview

Ph.D. Bruno C. Quint

Data Process Developer

bquint@gemini.edu

SUSD - GEMINI South, La Serena, Chile



Table of Contents

- What is Python?
- What will you need?
- Python as a terminal
- Python as a script
- Types of variables
- Loops and Control
- Using Methods and Libs
- Gathering Information
- Python Vs iPython



What is Python?

```
print("Hello world!!")
```

High level interpreted
programming language
developed for fast
development



1011000101010010010



What is Python?

```
print("Hello world!!")
```

High level interpreted
programming language
developed for fast
development



```
1011000101010010010
```



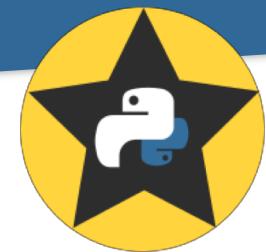
What is Python?

```
print("Hello world!!")
```

High level **interpreted**
programming language
developed for fast
development

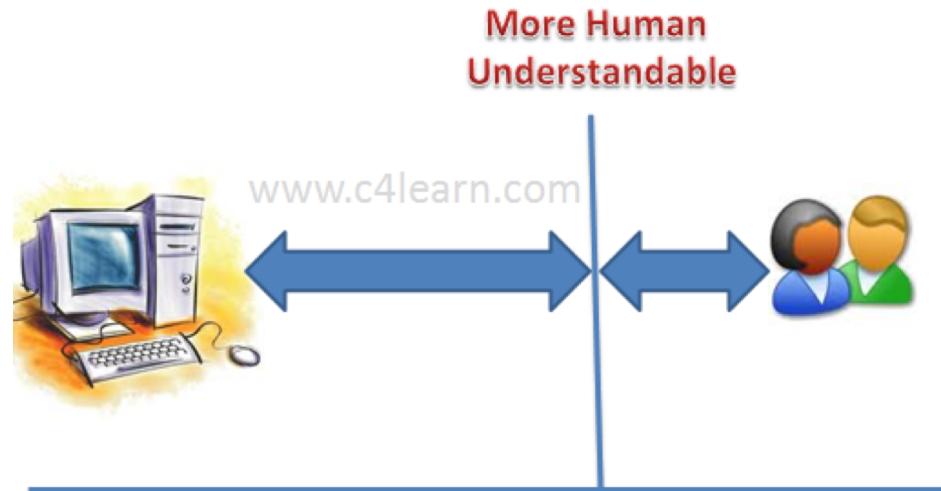


1011000101010010010

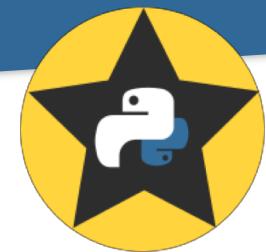


What is Python?

High level interpreted
programming language
developed for fast
development

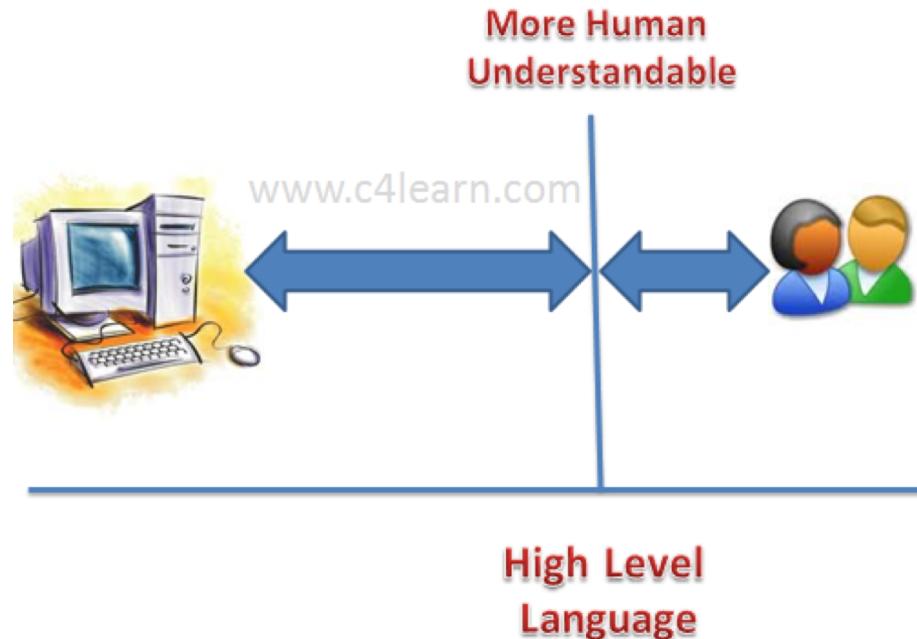


```
if x is not 5:  
    print("Blah")
```



What is Python?

High level interpreted
programming language
**developed for fast
development**

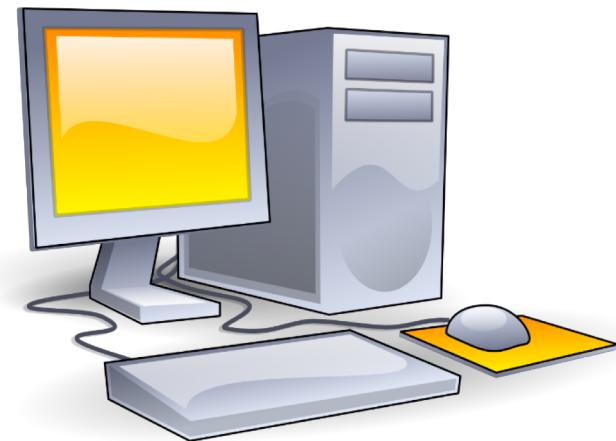


```
if x is not 5:  
    ...print("Blah")
```



What will you need?

- Computer
- Operational System
- Python
- Python Libs
- Text Editors
- Integrated Development Environment (IDE)



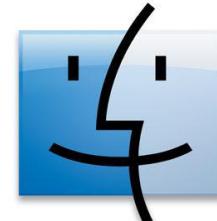


What will you need?

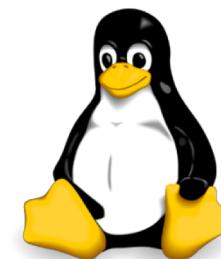
- Computer
- **Operational System**
- Python
- Python Libs
- Text Editors
- Integrated Development Environment (IDE)



Windows



MacOs



Linux™ Distributions



Linux Mint™



Ubuntu™



CentOS™

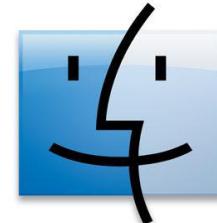


What will you need?

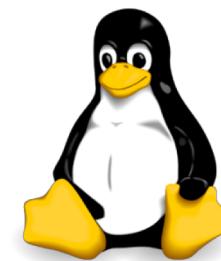
- Computer
- **Operational System**
- Python
- Python Libs
- Text Editors
- Integrated Development Environment (IDE)



Windows



MacOs



Linux™ Distributions



Linux Mint™



Ubuntu™



CentOS™



What will you need?

- Computer
- Operational System
- **Python**
- Python Libs
- Text Editors
- Integrated Development Environment (IDE)



<https://www.python.org/>

2.x

3.x



What will you need?

- Computer
- Operational System
- **Python**
- Python Libs
- Text Editors
- Integrated Development Environment (IDE)



<https://www.python.org/>

2.x

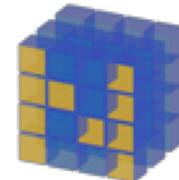
3.x

Python 2.x is legacy,
Python 3.x is the present and future of the language.
<https://wiki.python.org/moin/Python2orPython3>



What will you need?

- Computer
- Operational System
- Python
- **Python Libs**
- Text Editors
- Integrated Development Environment (IDE)



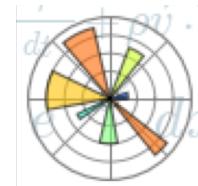
NumPy

Numerical Python



SciPy

Scientific Python



Matplotlib

Python Plotting



AstroPy

Astronomical Python



Jupyter

Python Notebooks



Conda

Python Package Manager



What will you need?

- Computer
- Operational System
- Python
- Python Libs
- **Text Editors**
- Integrated Development Environment (IDE)

Vim

GEdit

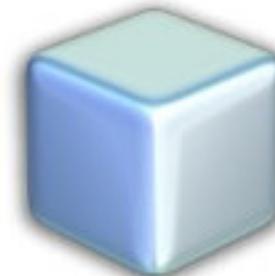
Notepad

EMACS



What will you need?

- Computer
- Operational System
- Python
- Python Libs
- Text Editors
- **Integrated Development Environment (IDE)**



NetBeans



PyCharm





What will you need?

- Computer
- Operational System
- Python
- Python Libs
- Text Editors
- **Integrated Development Environment (IDE)**





Python as a terminal

Getting Started

Start typing

```
$ python
```



Python as a terminal

Getting Started

Start typing

```
$ python
```

Say “Hello World!”

```
>>> print("Hello World")
Hello World
```



Python as a terminal

Getting Started

Start typing

```
$ python
```

Say “Hello World!”

```
>>> print("Hello World")
Hello World
```

Assign a variable

```
>>> x = 2
>>> print(x)
2
```



Python as a terminal

Getting Started

Start typing

```
$ python
```

Say “Hello World!”

```
>>> print("Hello World")
Hello World
```

Assign a variable

```
>>> mag_V = 28.1970
>>> print(mag_V)
28.1970
```



Python as a terminal

Getting Started

Start typing

```
$ python
```

Say “Hello World!”

```
>>> print("Hello World")
Hello World
```

Assign a variable

```
>>> mag_V = 28.1970
>>> print(mag_V)
28.1970
```

Which version am I using?

```
$ python --version
Python 3.6.7 :: Anaconda, Inc.
```



Python as a script

File “**say_hello_world.py**”

```
01 print("Hello world")
```

To run this file:

```
$ cd path_to_my_python_file  
$ python say_hello_world.py  
Hello world
```



Python as a script

File “**say_hello_world.py**”

```
01 #!/path/to/python
02 print("Hello World")
```

To run this file:

```
$ cd path_to_my_python_file
$ chmod a+x say_hello_world.py
$ ./say_hello_world.py
Hello Python
```



Python as a script

File “say_hello_world.py”

```
01 #!/path/to/python  
02 print("Hello \\"
```

```
#!/usr/bin/env python  
#!/usr/bin/python  
#!/usr/local/bin/python  
#! python
```

To run this file:

```
$ cd path_to_my_python_file  
$ chmod a+x say_hello_world.py  
$ ./say_hello_world.py  
Hello Python
```



Python as a script

File “say_hello_world.py”

```
01 #!/path/to/python  
02 print("Hello World")
```

#!/usr/bin/env python
#!/usr/bin/python
#!/usr/local/bin/python
#! python

To run this file:

```
$ cd path_to_my_python_file  
$ chmod a+x say_hello_world.py  
$ ./say_hello_world.py  
Hello Python
```



Python as a script

File “say_hello_world.py”

```
01 #!/path/to/python  
02 print("Hello World")
```

#!/usr/bin/env python
#!/usr/bin/python
#!/usr/local/bin/python
#! python

To run this file:

```
$ cd path_to_my_python_file  
$ chmod a+x say_hello_world.py  
$ ./say_hello_world.py  
Hello Python
```



Python as a script

File “**use_coding.py**”

```
01 #!/path/to/python
02 # -*- coding: utf8 -*-
03 print("áá éé çç")
```

To run this file:

```
path_to_file $ python use_coding.py
```

```
Aá éé çç
```



Python as a script

File “**say_hello_world.py**”

```
01 #!/path/to/python
02 # -*- coding: utf8 -*-
03 """
04 This is a docstring where information
05 about what you are doing and be written.
06 """
07 print("Hello world!")
```

To run this file:

```
path_to_file $ python say_hello_world.py
Hello world!
```



Types of variables

Integers

```
>>> 5 + 4
```

```
9
```

```
>>> 5 - 4
```

```
1
```

```
>>> 5 * 4
```

```
20
```

```
>>> 5 // 4
```

```
1
```

More on operators
(like +, -, /, *, **, >, <)
[HERE](#)



Types of variables

Integers

```
>>> 5 + 4  
9
```

```
>>> 5 - 4  
1
```

```
>>> 5 * 4  
20
```

```
>>> 5 // 4  
1
```

Float/Double

```
>>> 2.0 + 3.0  
5.0
```

```
>>> 2.0 - 3  
-1.0
```

```
>>> 2. * 3  
6.0
```

```
>>> 2 / 3  
0.666666...
```

More on operators
(like +, -, /, *, **, >, <)
[HERE](#)



Types of variables

Integers

```
>>> 5 + 4  
9
```

```
>>> 5 - 4  
1
```

```
>>> 5 * 4  
20
```

```
>>> 5 // 4  
1
```

Float/Double

```
>>> 2.0 + 3.0  
5.0
```

```
>>> 2.0 - 3  
-1.0
```

```
>>> 2. * 3  
6.0
```

```
>>> 2 / 3  
0.666666...
```

More on operators
(like +, -, /, *, **, >, <)
[HERE](#)



Types of variables

Integers

```
>>> 5 + 4  
9
```

```
>>> 5 - 4  
1
```

```
>>> 5 * 4  
20
```

```
>>> 5 // 4  
1
```

Float/Double

```
>>> 2.0 + 3.0  
5.0
```

```
>>> 2.0 - 3  
-1.0
```

```
>>> 2. * 3  
6.0
```

```
>>> 2 / 3  
0.666666...
```

More on operators
(like +, -, /, *, **, >, <)
[HERE](#)



Types of variables

Integers

```
>>> x = 2  
>>> type(x)  
<type 'int'>
```

Float/Double

```
>>> y = 1.  
>>> x = x + y  
>>> type(x)  
<type 'float'>
```



Types of variables

Complex

```
>>> (2 + 3j) + (5 - 4j)  
(7 - 1j)
```

```
>>> (2 + 3j) - (5 - 4j)  
(-3 + 7j)
```

```
>>> (2 + 3j) * (5 - 4j)  
(22 + 7j)
```

```
>>> (2 + 3j) / (5 - 4j)  
(-0.049 + 0.561j)
```

```
>>> c = (1 + 3.j)  
>>> type(c)  
<type 'complex'>
```

More on operators
(like +, -, /, *, **, >, <)
[HERE](#)



Types of variables

Booleans

```
>>> fruit = 'banana'           ← Variable assignment  
>>> fruit == 'apple'  
False  
>>> fruit != 'apple'  
True
```

More on boolean operators
(like +, *, ==, !=, <, >, <=, >=, is, not)
[HERE](#)



Types of variables

Booleans

```
>>> fruit = 'banana'  
>>> fruit == 'apple'  
False  
>>> fruit != 'apple'  
True
```



Assign variable

Compare variables

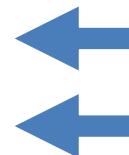
More on boolean operators
(like +, *, ==, !=, <, >, <=, >=, is, not)
[HERE](#)



Types of variables

Booleans

```
>>> fruit = 'banana'  
>>> fruit == 'apple'  
False  
>>> fruit != 'apple'  
True
```



Assign variable

Compare variables

More on boolean operators
(like +, *, ==, !=, <, >, <=, >=, is, not)
[HERE](#)



Types of variables

Booleans

```
>>> fruit = 'banana'  
>>> fruit == 'apple'  
False  
>>> fruit != 'apple'  
True
```

```
>>> mag_v = 15.5  
>>> mag_v < 10.0  
False  
>>> mag_v >= 10.0  
True
```

More on boolean operators
(like +, *, ==, !=, <, >, <=, >=, is, not)
[HERE](#)



Types of variables

Booleans

```
>>> fruit = 'banana'  
>>> fruit == 'apple'  
False  
>>> fruit != 'apple'  
True
```

```
>>> mag_V = 15.5  
>>> mag_V < 10.0  
False  
>>> mag_V >= 10.0  
True
```

```
>>> mag_V = 15.5  
>>> test = mag_V < 10.0  
>>> print(test)  
False
```

More on boolean operators
(like +, *, ==, !=, <, >, <=, >=, is, not)
[HERE](#)



Types of variables

Booleans

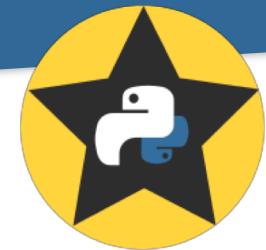
```
>>> True + False  
True
```

```
>>> True or False  
True
```

```
>>> True * False  
False  
>>> True and False  
False
```

```
>>> not False  
True  
>>> not True  
False
```

More on boolean operators
(like +, *, ==, !=, <, >, <=, >=, is, not)
[HERE](#)



Types of variables

Booleans

```
>>> x = 1  
>>> y = 1.  
>>> x == y  
True  
>>> x is y  
False
```

```
>>> x = 1  
>>> y = 1  
>>> x == y  
True  
>>> x is y  
True
```

More on boolean operators
(like +, *, ==, !=, <, >, <=, >=, is, not)
[HERE](#)



Types of variables

Booleans

```
>>> x = 1  
>>> y = 1.  
>>> x == y  
True  
>>> x is y  
False
```

```
>>> x = 1  
>>> y = 1  
>>> x == y  
True  
>>> x is y  
True
```

More on boolean operators
(like +, *, ==, !=, <, >, <=, >=, is, not)
[HERE](#)



Types of variables

Lists

```
>>> x = [2, 4, 6, 8]
```

Sum of lists (appending)

```
>>> x + x  
[2, 4, 6, 8, 2, 4, 6, 8]
```

Multiplication by integer

```
>>> 3 * x  
[2, 4, 6, 8, 2, 4, 6, 8, 2,  
4, 6, 8]
```



Types of variables

Lists

```
>>> x = [2, 4, 6, 8]
```

Sum of lists (appending)

```
>>> x + x  
[2, 4, 6, 8, 2, 4, 6, 8]
```

Multiplication by integer

```
>>> 3 * x  
[2, 4, 6, 8, 2, 4, 6, 8,  
 4, 6, 8]
```

C-Like Indexing (from 0 to n-1)

```
>>> x[0] # First element  
2
```

```
>>> x[3] # 4th element  
8
```

```
>>> x[-1] # Last element  
8
```



Types of variables

Lists

```
>>> x = [2, 4, 6, 8]
```

```
>>> x[i:j]
```

i - start index

j - last index (exclusive)

Slicing

```
>>> x[0:2]  
[2, 4]
```



Types of variables

Lists

```
>>> x = [2, 4, 6, 8]
```

```
>>> x[i:j]  
i - start index  
j - last index (exclusive)
```

Slicing

```
>>> x[0:2]  
[2, 4]
```

0-index can be omitted

```
>>> x[:2]  
[2, 4]
```



Types of variables

Lists

```
>>> x = [2, 4, 6, 8]
```

```
>>> x[i:j]  
i - start index  
j - last index (exclusive)
```

Slicing

```
>>> x[0:2]  
[2, 4]
```

0-index can be omitted

```
>>> x[:2]  
[2, 4]
```

-1 can be used as limit too!

```
>>> x[1:-1]  
[4, 6]
```



Types of variables

Lists

```
>>> x = [2, 4, 6, 8]
```

```
>>> x[i:j]  
i - start index  
j - last index (exclusive)
```

Slicing

```
>>> x[0:2]  
[2, 4]
```

0-index can be omitted

```
>>> x[:2]  
[2, 4]
```

-1 can be used as limit too!

```
>>> x[1:-1]  
[4, 6]
```

Go until the end of the list!

```
>>> x[2:]  
[6, 8]
```



Types of variables

Lists

```
>>> y = [1, 4, 9, 16, 25, 36, 49]
```

Slicing with steps

```
>>> y[0:6:2]  
[1, 9, 25]
```

```
>>> x[i:j:k]  
i - start index  
j - last index (exclusive)  
k - step
```



Types of variables

Lists

```
>>> y = [1, 4, 9, 16, 25, 36, 49]
```

Slicing with steps

```
>>> y[0:6:2]  
[1, 9, 25]
```

Last index 6 can be omitted

```
>>> y[1:6:2]  
[4, 16, 36]
```

```
>>> x[i:j:k]  
i - start index  
j - last index (exclusive)  
k - step
```



Types of variables

Lists

```
>>> y = [1, 4, 9, 16, 25, 36, 49]
```

Slicing with steps

```
>>> y[0:6:2]  
[1, 9, 25]
```

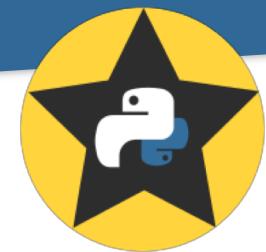
Last index 6 can be omitted

```
>>> y[1:6:2]  
[4, 16, 36]
```

Easily revert lists!

```
>>> y[::-1]  
[6, 5, 4, 3, 2, 1, 0]
```

```
>>> x[i:j:k]  
i - start index  
j - last index (exclusive)  
k - step
```



Types of variables

Lists

```
>>> y = [1, 4, 9, 16, 25, 36, 49]
```

One list can hold variables of different types!

```
>>> z = [5, 2.0, "abc"]
>>> z[1]
2.0
>>> z[2]
"abc"
```



Types of variables

Strings

```
>>> s = 'I am a string'  
>>> r = "I'm another string"
```

Strings behaves like lists

```
>>> s[3]  
'm'
```



Types of variables

Strings

```
>>> s = 'I am a string'  
>>> r = "I'm another string"
```

Strings behaves like lists

```
>>> s[3]  
'm'
```

```
>>> 5 * "xy"  
'xyxyxyxyxy'
```

```
>>> s + r  
"I am a stringI'm another  
string"
```



Types of variables

Strings

```
>>> s = 'I am a string'  
>>> r = "I'm another string"
```

Strings behaves like lists

```
>>> s[3]  
'm'
```

```
>>> 5 * "xy"  
'xyxyxyxyxy'
```

```
>>> s + r  
"I am a stringI'm another  
string"
```

Format using **.format()**

```
>>> x = 7  
>>> print("Number: {:03d}".format(x))  
Number 003
```

[Python – Format mini language specs](#)



Types of variables

Dictionaries

```
>>> fruits = {"orange": 1.5, 'apple': 3}
```



Types of variables

Dictionaries

Assign value to key



Key(word)s



Values





Types of variables

Dictionaries

Creating a dictionary and accessing its elements

```
>>> fruits = {"orange": 1.5, 'apple': 3}  
>>> fruits['apple']  
3
```



Types of variables

Dictionaries

Creating a dictionary and accessing its elements

```
>>> fruits = {"orange": 1.5, 'apple': 3}  
>>> fruits['apple']  
3
```

```
>>> fruits = dict(orange=1.5, apple=3)  
>>> fruits['apple']  
3
```



Types of variables

Dictionaries

Creating a dictionary and accessing its elements

```
>>> fruits = {"orange": 1.5, 'apple': 3}  
>>> fruits['apple']  
3
```

Adding elements to a dictionary

```
>>> fruits['banana'] = None  
>>> print(fruits)  
{'apple': 3, 'orange': 1.5, 'banana': None}
```



Types of variables

Dictionaries

Creating a dictionary and accessing its elements

```
>>> fruits = {"orange": 1.5, 'apple': 3}  
>>> fruits['apple']  
3
```

Check if a dictionary has an element

```
>>> 'apple' in fruits  
True  
>>> 'kiwi' in fruits  
False
```



Types of variables

Exercises

[Click here](#) to access the GitHub Page.

- Review code syntax and make sure you can run it.
- Give a meaningful name for x, y, and z.
- Group code parts only using blank lines.
- Let your code breath by adding spaces before and after =.
- Check if it prints the expected output:

I bought 5 oranges
and I will give you 1.
Then I will divide between
8 people.
Each one will receive 0.5 oranges.



Questions?

