



ITA6017

Python Programming

Dr. Arun Pandian J



Course Objectives

- To design and apply programming constructs in python
- To learn how to write loops and decision statements in python
- To learn how to use lists, tuples, and dictionaries in python programs
- To apply embedded programming features in python



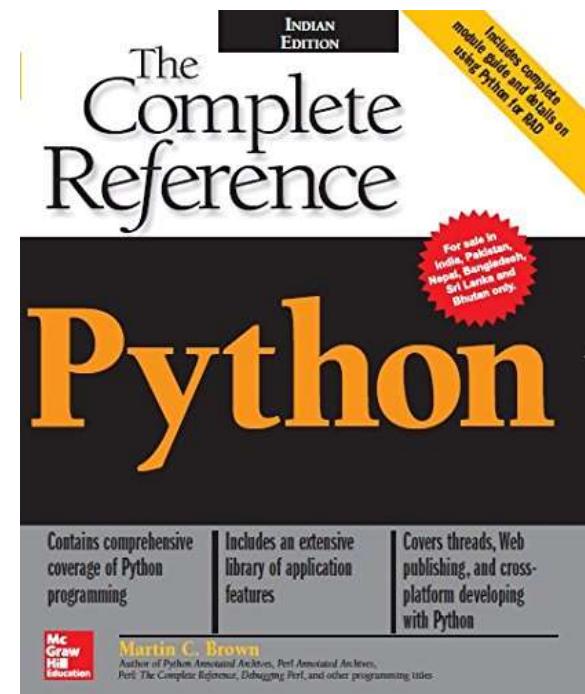
Expected Course Outcomes

1. Apply a solution clearly and accurately in a program using Python
2. Implement a given algorithm as a computer program using Python constructs
3. Demonstrate the implications of specialized data structures in Python
4. Develop simple embedded oriented applications in Python
5. Develop data visualization trends in Python



Text Book

- Martin C. Brown, Python: The Complete Reference, 20 Mar 2018, 4th Edition, McGraw Hill Education, USA.





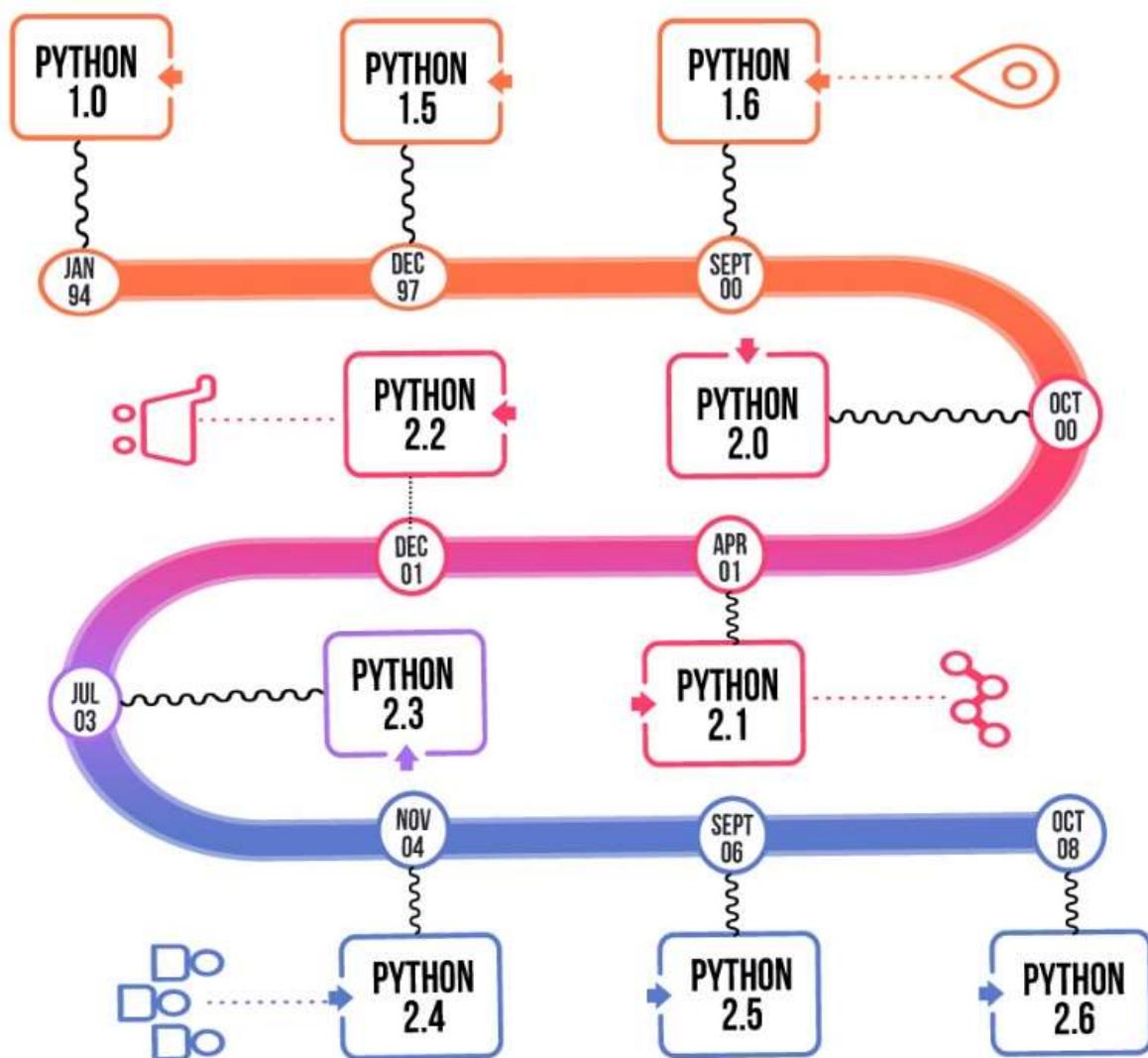
Module:1 Introduction to Python

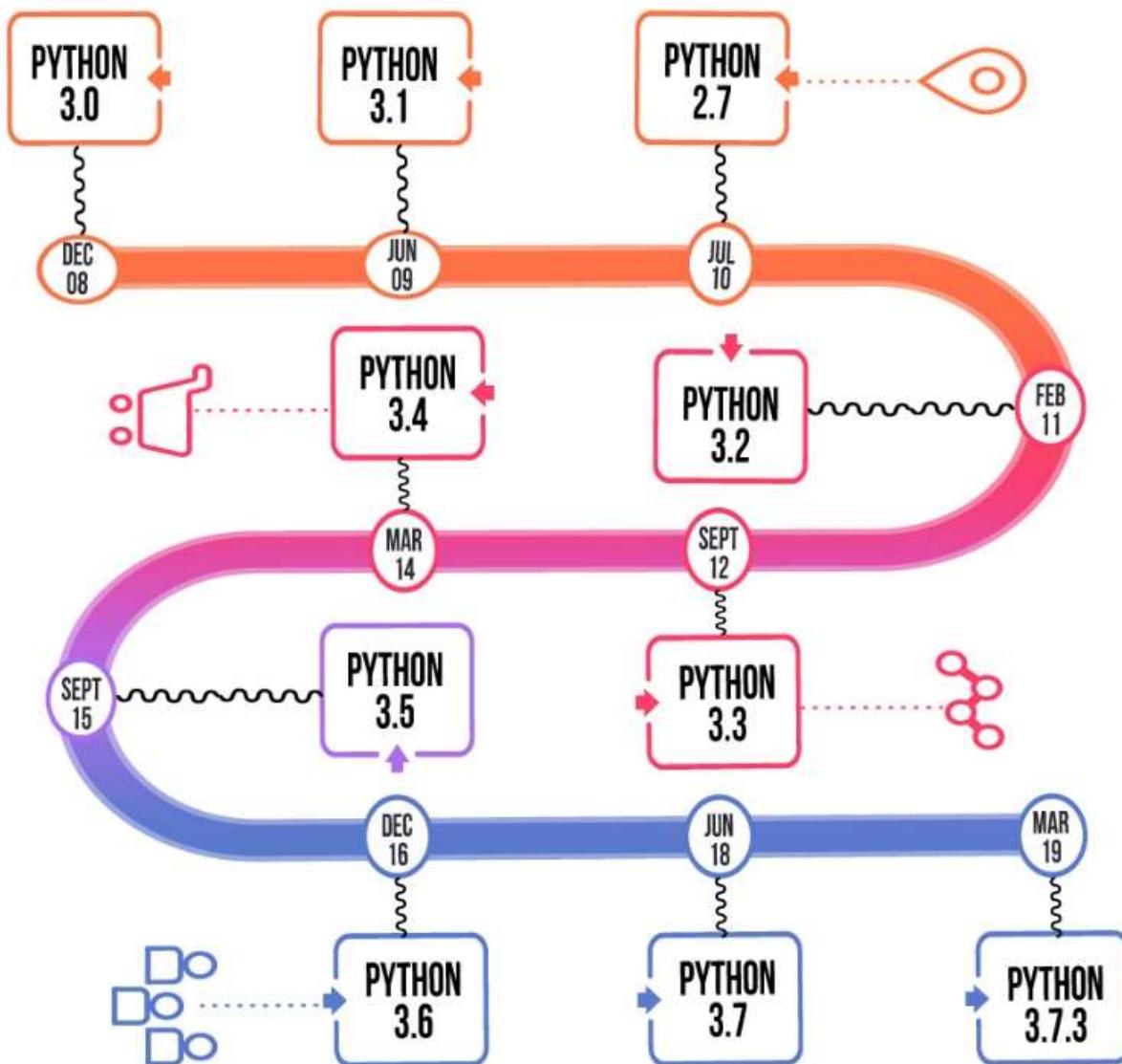
History of Python, Unique features of Python, Demo on IDE, Ipython, Spyder etc., “Hello world” program in Python, Keywords, Identifiers, Reading input from user-Demo, Python Data Types, Declaring and using Numeric data types: int, float, complex and string

(4 Hours)



History







*Current version of python is
3.11.2*

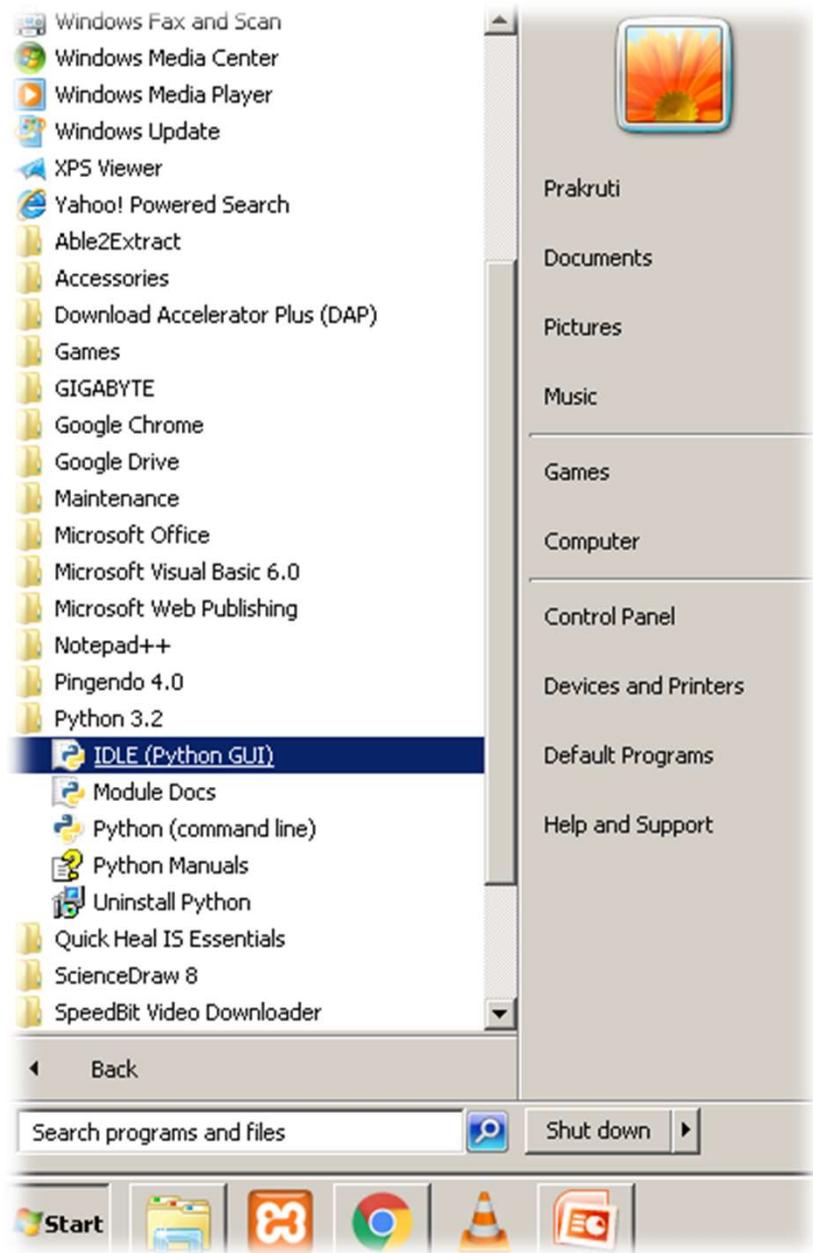


Unique features of Python

- Easy to Code
- Easy to Read
- Free and Open-Source
- Robust Standard Library
- Interpreted
- Portable
- Object-Oriented and Procedure-Oriented
- Extensible
- Expressive
- Support for GUI
- Dynamically Typed



Demo on Python Editors



PYTHON INRACTI VE MODE PYTHON IDLE



PYTHON SHELL

What is Shell?

A shell is usually an "interactive shell", usually termed a REPL which stands for "Read - Execute - Print - Loop". Most interpreted languages offer a REPL interface - whether its LISP, python, BASIC or Javascript or even DOS batch language or Unix Shells. The interpreter is what actually executes the lines of code.



PYTHON SHELL

What is Python Shell or Python Interactive Shell?

The Python interpreter can be used from an interactive shell. The interactive shell is also interactive in the way that it stands between the commands or actions and their execution. ... Python offers a comfortable command line interface with the Python shell, which is also known as the "Python interactive shell".

PYTHON SHELL

IDLE



Python Shell

File Edit Shell Debug Options Windows Help

```
Python 3.2.2 (default, Sep  4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2+1
3
>>>
```



PYTHON BASIC MODES

What are the basic modes of python?

Python has two basic modes:

- 1) Script and
- 2) Interactive.

PYTHON BASIC MODES



1) Script Mode:

The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.



PYTHON BASIC MODES

2) Interactive Mode:

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory.



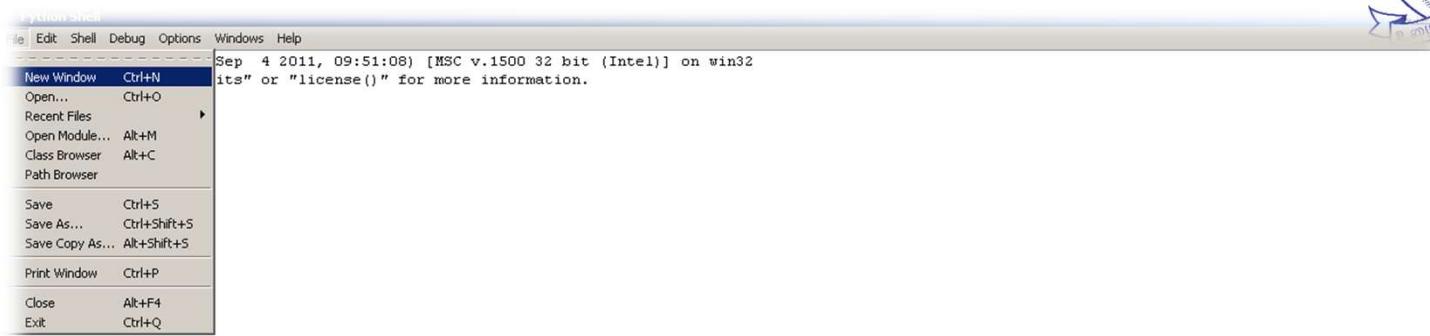
SCRIPT

What is Script?

Scripts are reusable. Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time. Scripts are editable.

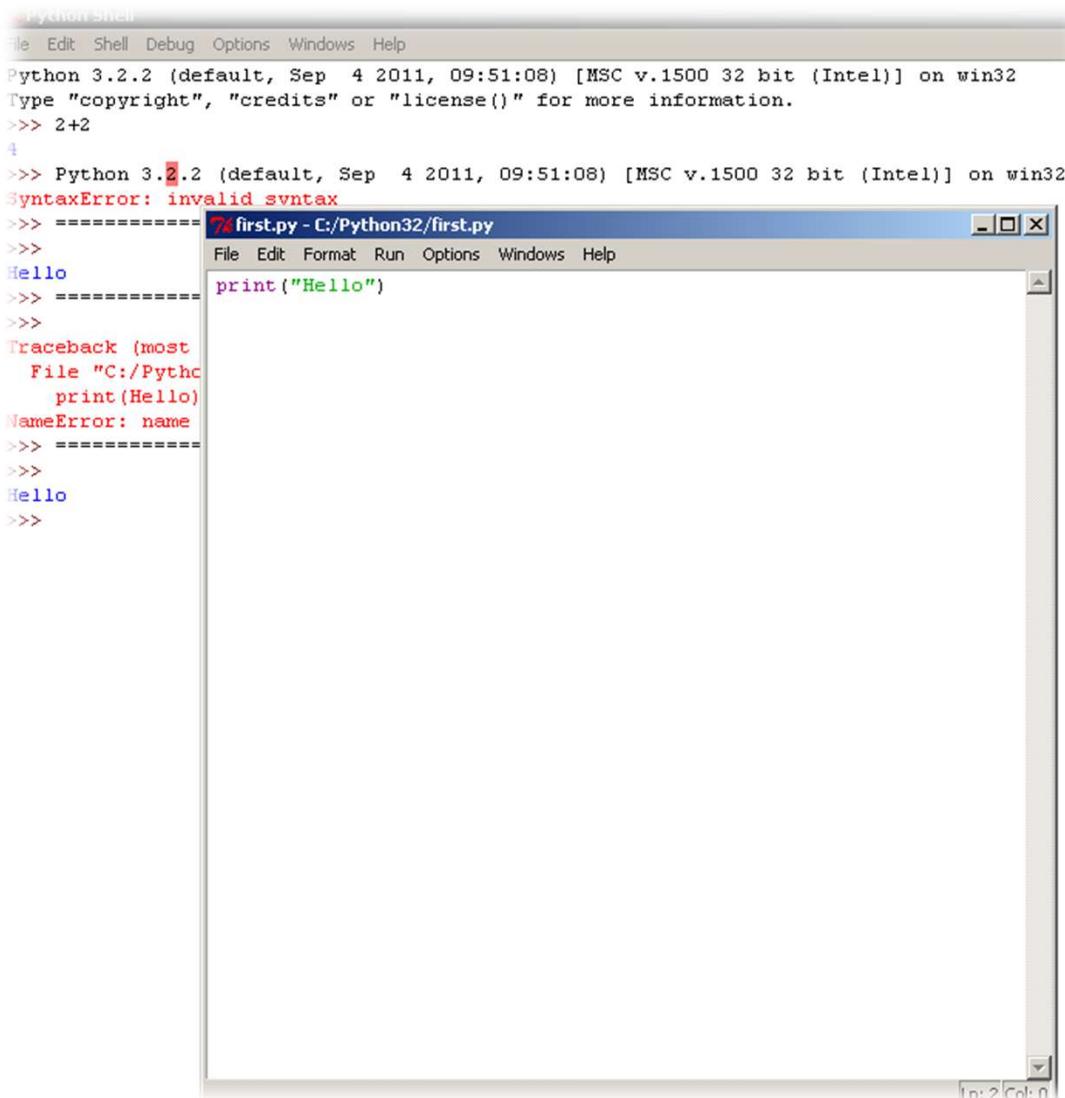


INVOKING SCRIPT MODE





INVOKING SCRIPT MODE



A screenshot showing two windows side-by-side. The left window is a "Python Shell" with a menu bar: File, Edit, Shell, Debug, Options, Windows, Help. The Python version is Python 3.2.2 (default, Sep 4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win32. It shows the following interaction:

```
>>> 2+2
4
>>> Python 3.2.2 (default, Sep 4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win32
SyntaxError: invalid syntax
>>> =====
>>>
Hello
>>> =====
>>>
Traceback (most
  File "C:/Python32/first.py", line 1
    print(Hello)
NameError: name
>>> =====
>>>
Hello
>>>
```

The right window is titled "76first.py - C:/Python32/first.py" and has a menu bar: File, Edit, Format, Run, Options, Windows, Help. It contains the following Python code:

```
print("Hello")
```



Saving Script / Program File

A screenshot of a Windows desktop environment. In the foreground, there is a Python IDLE window. The code editor shows the following Python script:

```
>>> print("Hello")
SyntaxError: invalid syntax
```

The window title is "74 second.PY - C:/Python32/second.PY". Above the code editor, the system tray shows icons for network, battery, and volume.

Overlaid on the IDLE window is a "Save As" dialog box. The dialog box has the following details:

- Save in:** Python32
- Name:** first.py
- Save as type:** Python files (*.py, *.pyw)

The "Save" button is visible at the bottom right of the dialog box.

Click On File
Select Save
Give filename with
py extension

For Example:
first.py
Or
first.PY



Running or Executing Program / Script

A screenshot of a Windows desktop showing a Python Shell window and a code editor window. The Python Shell window at the top shows standard startup text and a simple calculation. Below it, a code editor window titled 'first.py - C:/Python32/first.py' contains a single line of code: 'print("Hello")'. A context menu is open over this line, with 'Run Module F5' highlighted in blue. The status bar at the bottom indicates 'In [2] Col: 0'.

```
Python 3.2.2 (default, Sep  4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.

>>> 2+2
4
>>> Python 3.2.2 (default, Sep  4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win32
SyntaxError: invalid syntax
>>> =====
>>> print("Hello")
| |
| Python Shell
| Check Module Alt+X
| Run Module F5
>>> =====
>>>
>>> Hello
>>>
>>>
```

**Click On Run
Select**

**Run Module
Or
F5**



ANACONDA
Powered by Continuum Analytics®

WORKING IN ANACONDA



ANACONDA NAVIGATOR

A screenshot of the Anaconda Navigator application interface. The left sidebar includes links for Home, Environments, Projects (beta), Learning, Community, Documentation, Developer Blog, and Feedback, along with social media icons for Twitter, YouTube, and GitHub. The main area displays a grid of data science applications: jupyter (notebook 5.0.0), IP[y]: qtconsole (4.3.0), spyder (3.1.4), anypytools (0.10.10), glueviz (0.12.0), orange3 (3.8.0), psyplot-gui (1.0.1), and rstudio (1.1.383). Each application card shows its icon, name, version, and a brief description. Buttons for "Launch" or "Install" are present in each card.

JUPYTER NOTEBOOK IDE



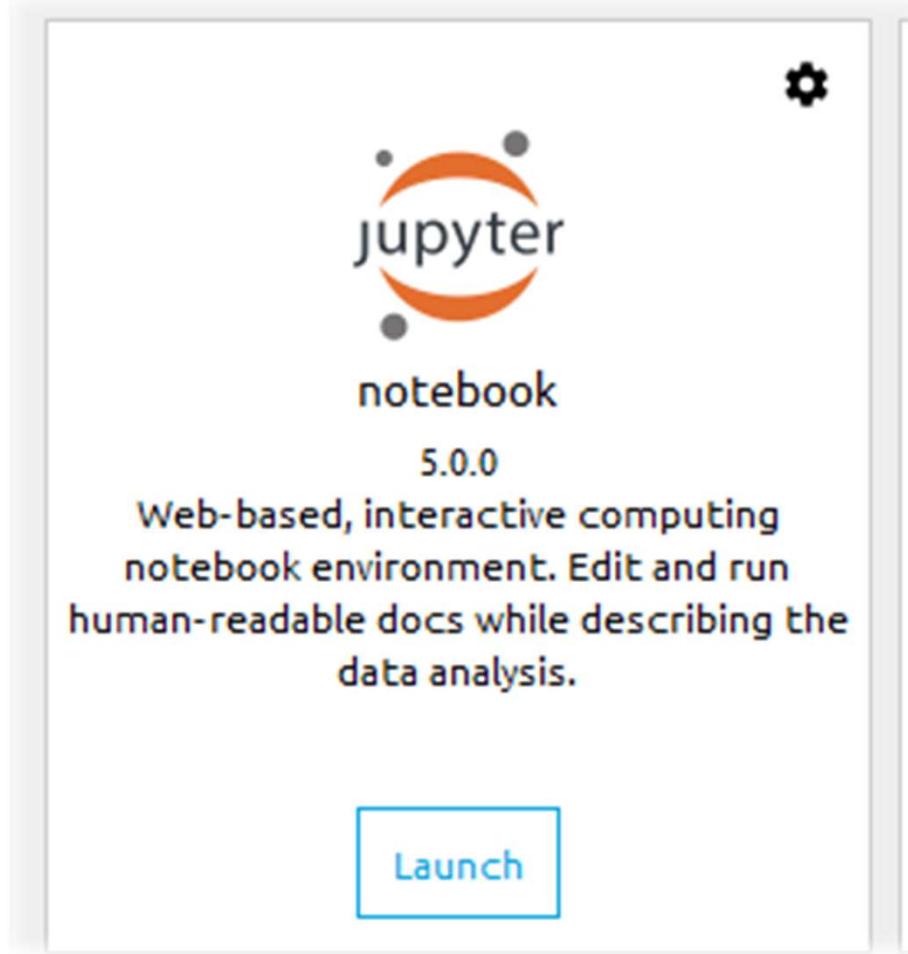
What is Jupyter Notebook?

The Jupyter Notebook is an ANACONDA TOOL and is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

Note: Jupyter Notebook runs on your browser.



JUPYTER NOTEBOOK IDE



JUPYTER NOTEBOOK IDE



JUPYTER DASHBOARD



The screenshot shows the Jupyter Dashboard running on a local host. The top bar includes standard OS X window controls and a title bar for 'localhost'. Below the title bar, the Jupyter logo is displayed. The main interface has a navigation bar with tabs: 'Files' (selected), 'Running', 'Clusters', and 'Conda'. A message 'Select items to perform actions on them.' is visible. On the left, a file tree shows a directory structure: a root folder containing a 'bi1' folder, which itself contains an ellipsis ('..') and a 'data' folder. On the right, there is a sidebar with a 'New' button and a dropdown menu. The dropdown menu lists several options: 'Text File', 'Folder', 'Terminal', 'Notebooks', 'Matlab', 'Python [conda root]' (which is highlighted with a green oval), and 'Python [default]'. The 'Python [conda root]' option is the active choice.



JUPYTER INTERACTIVE MODE



jupyter Untitled (autosaved)

File Edit View Insert Cell Kernel Help

In []:

Kernel menu open:

- Interrupt
- Restart
- Reconnect
- Change kernel ▾

Cell Toolbar: None

- Julia 0.3.1
- Python 2
- Python 3
- R



JUPYTER SCRIPT MODE



jupyter Untitled1 Last Checkpoint: 20 hours ago (unsaved changes) R O

File Edit View Insert Cell Kernel Help Cell Toolbar: None

```
In [22]: x <- rnorm(10)
y <- rnorm(10)
summary(lm(y~x))

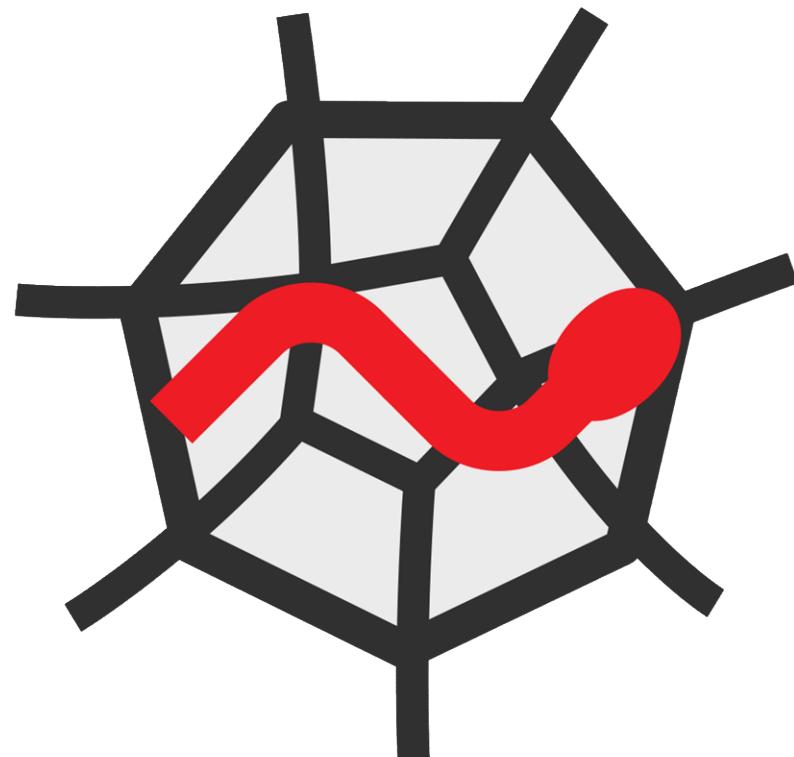
Out[22]:
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.4317 -0.8257 -0.1580  0.9709  1.4319 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 0.01308   0.34428   0.038   0.9706    
x           1.05533   0.44650   2.364   0.0457 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.088 on 8 degrees of freedom
Multiple R-squared:  0.4112,    Adjusted R-squared:  0.3376 
F-statistic: 5.586 on 1 and 8 DF,  p-value: 0.0457
```

SPYDER IDE





SPYDER

SPYDER IDE



The screenshot shows the Spyder IDE interface. At the top left is the Spyder logo (a red wavy line on a hexagonal grid). To the right is a gear icon. Below the logo, the word "spyder" is written in lowercase. Underneath it, the version "3.2.3" is displayed. A descriptive text block follows: "Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features". At the bottom center is a blue rectangular button with the word "Launch" in white.



SPYDER IDE



What is Spyder?

Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language



SPYDER IDE



Spyder (Python 2.7)

File Edit Search Source Run Debug Interpreters Tools View ?

Editor - D:\projects\python\scratch.py

scratch.py* X

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Dec 19 16:49:31 2015
4 """
5
```

Object inspector

Source Console Object plt.plot

plot

Definition : plot(*args, **kwargs)
Type : Function of matplotlib.pyplot module

Object inspector Variable explorer File explorer

Console

Python 1 X 00:48:11

>>>

Internal console Console History log

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 1 Column: 1 Memory: 30 %

This screenshot shows the Spyder IDE interface. The main window title is "Spyder (Python 2.7)". The menu bar includes File, Edit, Search, Source, Run, Debug, Interpreters, Tools, View, and ?. The toolbar has various icons for file operations like Open, Save, and Run. The left pane is the "Editor" showing a Python script named "scratch.py" with the following content:

```
1 # -*- coding: utf-8 -
2 """
3 Created on Sat Dec 19 16:49:31 2015
4 """
5
```

The "Object inspector" panel is open, focusing on the "plt.plot" function, displaying its definition and type. The "Console" panel at the bottom shows a single line starting with '>>>'. The status bar at the bottom provides system information like permissions, encoding, and memory usage.



PyScripter IDE



PyScripter IDE



What is PyScripter?

PyScripter is a free and open-source Python Integrated Development Environment (IDE) created with the ambition to become competitive in functionality with commercial Windows-based IDEs available for other languages



PyScripter IDE



The screenshot shows the PyScripter IDE interface. The main window displays a Python script named `module1`. The code is as follows:

```
#-----#
# Name:      Quadratic Formula
# Purpose:
#
# Author:     David Johns
#
# Created:   29/11/2011
# Copyright: (c) David Johns 2011
# Licence:   <your licence>
#-
#!/usr/bin/env python

def mymain():
    import math, time

    # assign variables a, b, and c from the quadratic function
    a = raw_input('What is the coefficient of the x^2 term, a?')
    b = raw_input('What is the coefficient of the x term, b?')
    c = raw_input('What is the constant, a?')

if __name__ == '__main__':
    mymain()
```

The PyScripter interface includes a toolbar at the top, a `File Explorer` sidebar on the left showing drives `Computer`, `OS (C:)`, and `DVD RW Drive (D:)`, and a `Python Interpreter` panel at the bottom showing a prompt:

```
>>>
```



Hello World Program

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



Demo on IDE

A screenshot of the IDLE Shell 3.11.2 interface. The title bar says "IDLE Shell 3.11.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays Python code and its output. The code is:

```
Python 3.11.2 (tags/v3.11.2:878ead1, AMD64) ] on win32
Type "help", "copyright", "credits" o
>>> print("hello World")
hello World
>>>
```

The text "hello World" is colored green, while the rest of the text is black.



What is Keyword or reserved word?



Keyword/Reserved Word

What is Keyword?

Keywords are also called as reserved words these are having special meaning in python language. The words are defined in the python interpreter hence these cant be used as programming identifiers.



Some Keywords of Python Language

and	assert
break	class
continue	def
del	elif
else	except
exec	finally
for	from



Some Keywords of Python Language

global	if
import	in
is	lambda
not	or
pass	print
raise	return
try	while
with	yield



What is an identifier?



IDENTIFIERS

What is an identifier?

A Python Identifier is a name given to a function, class, variable, module, or other objects that you'll be using in your Python program.

In short, it's a name appeared in the program.

For example: a, b, c

a b and c are the identifiers and

a b & c and , are the tokens



PYTHON NAMING CONVENTIONS



PYTHON NAMING CONVENTIONS

What are the python naming conventions?

- An identifier can be a combination of uppercase letters, lowercase letters, underscores, and digits (0-9).
- Hence, the following are valid identifiers: myClass, my_variable, var_1, and print_hello_world.



PYTHON NAMING CONVENTIONS

What are the python naming conventions?

- The first character must be letter.
- Special characters such as %, @, and \$ are not allowed within identifiers.
- An identifier should not begin with a number. Hence, 2variable is not valid, but variable2 is acceptable.

PYTHON NAMING CONVENTIONS



What are the python naming conventions?

- Python is a case-sensitive language and this behaviour extends to identifiers. Thus, Labour and labour are two distinct identifiers in Python.
- You cannot use Python keywords as identifiers.



PYTHON NAMING CONVENTIONS

What are the python naming conventions?

- You cannot use Python keywords as identifiers.
- You can use underscores to separate multiple words in your identifier.



PYTHON NAMING CONVENTIONS

SOME VALID IDENTIFIERS:

Myfile1

DATE9_7_8

y3m9d3

_xs

MYFILE

_FXd

SOME INVALID IDENTIFIERS:

MY-REC

28dre

break

elif

false

del



Things to Remember

- Python is a case-sensitive language. This means, Variable and variable are not the same.
- Always give the identifiers a name that makes sense. While `c = 10` is a valid name, writing `count = 10` would make more sense, and it would be easier to figure out what it represents when you look at your code after a long gap.
- Multiple words can be separated using an underscore, like `this_is_a_long_variable`.



Swap two numbers without using third variable

$x = 5.4$

$y = 10.3$

After swapping X and Y, we get :

$x = 10.3$

$y = 5.4$



Swap: Using simple built-in method

`x = 5.4`

`y = 10.3`

`# Swap Code`

`x, y = y, x`



Swap: Using Addition and Subtraction Operators

$x = 5.4$

$y = 10.3$

Swap Code

$x = x + y$

$y = x - y$

$x = x - y$



Swap: Using Division and Multiplication Operators

$x = 5.4$

$y = 10.3$

```
# Swap code
```

```
x = x * y
```

```
y = x / y
```

```
x = x / y
```



Reading Input from user

input ():

- This function first takes the input from the user and converts it into a string.
- The type of the returned object always will be <type ‘str’>. It does not evaluate the expression it just returns the complete statement as String.
 - For example, Python provides a built-in function called `input` which takes the input from the user. When the `input` function is called it stops the program and waits for the user’s input. When the user presses enter, the program resumes and returns what the user typed.

Syntax:

```
inp = input('STATEMENT')
```



Example

```
num = input ("Enter number :")
print(num)
name1 = input("Enter name : ")
print(name1)
```

```
# Printing type of input value
print ("type of number", type(num))
print ("type of name", type(name1))
```



Python Datatypes

- Everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.
- Standard Built-in types
 - **Numeric**
 - **Sequence Type**
 - **Boolean**
 - **Set**
 - **Dictionary**



Python Datatypes

Text Type: `str`

Numeric Types: `int`, `float`, `complex`

Sequence Types: `list`, `tuple`, `range`

Mapping Type: `dict`

Set Types: `set`, `frozenset`

Boolean Type: `bool`

Binary Types: `bytes`, `bytearray`, `memoryview`

None Type: `NoneType`

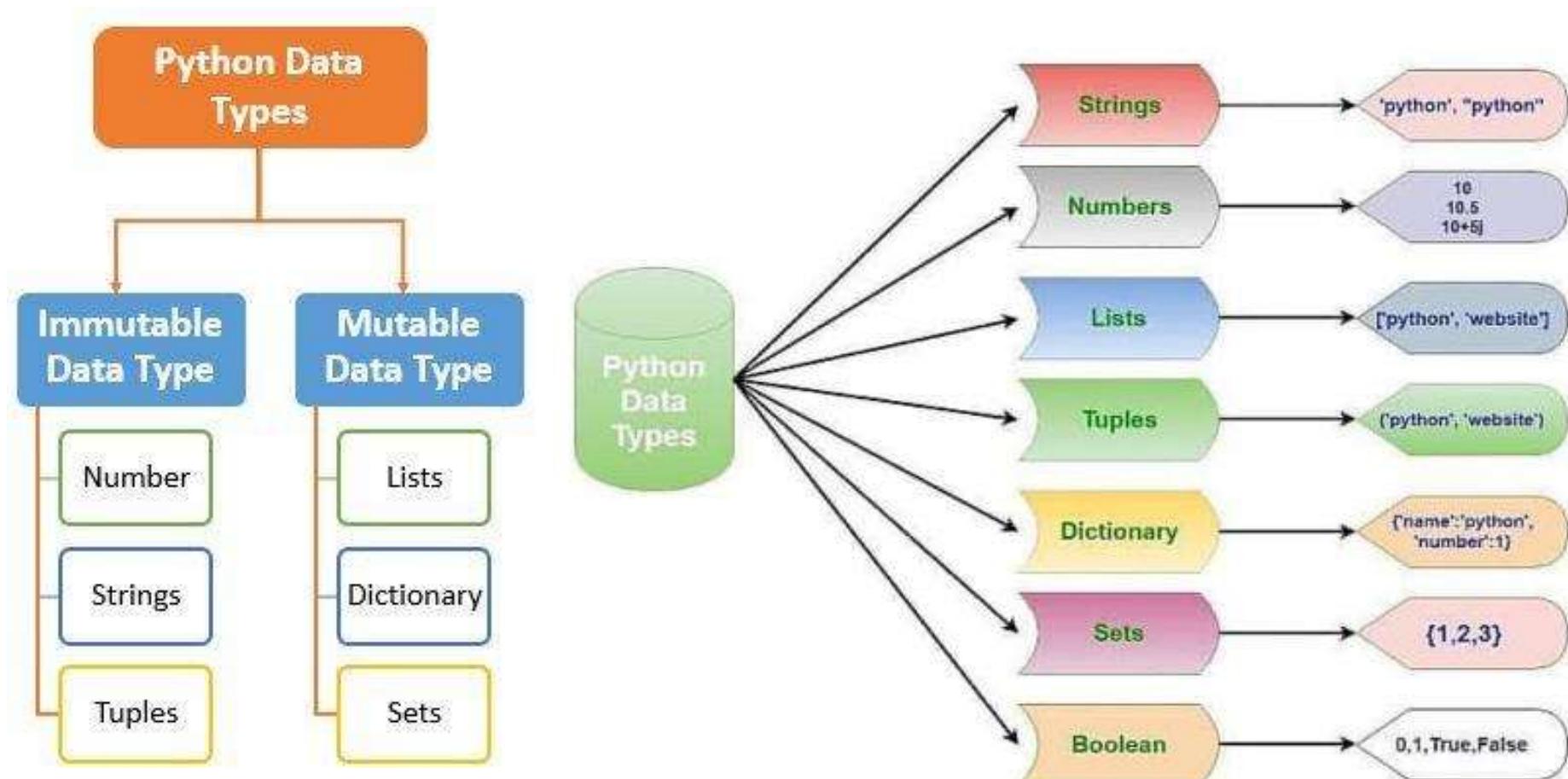


Python Datatypes

Name	Type	Description
Integers	int	Whole numbers, such as: 3 300 200
Floating point	float	Numbers with a decimal point: 2.3 4.6 100.0
Strings	str	Ordered sequence of characters: "hello" 'Sammy' "2000" "楽しい"
Lists	list	Ordered sequence of objects: [10,"hello",200.3]
Dictionaries	dict	Unordered Key:Value pairs: {"mykey": "value", "name": "Frankie"}
Tuples	tup	Ordered immutable sequence of objects: (10,"hello",200.3)
Sets	set	Unordered collection of unique objects: {"a","b"}
Booleans	bool	Logical value indicating True or False



Python Datatypes





Python Datatypes

- A **mutable** object can be changed after it is created, and an **immutable** object can't.
- mutable objects by changing the element the `id()` value will not change , whereas for immutable objects `id()` value will change.

Class	Description	Immutable?
<code>bool</code>	Boolean value	✓
<code>int</code>	integer (arbitrary magnitude)	✓
<code>float</code>	floating-point number	✓
<code>list</code>	mutable sequence of objects	
<code>tuple</code>	immutable sequence of objects	✓
<code>str</code>	character string	✓
<code>set</code>	unordered set of distinct objects	
<code>frozenset</code>	immutable form of set class	✓
<code>dict</code>	associative mapping (aka dictionary)	



Mutable objects:
list, dict, set, byte array

```
>>> list1 = [1,2,3,4]
>>> id(list1)
23321928
>>> list1[1] = 10
>>> id(list1)
23321928
```

Immutable objects:
int, float, complex, string, tuple, frozen set
[note: immutable version of set], bytes

```
>>> # Example 1
>>> var1 = 10
>>> id(var1)
1852024896
>>> var1 = 20
>>> id(var1)
1852025056
```

```
>>> string1 = "abc"
>>> id(string1)
22712096
>>> string1 = string1 + "def"
>>> id(string1)
23392064
```



Declaring and using Numeric data types

- Integer
- Float
- Complex
- String



Integer

- Integers are one of the Python data types. An integer is a whole number, negative, positive or zero.
- In Python, integer variables are defined by assigning a whole number to a variable. Python's `type()` function can be used to determine the data type of a variable.

```
>>> a = 5
>>> type(a)
<class 'int'>
```



Float

- Floating point numbers or floats are another Python data type.
- Floats are decimals, positive, negative and zero.
- Floats can also be represented by numbers in scientific notation which contain exponents.
- Both a lower case e or an upper case E can be used to define floats in scientific notation.
- In Python, a float can be defined using a decimal point . when a variable is assigned.

```
>>> c = 6.2
>>> type(c)
<class 'float'>
>>> d = -0.03
>>> type(d)
<class 'float'>
>>> Na = 6.02e23
>>> Na
6.02e+23
>>> type(Na)
<class 'float'>
```



Complex

- Another useful numeric data type for problem solvers is the complex number data type.
- A complex number is defined in Python using a real component + an imaginary component j.
- The letter j must be used to denote the imaginary component.
- Using the letter i to define a complex number returns an error in Python.

```
>>> comp = 4 + 2j  
>>> type(comp)  
<class 'complex'>
```



String

- Numbers and decimals can be defined as strings too. If a decimal number is defined using quotes '', the number is saved as a string rather than as a float.
- Integers defined using quotes become strings as well.

```
>>> num = '5.2'  
>>> type(num)  
<class 'str'>
```

```
>>> num = '2'  
>>> type(num)  
<class 'str'>
```



Thank You