



Python Bootcamp Basics I

PhD Bruno C. Quint
bquint@ctio.noao.edu

Resident astronomer at SOAR Telescope



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?

High level interpreted
programming language
developed for fast
development

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



1011000101010010010



What is Python?

High level interpreted
programming language
developed for fast
development

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



1011000101010010010



What is Python?

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

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

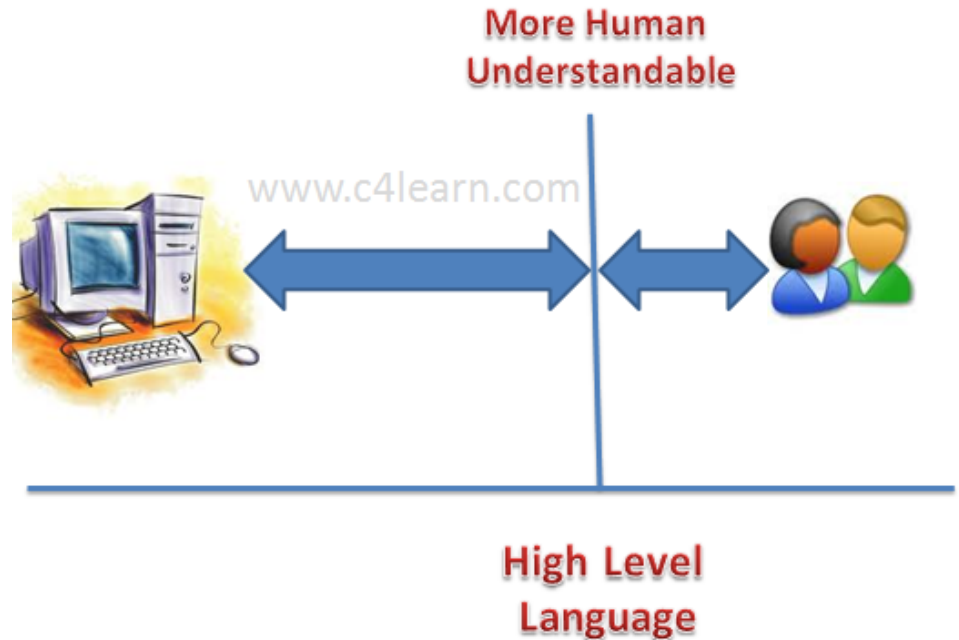


1011000101010010010



What is Python?

High level interpreted programming language developed for fast development

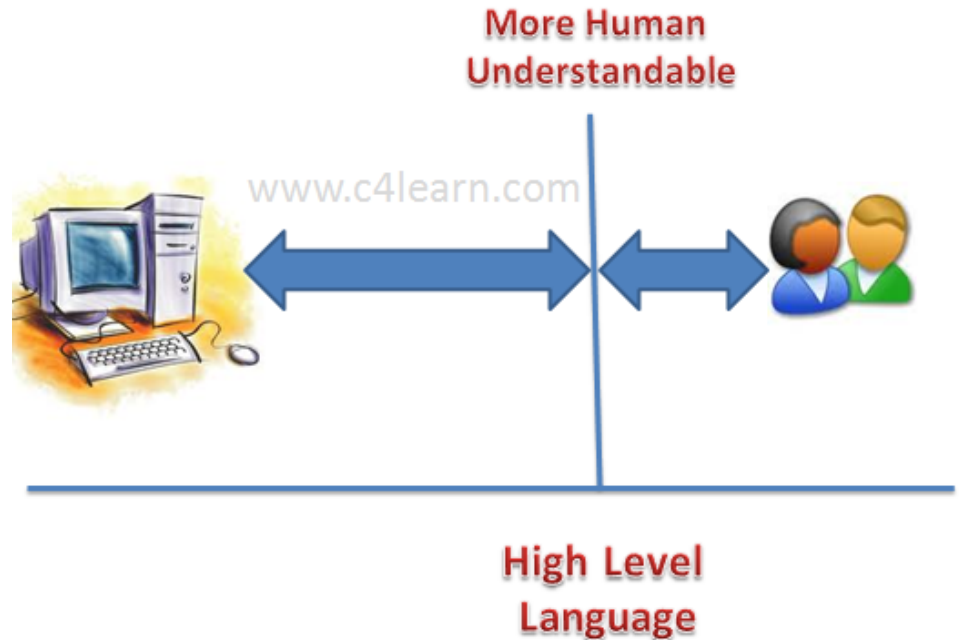


```
if x is not 5:  
    print "Blá"
```



What is Python?

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



```
if x is not 5:  
    print "Blá"
```



What will you need?

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





What will you need?

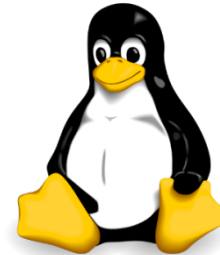
- Computer
- **Operational System**
- Python
- Python Libs
- Text Editors
- Integrated Development Environment (IDE)



Windows



MacOs



Linux™ Distributions



Linux Mint™



Ubuntu™



Fedora™



What will you need?

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



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

2.x

3.x

<https://wiki.python.org/moin/Python2orPython3>



What will you need?

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



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

2.x

3.x

<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



PyFITS



What will you need?

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

GEdit

Vim

Notepad

EMACS



What will you need?

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



NetBeans



PyCharm





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
2.x



Python as a terminal Getting Started

Start typing

```
$ python
```

Say "Hello World!"

```
>>> print("Hello world")  
Hello world
```

Python
2.x and 3.x!



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
2.x and 3.x!



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
2.x and 3.x!



Python as a script

File “**say_hello_world.py**”

```
01 print(“Hello world”)
```

To run this file:

```
path_to_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:

```
path_to_file $ chmod a+x say_hello_world.py  
Path_to_file $ ./say_hello_world.py  
Hello Python
```



Python as a script

File “**use_coding.py**”

```
01 #!/path/to/python  
02 # -*- coding: utf8 -*-  
03 print(“aá eé cç”)
```

To run this file:

```
path_to_file $ python use_coding.py  
Aá eé cç
```



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

```
>>> x = 2
>>> print type(x)
<type 'int'>
>>>
>>> x = x + 2
>>> print x.__class__
<type 'int'>
```

Any assigned **variable** in Python is an **object** and objects have **attributes**.

Float/Double

```
>>> x = 2.0
>>> print type(x)
<type 'float'>
>>>
>>> y = 1
>>> y = x + y
>>> print y.__class__
<type 'float'>
```

Use `.__class__` attribute to find what type is your object.



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)
>>> print c.__class__
<type 'complex'>
```

More on operators
(like +, -, /, *, **, >, <)

[HERE](#)



Types of variables

Booleans

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

More on boolean operators

(like +, *, ==, !=, <, >, <=, >=, is, not)

[HERE](#)



Types of variables

Booleans

```
>>> fruit = 'banana'
>>> fruit == 'maçã'
False
>>> fruit != 'maçã'
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 == 'maçã'
False
>>> fruit != 'maçã'
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 = x
>>> 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 = x
>>> x == y
True
>>> x is y
True
```



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

[HERE](#)



Types of variables

Lists

```
>>> x = [5, 8, 2, 3]
```

Sum of lists (appending)

```
>>> x + x  
[5, 8, 2, 3, 5, 8, 2, 3]
```

Multiplication by integer

```
>>> 3 * x  
[5, 8, 2, 3, 5, 8, 2, 3, 5,  
8, 2, 3]
```



Types of variables

Lists

```
>>> x = [5, 8, 2, 3]
```

Sum of lists (appending)

```
>>> x + x  
[5, 8, 2, 3, 5, 8, 2, 3]
```

Multiplication by integer

```
>>> 3 * x  
[5, 8, 2, 3, 5, 8, 2, 3, 5, 8, 2, 3]
```

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

```
>>> x[0] # First element  
5
```

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

```
>>> x[-1] # Last element  
3
```



Types of variables

Lists

```
>>> x = [5, 8, 2, 3]
```

Slicing

```
>>> x[0:2]  
[5, 8]
```

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



Types of variables

Lists

```
>>> x = [5, 8, 2, 3]
```

Slicing

```
>>> x[0:2]  
[5, 8]
```

```
>>> x[:2]  
[5, 8]
```

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



Types of variables

Lists

```
>>> x = [5, 8, 2, 3]
```

Slicing

```
>>> x[0:2]  
[5, 8]
```

```
>>> x[:2]  
[5, 8]
```

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

```
>>> x[2:-1]  
[2]
```



Types of variables

Lists

```
>>> x = [5, 8, 2, 3]
```

Slicing

```
>>> x[0:2]  
[5, 8]
```

```
>>> x[:2]  
[5, 8]
```

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

```
>>> x[2:-1]  
[2]
```

```
>>> x[2:]  
[2, 3]
```




Types of variables

Lists

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

Slicing

```
>>> y[::2]  
[0, 2, 4, 6]
```

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



Types of variables

Lists

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

Slicing

```
>>> y[::2]  
[0, 2, 4, 6]
```

```
>>> y[1::2]  
[1, 3, 5]
```

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



Types of variables

Lists

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

Slicing

```
>>> y[::2]  
[0, 2, 4, 6]
```

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

```
>>> y[1::2]  
[1, 3, 5]
```

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



Types of variables

Lists

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

One list can hold variables of diferente 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'
```



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"
```




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"
```

Formating strings like C **printf**

```
>>> x = 7
>>> print("%03d" % x)
003
```

<http://pt.wikipedia.org/wiki/Printf>

<http://www.cplusplus.com/reference/cstdio/printf/>



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'
```

Or using **.format()**

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

[Python – Format mini language specs](#)



Types of variables

Dictionaries

Creating a dictionary and accessing its elements

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



Types of variables

Dictionaries

Creating a dictionary and accessing its elements

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

Check if a dictionary has an element

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



Questions?

