

An Introduction to Python

Professor: Dr Abas Bahrololom

Milad Abolhassani

Asgar Jangah

- ▶ Introduction
- ▶ Developer
- ▶ Python naming
- ▶ Why Python
- ▶ Supported platforms
- ▶ Some charts
- ▶ Comparing Syntaxes
- ▶ Python vs Matlab
- ▶ Where Python is being used?
- ▶ Python 2 vs Python 3
- ▶ Installation guide
- ▶ IDE
- ▶ SheBang
- ▶ Data Types and operators
- ▶ Statements
- ▶ OOP
- ▶ Introducing to some libraries
- ▶ What's next

Introduction

3

Python is a **general purpose** programming language which is commonly used for both **standalone** programs and **scripting** applications in a wide variety of domains, and is generally considered to be one of the most widely used programming languages in the world.

- ▶ Programming and Scripting
- ▶ General-purpose (Variety of domains)
- ▶ Free software
- ▶ Object-oriented

Introduction

- ▶ Scripting languages
 - ▶ All scripting languages are programming languages
 - ▶ Scripting languages do not require the compilation step and are rather interpreted
- ▶ Applications (sample scripts [GH](#), [AU](#)):
 - ▶ To automate certain tasks in a program
 - ▶ Text processing and extracting information from a data set
 - ▶ Standalone?
- ▶ Python also can be compiled
 - ▶ bytecode [c]

Developer

5



- ▶ Python's origins lie way back in distant December 1989.
Created by Guido van Rossum.
- ▶ Started as a 'Hobby' programming project that would keep him occupied during the week around Christmas as his office was closed.

History of Python naming

6



- ▶ Often people assume that the name Python was written after a snake. Even the logo of Python programming language depicts the picture of two snakes, blue and yellow. But, the story behind the naming is somewhat different.
- ▶ Back in the 1970s, there was a popular BBC comedy TV show called Monty Python's Fly Circus and Van Rossum happened to be the big fan of that show. So when Python was developed, Rossum named the project 'Python'.

Why Python?

7

- ▶ Easy to learn
- ▶ Elegant Syntax
- ▶ Easy to run
- ▶ Healthy, Active and Supportive Community
- ▶ Amazing Libraries
- ▶ Awesome Package Manager
- ▶ Free Software (LIB, Usage)
 - ▶ Copyright
- ▶ General Purpose
 - ▶ Useful for a really broad range of programming tasks from little shell scripts to enterprise web applications to scientific uses.

Why Python?

8

- ▶ Python is the future of AI and Machine learning
 - ▶ Scikit-learn (Machine learning)
 - ▶ Keras (Neural network)
 - ▶ TensorFlow (ML, NN)
- ▶ Fast, Well designed, Portable, Scalable
 - ▶ These are the most important factors for AI applications
- ▶ Most popular and widely used deep-learning frameworks are implemented in Python
- ▶ Lots of data structures, interpretive run-time
 - ▶ Prototypes algorithms quickly

Why Python?

9

- ▶ Spend more time on the algorithms instead of learning the tool and its interface.
- This page attempts to collect information and links pertaining to the practice of AI and Machine Learning in python:
 - <https://wiki.python.org/moin/PythonForArtificialIntelligence>
- Role of Python in Artificial Intelligence:
 - <http://www.cuelogic.com/blog/role-of-python-in-artificial-intelligence/>

Supported Platforms

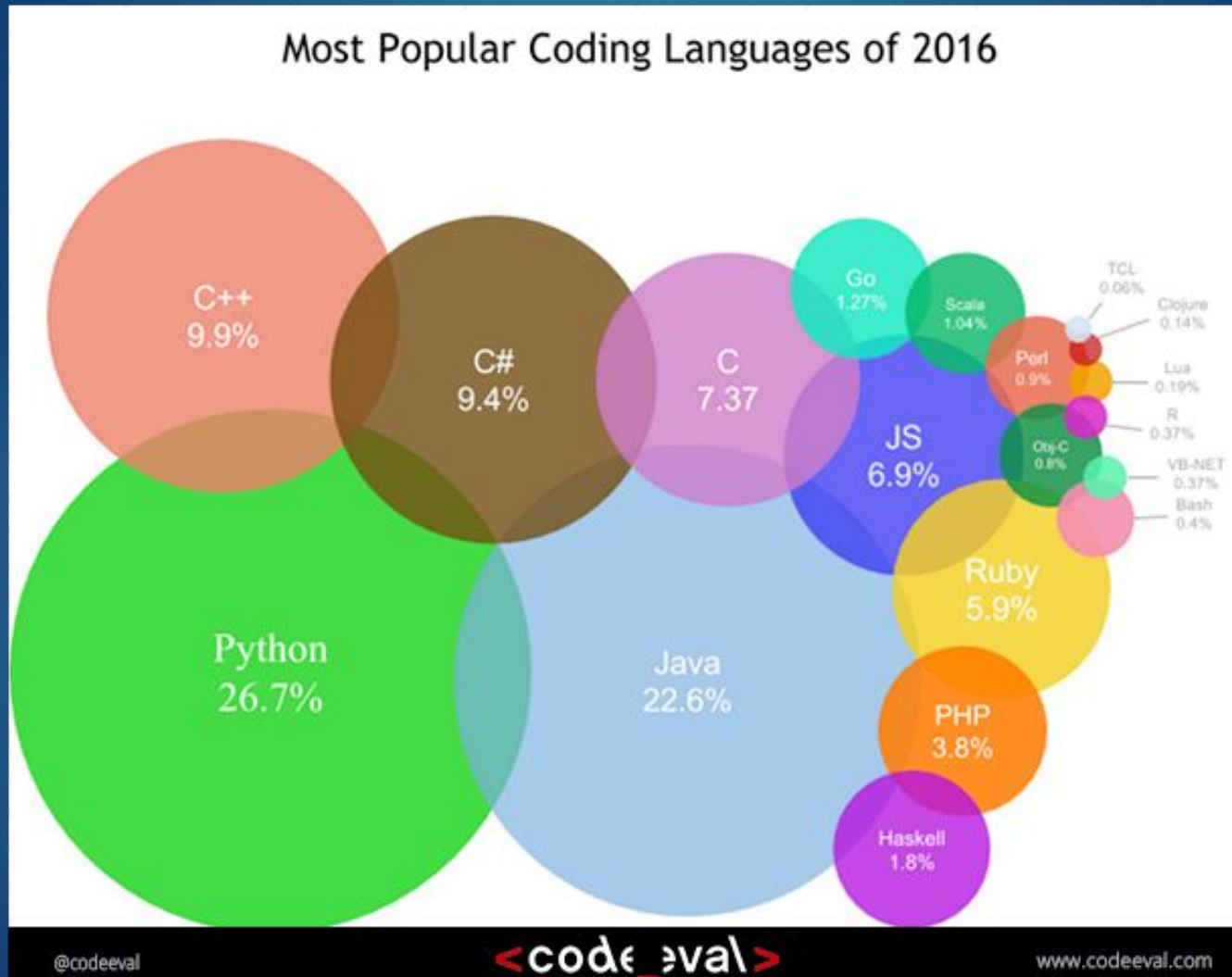
10

- ▶ Python programs run on most platforms
 - ▶ Linux
 - ▶ Unix
 - ▶ Mac
 - ▶ Windows
 - ▶ etc.



Popular Languages of 2016

11

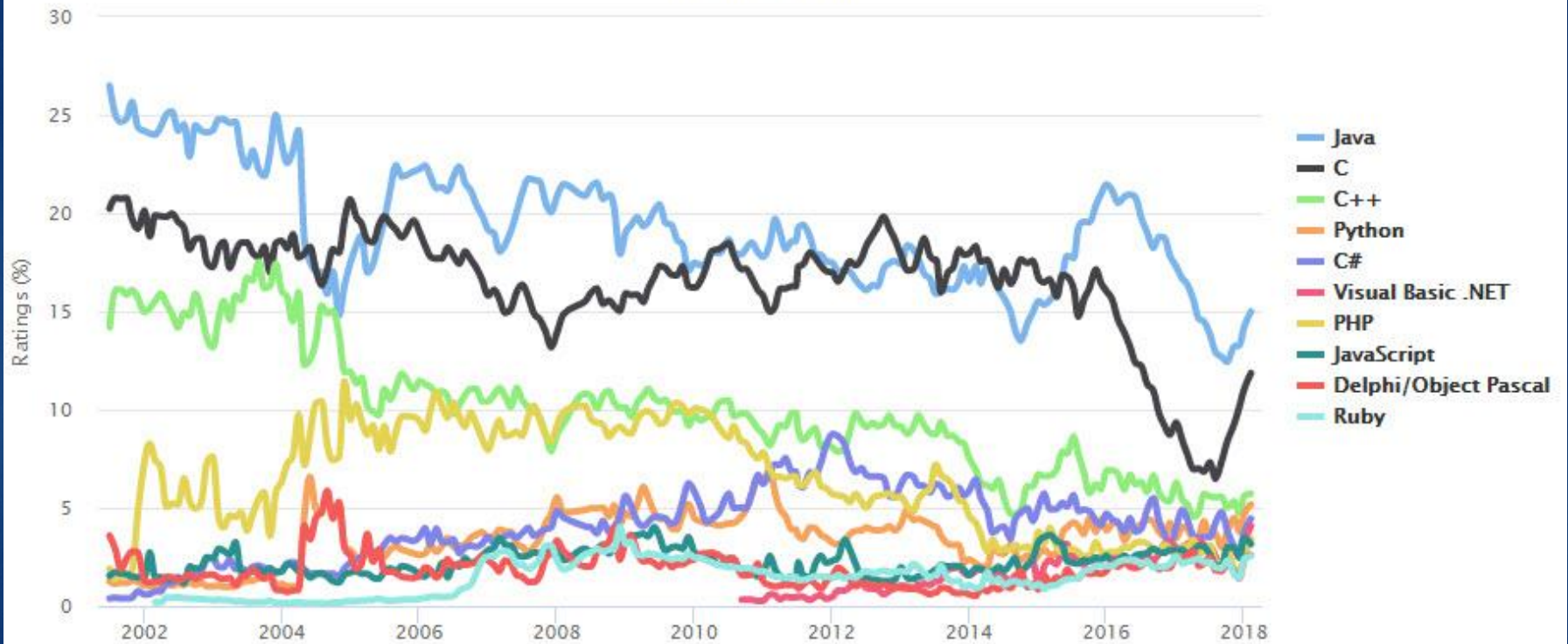


TIOBE Index

12

TIOBE Programming Community Index

Source: www.tiobe.com



TIOBE Index

13

Feb 2018	Feb 2017	Change	Programming Language	Ratings	Change
1	1		Java	14.988%	-1.69%
2	2		C	11.857%	+3.41%
3	3		C++	5.726%	+0.30%
4	5	▲	Python	5.168%	+1.12%
5	4	▼	C#	4.453%	-0.45%
17	10	▼	Perl	1.759%	-0.41%
18	14	▼	Go	1.417%	-0.69%
19	17	▼	MATLAB	1.228%	-0.49%
20	19	▼	Objective-C	1.130%	-0.41%

Compare Syntax

14

C++

```
// my first program in C++  
#include <iostream>  
int main()  
{  
std::cout << "Hello World!";  
return 0;  
}
```

C

```
#include <stdio.h>  
int main()  
{  
/* my first program in C */  
printf("Hello, World! \n");  
return 0;  
}
```

java

```
public class MyFirstJavaProgram {  
    /* This is my first java program. * This will print  
    'Hello World' as the output */  
    public static void main(String []args) {  
        System.out.println("Hello World");  
    }  
}
```


Compare Syntax

15

C#

```
using System;
namespace HelloWorldApplication
{
    class HelloWorld
    {
        static void Main(string[] args)
        {
            /* my first program in C# */
            Console.WriteLine("Hello World");
            Console.ReadKey();
        }
    }
}
```

V
S

Python

```
Print ("Hello world!")
```

Python vs Matlab

16

- ▶ More compact and readable than Matlab code.
 - ▶ there is no **end**
 - ▶ like all languages Python uses [] for indexing and () for calls
 - ▶ Matlab uses () for both. hard to understand.
 - ▶ Python NumPy's array methods min, max, mean, etc. operate by default on all dimensions of an array.
 - ▶ Matlab = mean(mean(mean(mean(x))));
 - ▶ Python = mean(x)
 - ▶ Python does not arbitrarily restrict the use of literals in expressions [c]
- ▶ Python data structures are superior to Matlab data structures.

Python vs Matlab

17

- ▶ Python follows standards
 - ▶ Like almost every programming language other than Matlab, Python uses zero-based indexing
 - ▶ Programmers who must implement published algorithms, convert code from one language to another
- ▶ Python offers excellent support for dictionaries (hashes) [c]
 - ▶ Matlab provides support for dictionaries (these are called hash maps in Matlab terminology), but imposes the pointless restriction that all keys have the **same type**
- ▶ OOP in Python is simple and elegant
- ▶ Python is free and open.
 - http://phillipmfeldman.org/Python/Advantages_of_Python_Over_Matlab.html

Where Python is being used?

18

- ▶ Applications
- ▶ Web Applications
- ▶ Package management
- ▶ Video Games
- ▶ Web Frameworks
- ▶ Graphics Frameworks
- ▶ UI Frameworks
- ▶ Scientific packages
- ▶ Mathematical Libraries
- ▶ Numerical Libraries
- ▶ Development Packages
- ▶ Different Implementations



Where Python is being used?

19

- ▶ UI Frameworks

- ▶ Tkinter
- ▶ Kivy
- ▶ PyQT
- ▶ PyGTK

- ▶ Video Games

- ▶ Civilization IV
- ▶ Battlefield 2

- ▶ Web Frameworks

- ▶ Django
- ▶ Flask
- ▶ Tornado

Where Python is being used?

20

- ▶ Applications

- ▶ DropBox
- ▶ BitTorrent
- ▶ Bazaar
- ▶ OpenStack

- ▶ Web services

- ▶ Reddit
- ▶ Youtube

- ▶ Web Applications

- ▶ Mailman (mail list)
- ▶ Roundup (bug tracking system)

Python 2

- ▶ Python 2.0 - October 16, 2000
- ▶ Python 2.1 - April 17, 2001
- ▶ Python 2.2 - December 21, 2001
- ▶ Python 2.3 - July 29, 2003
- ▶ Python 2.4 - November 30, 2004
- ▶ Python 2.5 - September 19, 2006
- ▶ Python 2.6 - October 1, 2008
- ▶ Python 2.7 - July 3, 2010

Python 3

- ▶ Python 3.0 - December 3, 2008
- ▶ Python 3.1 - June 27, 2009
- ▶ Python 3.2 - February 20, 2011
- ▶ Python 3.3 - September 29, 2012
- ▶ Python 3.4 - March 16, 2014
- ▶ Python 3.5 - September 13, 2015
- ▶ Python 3.6 - May 30, 2016

PYTHON 2 VS PYTHON 3

22

- ▶ Python 2.x is legacy, Python 3.x is the present and future of the language
- ▶ Clean up Python 2.x properly, with less regard for backwards compatibility



PYTHON 2 VS PYTHON 3

23

- ▶ Use Python 2 if:
 - ▶ An environment you don't control
 - ▶ If you want to use a specific third party package
- ▶ The real difference for someone new to python [c]:
 - ▶ Print
 - ▶ Division
 - ▶ Input
 - ▶ Amount of libraries (Not anymore)
- ▶ Different modules and functions

How to download and install

24

- ▶ The Python download requires about 30 Mb of disk space And When installed, Python requires about an additional 90 Mb of disk space.
- <https://www.python.org/downloads>



How to download and install

25

- ▶ The file named **python-3.6.4.exe** should start downloading into your standard download folder. This file is about 30 Mb so it might take a while to download fully if you are on a slow internet connection.
- ▶ After download , go to the location of file.

Installation

26

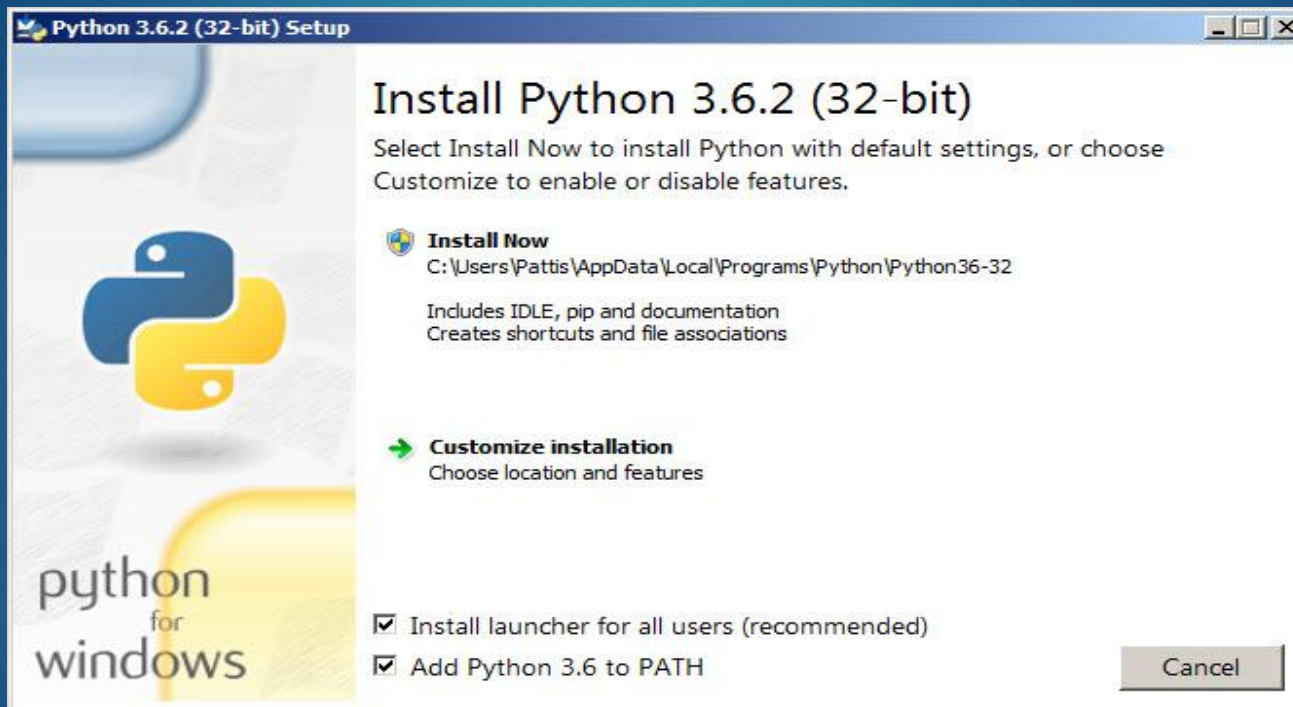
- ▶ Double-click the icon labeling the file **python-3.6.4.exe**.
- ▶ An **Open File - Security Warning** pop-up window will appear.



Installation

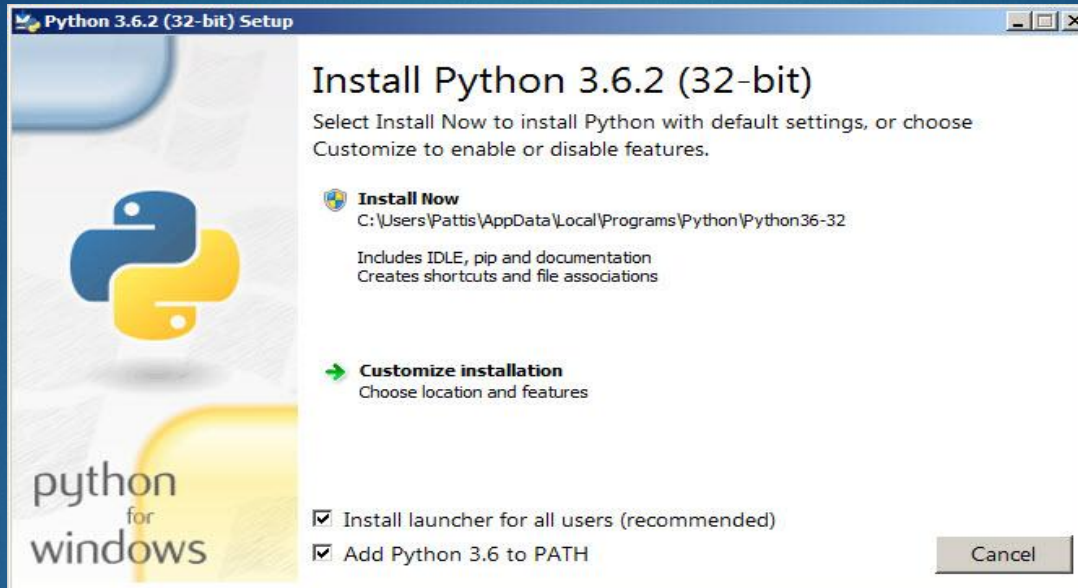
27

- ▶ Click **Run**.
- ▶ A **Python 3.6.4 (32-bit) Setup** pop-up window will appear.



Installation

28

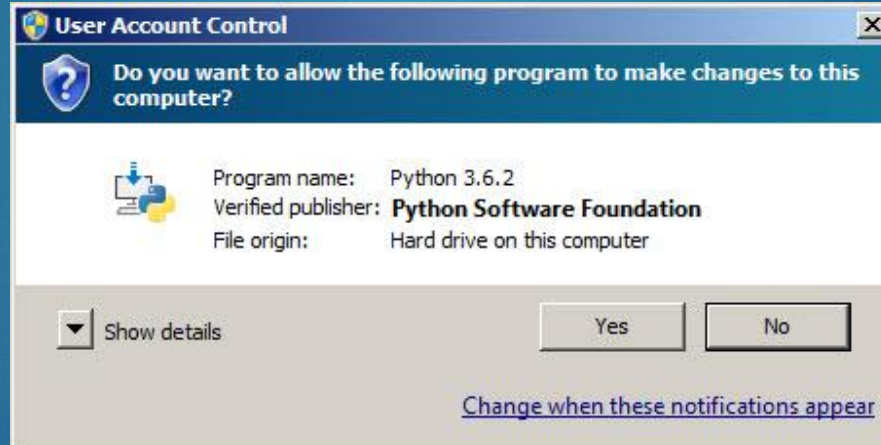


- ▶ Ensure that the **Install launcher for all users (recommended)** and the **Add Python 3.6 to PATH** checkboxes at the bottom are checked.
- ▶ If the Python Installer finds an earlier version of Python installed on your computer, the **Install Now** message will instead appear as **Upgrade Now** (and the checkboxes will not appear).

Installation

29

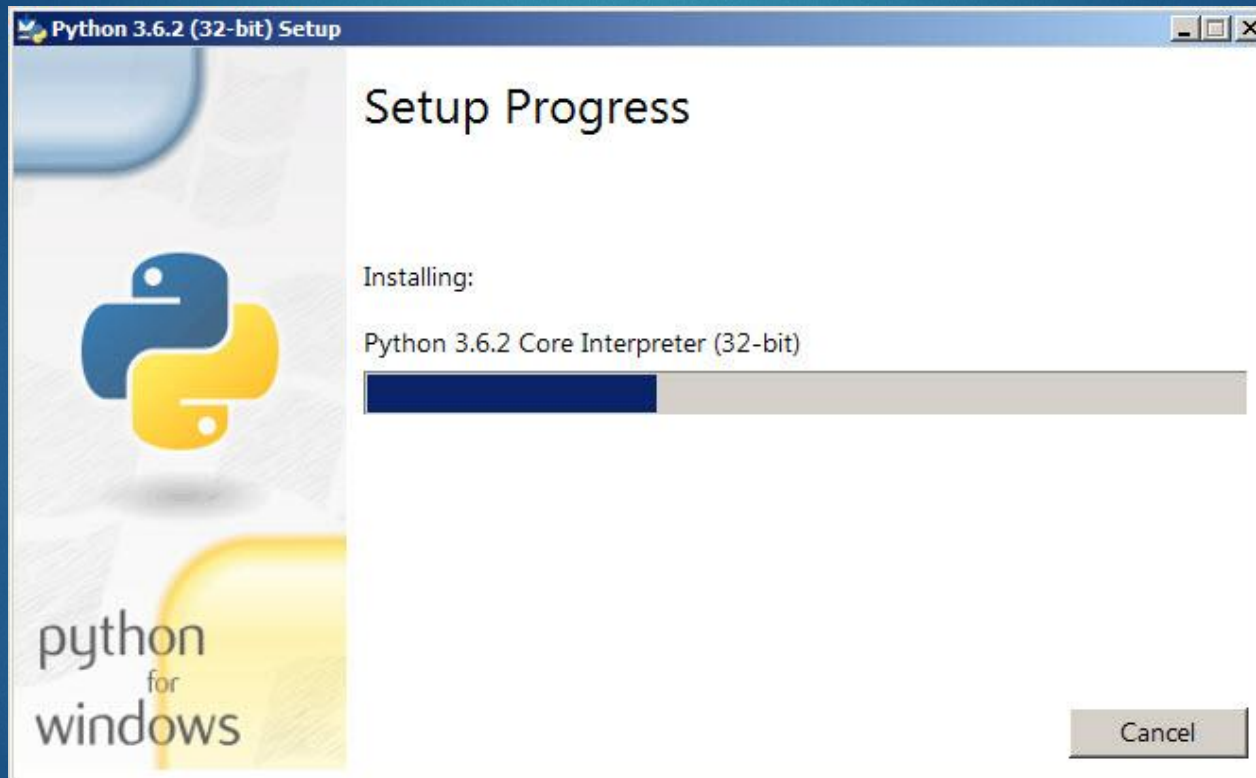
- ▶ Highlight the **Install Now** (or **Upgrade Now**) message, and then click it.
- ▶ A **User Account Control** pop-up window will appear, posing the question **Do you want to allow the following program to make changes to this computer?**
- ▶ Click the **Yes** button.



Installation

30

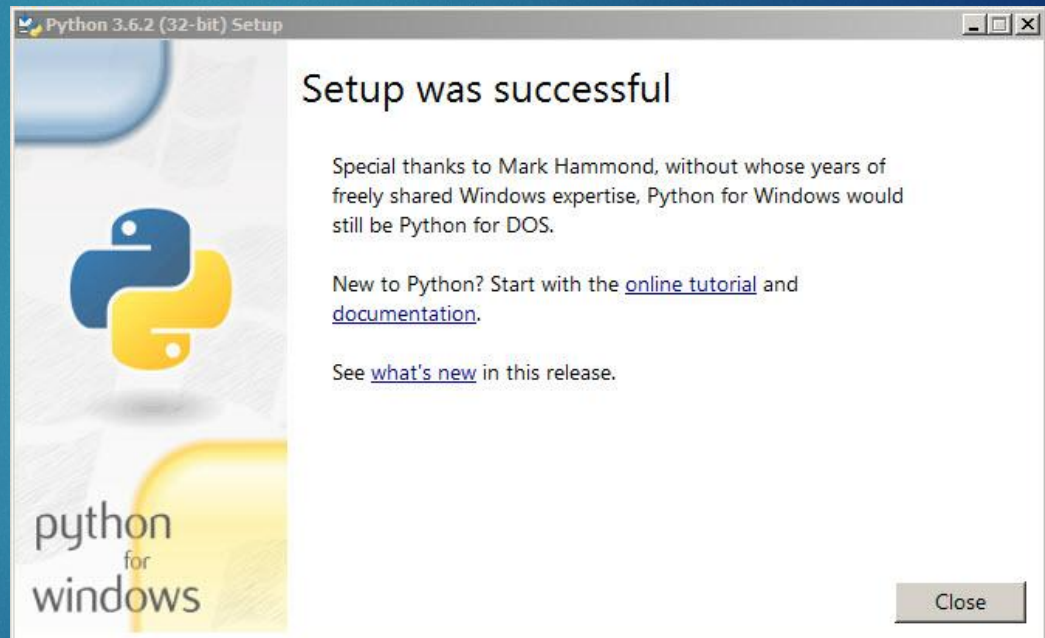
- ▶ A new **Python 3.6.4 (32-bit) Setup** pop-up window will appear with a **Setup Progress** message and a progress bar.



Installation

31

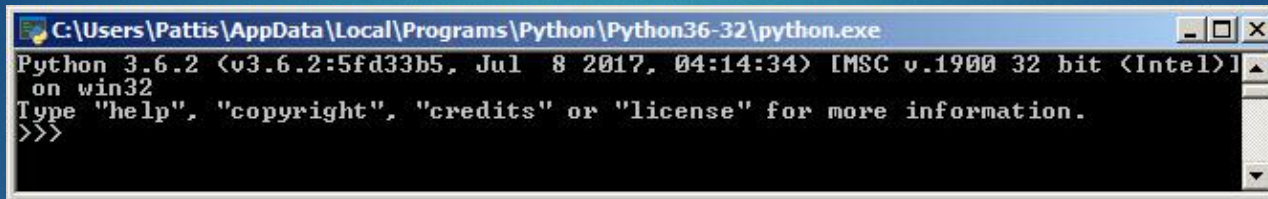
- ▶ During installation, it will show the various components it is installing and move the progress bar towards completion. Soon, a new **Python 3.6.4 (32-bit) Setup** pop-up window will appear with a **Setup was successfully** message.
- ▶ Click the **Close** button.



Verifying

32

- ▶ Navigate to the directory **C:\Users\Pattis\AppData\Local\Programs\Python\Python36-32** (or to whatever directory Python was installed: see the pop-up window for Installing step 3).
- ▶ Double-click the icon/file **python.exe**. The following pop-up window will appear.



```
C:\Users\Pattis\AppData\Local\Programs\Python\Python36-32\python.exe
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

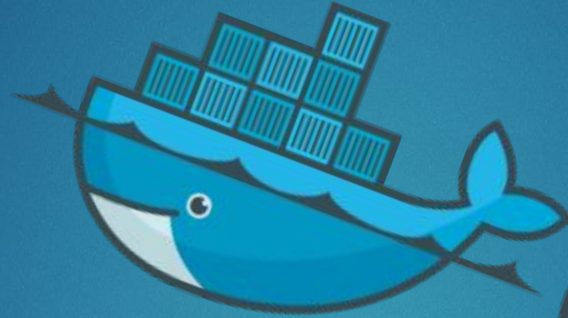
A pop-up window with the title **C:\Users\Pattis\AppData\Local\Programs\Python\Python36-32** appears, and inside the window; on the first line is the text **Python 3.6.4 ...** (notice that it should also say 32 bit). Inside the window, at the bottom left, is the prompt **>>>**: type **exit()** to this prompt and press enter to terminate Python.

You should keep the file **python-3.6.4.exe** somewhere on your computer in case you need to reinstall Python (not likely necessary).

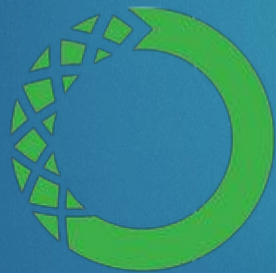
Other solutions

33

- ▶ Docker
- ▶ Anaconda



docker



ANACONDA®

- ▶ Anaconda
 - ▶ Anaconda is an open source distribution of the Python and R programming languages for large-scale data processing and scientific computing, that aims to simplify package management and deployment.
 - ▶ Package versions are managed by the package management system conda.
- <https://www.anaconda.com/download>

Anaconda

35

Anaconda Navigator

File Help

ANACONDA NAVIGATOR [Sign in to Anaconda Cloud](#)

[Home](#)

[Environments](#)

[Projects \(beta\)](#)

[Learning](#)

[Community](#)








[Documentation](#)

[Developer Blog](#)

[Feedback](#)

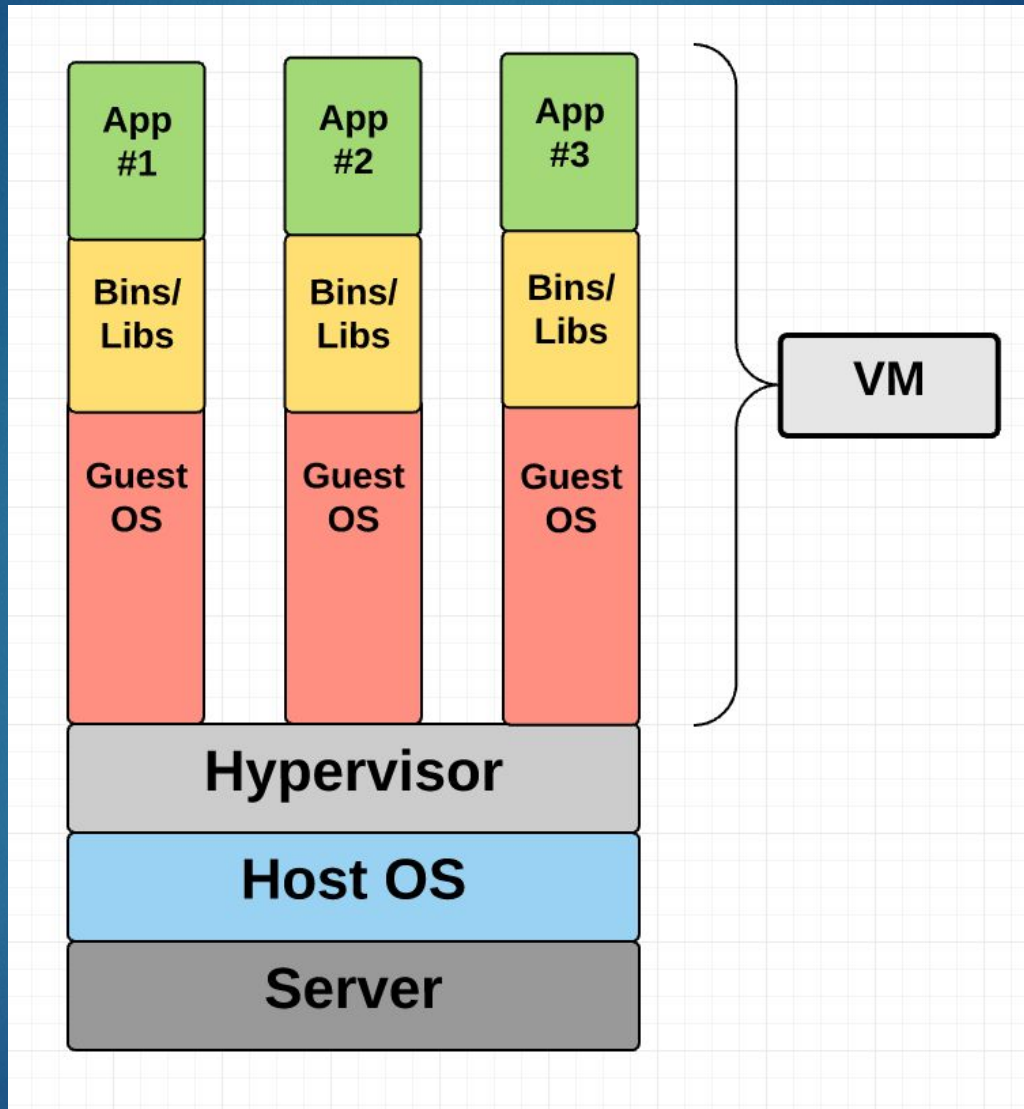
[Twitter](#) [YouTube](#) [GitHub](#)

Applications on Channels [Refresh](#)

 jupyterlab 0.27.0 An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. Launch	 jupyter notebook 5.0.0 Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. Launch	 qtconsole 4.3.1 PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. Launch	 spyder 3.2.4 Scientific PYTHON Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features Launch
 glueviz 0.10.4 Multidimensional data visualization across files. Explore relationships within and among related datasets. Launch	 orange3 3.4.1 Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox. Launch	 rstudio 1.1.383 A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks. Launch	

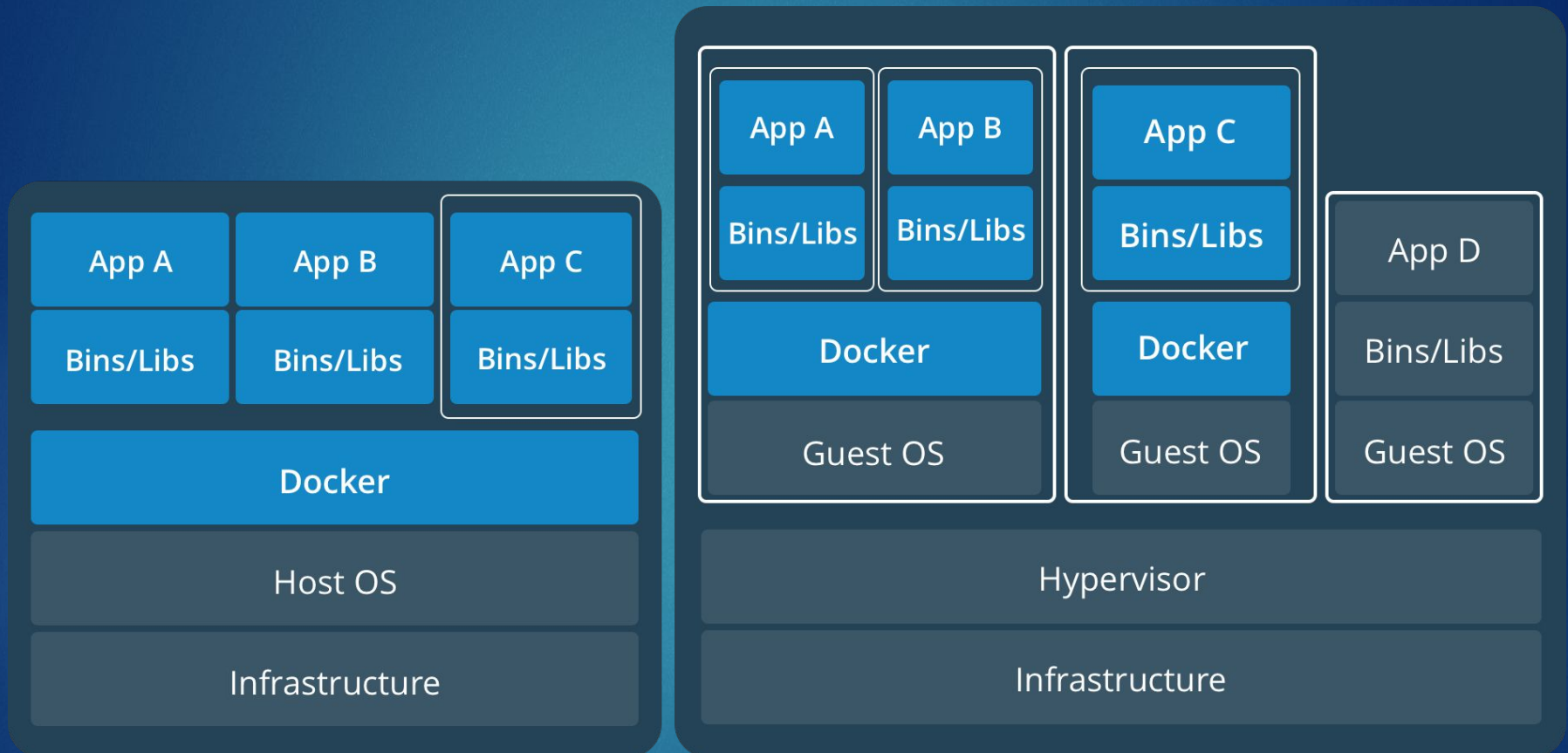
Virtualization [1]

36



Docker

37



<https://djangostars.com/blog/what-is-docker-and-how-to-use-it-with-python/>

What are Containers?

- ▶ IDE is an Integrated Development Environment
 - ▶ Multi-window text editor
 - ▶ Syntax highlighting
 - ▶ Auto completion
 - ▶ Smart indenting
 - ▶ Debugger

- ▶ Some IDE for Python:
 - ▶ eclipse (PyDev)
 - ▶ PyCharm
 - ▶ Visual Studio 17
 - ▶ Atom (Editor)
 - ▶ VIM (More than of an editor)
 - ▶ Jupyter Notebook (Not an IDE)

```

01.1_setup_and_introduction.ipynb x Titanic_Kaggle.ipynb x
Code ▼

In [5]: # Get Kaggle Titanic Datasets
t_train = pd.read_csv('/Users/williamliu/Dropbox/NYC-DAT-08/Homework_8/input/titanic_train.csv')
t_test = pd.read_csv('/Users/williamliu/Dropbox/NYC-DAT-08/Homework_8/input/titanic_test.csv')

In [6]: print t_train.head()

   PassengerId  Survived  Pclass    Name  Sex  Age  SibSp  Parch    Ticket
Fare Cabin Embarked
0         1         0       3    Braund, Mr. Owen Harris   male  22    1    0    A/5 21171  7.2500  S
1         2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38    1    0    PC 17566  53.0000  S
599       3         1       3    Heikkinen, Miss. Laina   female  26    0    0  STON/O2. 3101282  51.0000  S
7.9250  NaN
3         4         1       1  Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35    1    0    113803  53.1000  S
53.1000  C123
4         5         0       3    Allen, Mr. William Henry   male  35    0    0    373450  8.0500  S
00      NaN    S

In [691]: t_train['BoolSex'] = [1 if field=='male' else 0 for field in t_train.Sex]
t_test['BoolSex'] = [1 if field=='male' else 0 for field in t_test.Sex]
t_

v t_test
v t_test
In [692] v t_train
v tree_model
v print_function
m __import__ (name, globals, locals, fromlist, level) __future__ __builtin__
Press ^ to choose the selected (or first) suggestion and insert a dot afterwards >>> π

```


Visual Studio

41

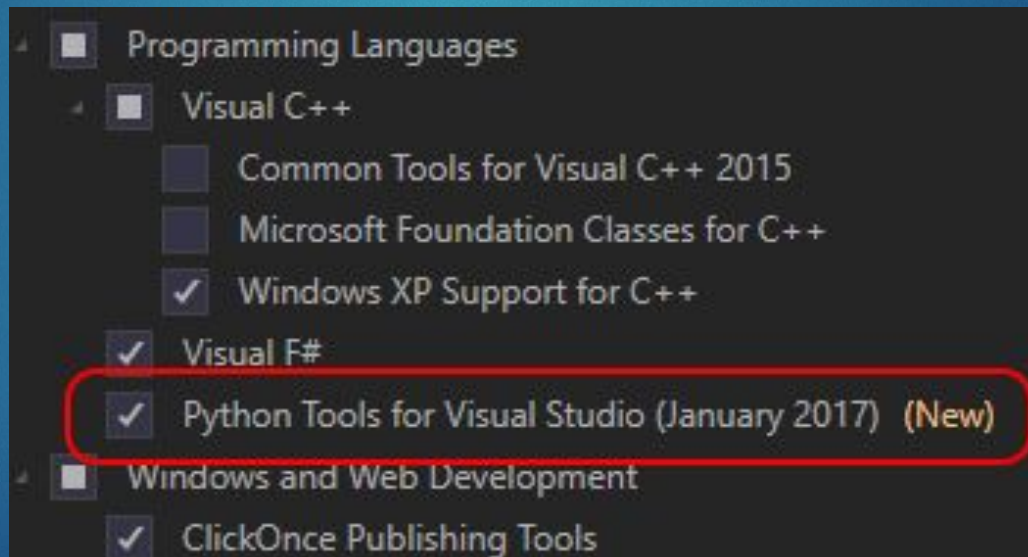


Python development

Editing, debugging, interactive development and source control for Python.

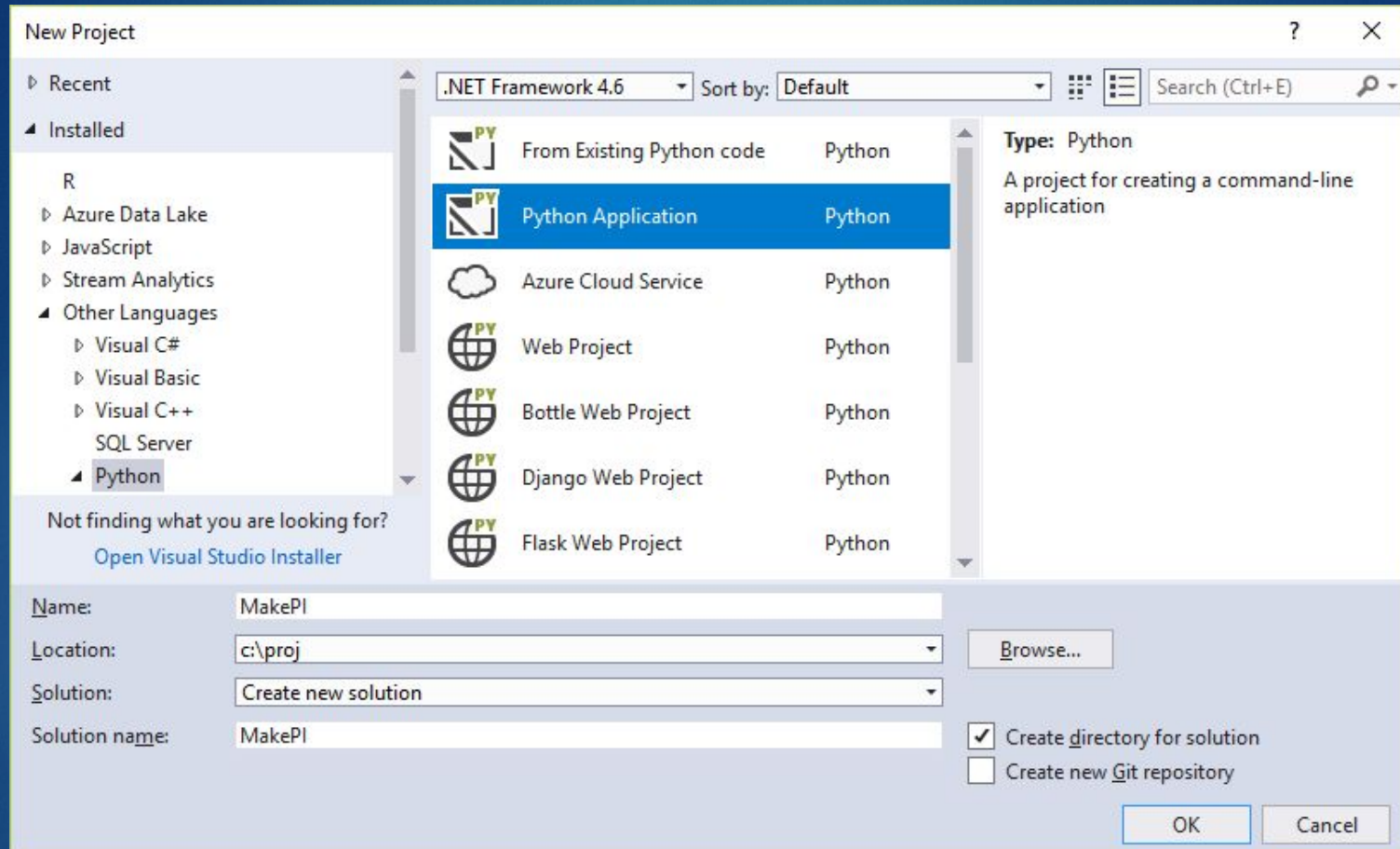


Control Panel > Programs and Features, selecting Microsoft Visual Studio 2015 and then Change.Modify - Programming Languages > Python Tools for Visual Studio and then Next.



Visual Studio

42



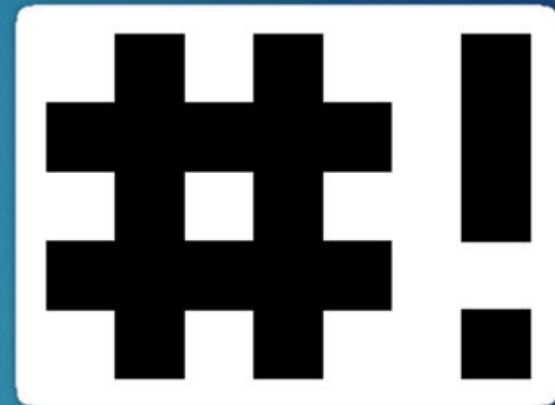
#!

43

`#!/usr/bin/env python`

`#!/usr/bin/python`

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



Naming rules [c]

44

- ▶ User-defined names **start** with a letter or underscore (`_`), followed by any number of letters, digits, or underscores.
- ▶ User-defined names cannot be the same as any Python **reserved word**
- ▶ User-defined names and reserved words are always **case sensitive**

Reserved words

45

None	continue	for	lambda	return
False	class	finally	is	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	break
except	in	raise	-	

Python is a dynamically typed language

[c]

46

- ▶ A language is dynamically typed if the type is associated with run-time values [c]
- ▶ This means that you as a programmer can write a little quicker because you do not have to specify types every time
 - ▶ Perl, Python, Ruby, PHP, JavaScript

Data Types

47

- ▶ Numbers
- ▶ Strings
- ▶ Lists
- ▶ Dictionaries
- ▶ Tuples
- ▶ Files
- ▶ Sets

Data types (Numbers)_[c]

48

Numbers are *immutable* (unchangeable) values, supporting numeric operations.

- ▶ Integers
 - ▶ 1234, -24, +42, 0
- ▶ Float
 - ▶ 1.23, 3.14e-10
- ▶ Octal, hex, and binary
 - ▶ 0o177, 0x9ff, 0b1111
- ▶ Complex numbers.
 - ▶ 3+4j, 3.0+4.0j, 3J

Data types (Numbers)

49

- ▶ `int(9.1)`
- ▶ `int('-9')`
- ▶ `int('0b1111', 0)`
- ▶ `float(9)`
- ▶ `float('1e2')`
- ▶ `float('-.1')`

Math Operators [c]

50

Operator	Description
$X + Y$	Add
$X - Y$	subtract
$X * Y$	Multiply
X / Y	divide
$X // Y$	floor
$X \% Y$	remainder
$X ** Y$	X to the power Y

Data types (Strings)

51

The normal str string object is an *immutable* (unchangeable) *sequence* of characters accessed by *offset* (position).

- ▶ 'Python's', "Python's"
 - ▶ Single and double quotes work the same, and each can embed unescaped quotes of the other kind.
- ▶ """This is a multiline block"""
 - ▶ Triple-quoted blocks collect multiple lines of text into a single string, with end-of-line markers (`\n`) inserted between the original quoted lines.

Data types (List) [c]

52

Lists are *mutable* (changeable) sequences of object references accessed by *offset*.

- ▶ `[]`
 - ▶ An empty list.
- ▶ `[0, 1, 2, 3]`
 - ▶ A four-item list: indexes 0 through 3.
- ▶ `L = ['spam', [42, 3.1415], 1.23, {}]`
 - ▶ Nested sublists: `L[1][0]` fetches 42.

Data types (List)

53

- ▶ `L = list('spam')`
 - ▶ Creates a list of all items in any iterable, by calling the type constructor function.
- ▶ `L = [x ** 2 for x in range(9)]`
 - ▶ Creates a list by collecting expression results during iteration.

Data types (List Operators)

54

- ▶ `L.append(X)`
 - ▶ Inserts the single object `X` at the end of `L`, changing the list in-place.
- ▶ `L.sort(key=None, reverse=False)`
 - ▶ Sorts `L` in-place, in ascending order by default.
- ▶ `L.reverse()`
 - ▶ Reverses items in `L` in-place.
- ▶ `L.index(X)`
 - ▶ Returns the index of the first occurrence of object `X` in `L`; raises an exception if not found.

Data types (List Operators)

55

- ▶ `L.insert(i, X)`
 - ▶ Inserts single object `X` into `L` at offset `i`
- ▶ `L.count(X)`
 - ▶ Returns the number of occurrences of `X` in `L`.
- ▶ `L.remove(X)`
 - ▶ Deletes the first occurrence of object `X` from `L`;
- ▶ `L.pop([i])`
 - ▶ Deletes and returns the last (or offset `i`) item in `L`.
- ▶ `L.clear()`
 - ▶ Removes all items from `L`.

Data types (Dictionaries) [c]

56

Dictionaries are mutable (changeable) mappings of object references accessed by key (not position).

- ▶ `{}`
 - ▶ An empty dictionary (not a set).
- ▶ `{'spam': 2, 'eggs': 3}`
 - ▶ A two-item dictionary: keys 'spam' and 'eggs', values 2 and 3
- ▶ `D = {'info': {42: 1, type(''): 2}, 'spam': []}`
 - ▶ Nested dictionaries: `D['info'][42]` fetches 1.

Data types (Dictionaries Operations)

57

- ▶ `D.keys()`
 - ▶ All keys in D.
- ▶ `D.values()`
 - ▶ All stored values in D.
- ▶ `D.items()`
 - ▶ A tuple like view of (key, value)
- ▶ `D.clear()`
 - ▶ Removes all items from D.
- ▶ `'K' in D`
 - ▶ Returns True if D has a key K, or False otherwise

Data Types (Tuples) [c]

58

Tuples are immutable (unchangeable) sequences of object references

- ▶ `()`
 - ▶ An empty tuple.
- ▶ `(0)`
 - ▶ A one-item tuple
- ▶ `(0, 1, 2, 3)`
 - ▶ A four-item tuple.
- ▶ `T = ('spam', (42, 'eggs'))`
 - ▶ Nested tuples: `T[1][1]` fetches 'eggs'.

Data Types (Tuples Operations)

59

- ▶ `T.index(X)`
 - ▶ Returns the index of the first occurrence of object X in tuple T;
- ▶ `T.count(X)`
 - ▶ Returns the number of occurrences of X in tuple T.
- ▶ PS:
 - ▶ Tuples are faster than list
 - ▶ It makes your code safer if you “write-protect” data that does not need to be changed.

Data Types (Files) [c]

60

The built-in `open()` function creates a file object, the most common interface to external files.

- ▶ Input files

- ▶ `infile = open(filename, 'r')`

- ▶ Creates input file object, connected to the named external file. `filename` is normally a string (e.g., `'data.txt'`), and maps to the current working directory unless it includes a directory path prefix (e.g., `r'c:\dir\data.txt'`).
 - ▶ Argument two gives file mode: `'r'` reads text.

- ▶ `infile.read()`

- ▶ Reads entire file, returning its contents as a single string. In text mode (`'r'`), line-ends are translated to `'\n'` by default.

Data Types (Files)

61

- ▶ Output files
 - ▶ `outfile = open(filename, 'w')`
 - ▶ Creates output file object, connected to external file named by `filename` (defined in the preceding section). Mode 'w' writes text
 - ▶ `outfile.write(S)`
 - ▶ Writes all content in string `s` onto file, with no formatting applied. In text mode, `'\n'` is translated to the platform specific line-end marker sequence by default.
- ▶ `file.close()`
 - ▶ Manual close to free resources

Other data types

62

- ▶ Sets [c]
 - ▶ Sets are mutable (changeable) and unordered collections of unique and immutable objects. Sets support mathematical set operations such as union and intersection.
- ▶ Boolean
 - ▶ The Boolean type, named `bool`, provides two predefined constants added to the built-in scope, named `True` and `False`.

Convertors

63

Converter	Converts from	Converts to
<code>list(X)</code>	String, tuple, any iterable	List
<code>tuple(X)</code>	String, list, any iterable	Tuple
<code>int(S [, base])</code>	String or number	Integer, float
<code>float(S)</code>	String or number	Integer, float
<code>str(X)</code>	Any Python object	String
<code>hex(X)</code>	Integer types	Hexadecimal
<code>oct(X)</code>	Integer types	octal

Comparison/Boolean operators

64

Operator	Description
<code>X < Y</code>	Less than
<code>X <= Y</code>	Less than or equal to
<code>X > Y</code>	Greater than
<code>X >= Y</code>	Greater than or equal to
<code>X == Y</code>	Equal to (same value)
<code>X != Y</code>	Not equal to
<code>X is Y</code>	Same object
<code>X is not Y</code>	Negated object identity
<code>X < Y < Z</code>	Chained comparisons
<code>not X</code>	If X is false then True; else, False
<code>X or Y</code>	If X is false then Y; else, X
<code>X and Y</code>	If X is false then X; else, Y

Statements And Syntax

65

- ▶ Control flow
 - ▶ Statements execute **sequentially**, one after another, **unless** control-flow statements are used to branch elsewhere in code.
- ▶ Blocks
 - ▶ A nested block is delimited by indenting all of its statements by the same amount, with any number of spaces or tabs used consistently.
- ▶ Statements
 - ▶ A statement **ends at the end of a line**, but can continue over multiple lines if a physical line ends with a \; an unclosed (), [], or {} pair; or an unclosed, triple-quoted string.
Multiple simple statements can appear on a single line if they are separated with a semicolon (;).

Statements And Syntax

66

- ▶ Comments
 - ▶ Comments start with a # in any column (and not in a string constant) and span to the end of the line; they are ignored by the Python interpreter.
- ▶ Whitespace
 - ▶ Generally significant only to the left of code, where indentation is used to group blocks. Blank lines and spaces are otherwise ignored and optional except within string constants.

Statements

67

- ▶ The Assignment Statement
- ▶ The print Statement
- ▶ The if Statement
- ▶ The while Statement
- ▶ The for Statement
- ▶ The break Statement
- ▶ The continue Statement
- ▶ The del Statement
- ▶ The def Statement
- The return Statement
- The global Statement
- The pass Statement
- The import Statement
- The from Statement
- The class Statement
- The try Statement

The assignment statement

- ▶ `target = expression`
- ▶ `target1 = target2 = expression`
- ▶ `target1, target2 = expression1, expression2`
- ▶ `target1 += expression`

<code>X += Y</code>	<code>X &= Y</code>	<code>X -= Y</code>	<code>X = Y</code>
<code>X *= Y</code>	<code>X ^= Y</code>	<code>X /= Y</code>	<code>X >>= Y</code>
<code>X %= Y</code>	<code>X <<= Y</code>	<code>X **= Y</code>	<code>X //= Y</code>

Sequence assignment

69

Any sequence of values may be assigned to any sequence of names

Example :

```
>>> a, b, c, d = [1, 2, 3, 4]
```

```
>>> a, d
```

```
(1, 4)
```

Extended sequence assignment ^[c]

70

The starred name collects all otherwise unmatched items in a new list

Example :

```
>>> a, *b = [1, 2, 3, 4]
```

```
>>> a, b
```

```
(1, [2, 3, 4])
```

```
>>> a, *b, c = (1, 2, 3, 4)
```

```
>>> a, b, c
```

```
(1, [2, 3], 4)
```


Print statement

71

```
print([value [, value]*] [, sep=str] [, end=str] [, file=object] )
```

- ▶ `sep`
 - ▶ A string to place between values (default is space: ' ').
- ▶ `end`
 - ▶ A string to place at the end of the text printed (default is newline: '\n').
- ▶ `file`
 - ▶ The file-like object to which text is written (default is standard output: `sys.stdout`).

The if statement

72

The if statement selects from among one or more actions

```
if test:  
    suite  
[elif test:  
    suite]*  
[else:  
    suite]
```

There is no switch statement

73

There should be one — and preferably only one — obvious way to do it.

- <https://wiki.python.org/moin/TOOWTDI>



The while statement

75

- ▶ The while loop is a general loop that keeps running the first suite while the test at the top is true. It runs the optional else suite once on exit if the loop ends without running into a break statement in the first suite.

```
while test:
```

```
    suite
```

```
[else:
```

```
    suite]
```

The for statement ^[c]

76

The for loop is a sequence (or other iterable) iteration that assigns items in iterable to target and runs the first suite for each. The for statement runs the optional else suite once on exit if the loop ends without running into a break statement in the first suite.

```
for target in iterable:
```

```
    suite
```

```
[else:
```

```
    suite]
```


The break statement

77

This immediately exits the closest enclosing while or for loop statement, skipping its associated else (if any).

break

The continue statement

78

This immediately goes to the top of the closest enclosing while or for loop statement

continue

The pass statement

79

This is a do-nothing placeholder statement, and is used when syntactically necessary

```
pass
```

```
>>> help('pass')
```

It is useful as a placeholder when a statement is required syntactically, but no code needs to be executed

The del statement ^[c]

80

The del statement deletes variables, items, keys, and attributes.

name is a variable.

```
del name
```

```
del name[i]
```

The def statement [c]

81

The def statement makes new functions, which may also serve as methods in classes.

```
def name([arg,... arg=value,... *arg, **arg]):  
    suite
```

arg	arg=value	*arg	**arg
Matched by name or position	Default value if name is not passed	Collects extra positional arguments as new tuple name	Collects extra keyword arguments as a new dictionary name

The return statement

82

The return statement exits the enclosing function and returns an expression value as the result of the call to the function. If expression is omitted, it defaults to None, which is also the default return value for functions that exit without a return.

Hint: return a tuple for multiple-value function results.

The global statement ^[c]

83

The global statement is a namespace declaration:

When used inside a class or function definition statement, it causes all appearances of name in that context to be treated as references to a global variable of that name.

Whether name is assigned or not, and whether name already exists or not.

This statement allows globals to be created or changed within a function or class.

The import statement

84

The import statement provides module access: it imports a module as a whole.

```
import [package.]* module [as name]
```

The from statement ^[c]

85

The from statement imports a module just as in the import statement (see the preceding section), but also copies variable names from the module to be used without qualification

```
from [package.]* module import name [as othername]
```


The try statement ^[c]

86

The **try** statement catches exceptions. **try** statements can specify **except** clauses with suites that serve as handlers for exceptions raised during the try suite; **else** clauses that run if no exception occurs during the try suite; and **finally** clauses that run whether an exception happens or not.

Object-Oriented Programming

87

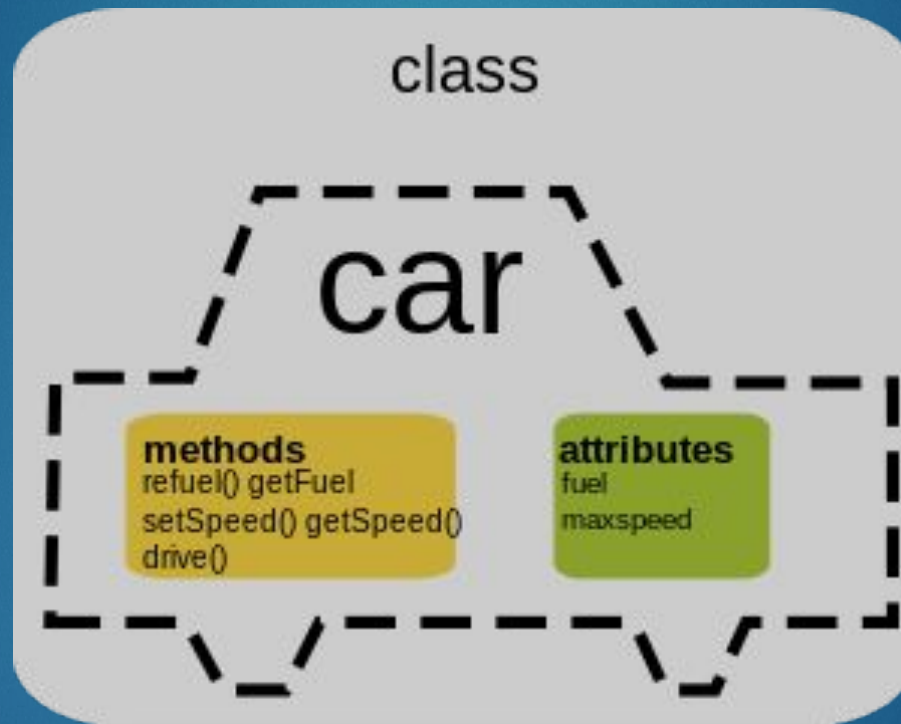
Object-oriented programming (OOP) is a **programming paradigm** based on the concept of "objects", which may contain data, often known as attributes; and code, in the form of procedures, often known as methods.

PS: Paradigms are concerned mainly with the way that code is organized.

- ▶ Encapsulation
 - ▶ insists that you **think about what you expose** to the outside world, it lets you change the implementation of an object without affecting any other code.
- ▶ Inheritance
 - ▶ it lets you write a set of functions, then **expand** them in different direction without changing or copying them in any way.
- ▶ Polymorphism
 - ▶ it allows you to have many different functions, all with the same name, all doing the same job, but on different data.

Object-Oriented Programming

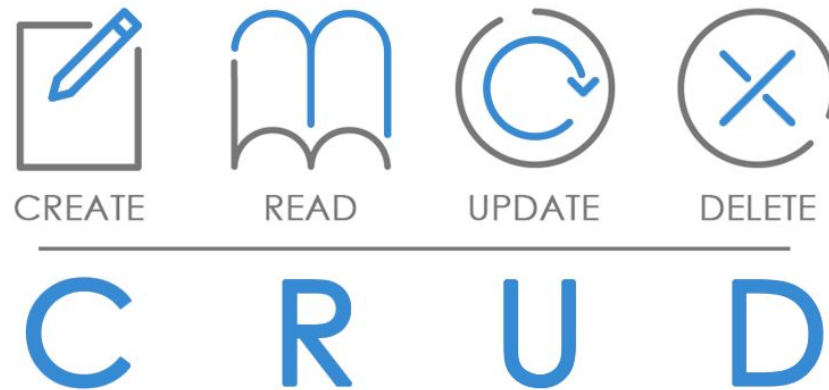
88



The Class Statement _[c]

89

```
class name [ ( super [, super]* [, metaclass=M] ) ]:  
    suite
```



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

NumPy [c]

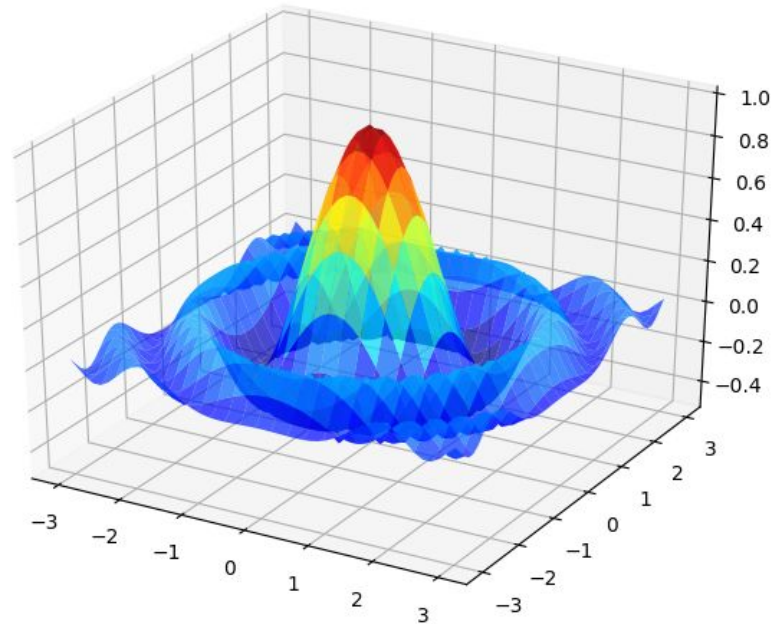
- ▶ Provides support for large, multi-dimensional arrays
- ▶ We can express images as multi-dimensional arrays
 - ▶ Representing images as NumPy arrays is not only computational and resource efficient, but many other image processing and machine learning libraries use NumPy array representations as well
- ▶ Furthermore, by using NumPy's built-in high-level mathematical functions, we can quickly perform numerical analysis on an Matrix/Images
- <http://www.numpy.org/>

SciPy

- ▶ SciPy is a Python-based ecosystem of open-source software for mathematics, science, and engineering.
- ▶ SciPy contains modules for optimization, linear algebra, integration, special functions, FFT, signal and image processing and other tasks common in science and engineering.

matplotlib [c]

- ▶ Matplotlib is a plotting library.
- ▶ Matplotlib tries to make easy things easy and hard things possible.
- ▶ You can generate plots, histograms, bar charts, etc. with just a few lines of code.



PIL (Python Image Library)

- ▶ Provides general image handling and lots of useful basic image operations like resizing, cropping, rotating, color conversion and much more. With PIL you can read images from most formats and write to the most common ones.

OpenCV

- ▶ OpenCV comes with functions for reading and writing images as well as matrix operations and math libraries.
- ▶ Mainly aimed at real-time computer vision.
 - ▶ <https://en.wikipedia.org/wiki/OpenCV#Applications>
- ▶ Videos

H5PY

- ▶ h5py package is a Pythonic interface to the HDF5 binary data format.
 - ▶ It lets you store huge amounts of numerical data, and easily manipulate that data from NumPy.
 - ▶ For example, you can slice into multi-terabyte datasets stored on disk, as if they were real NumPy arrays.
 - ▶ Thousands of datasets can be stored in a single file, categorized and tagged however you want.
 - ▶ The **h5py** library is the de-facto standard in Python to store large numerical datasets.
- <http://docs.h5py.org/en/latest/quick.html>



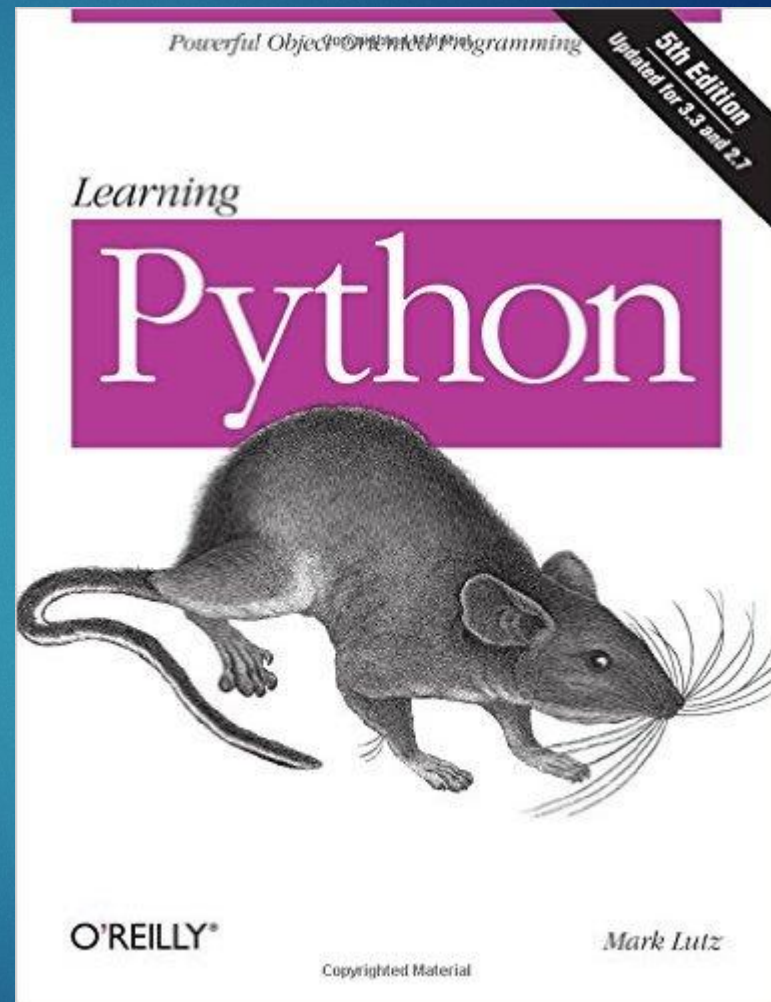
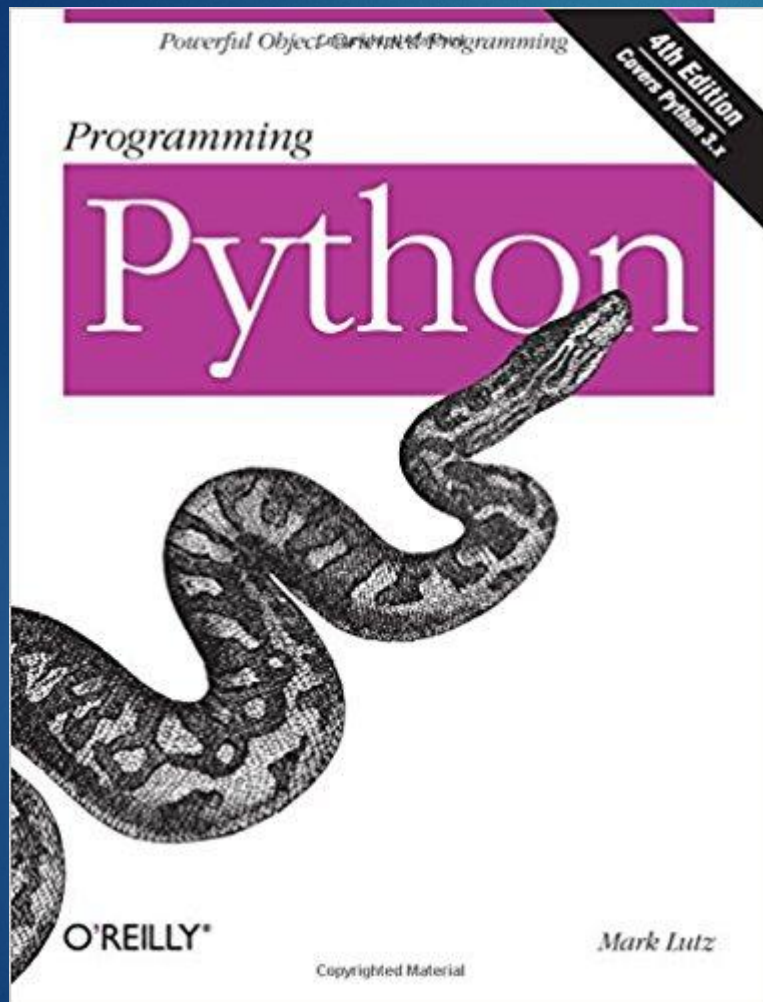
Python's user base is vast and growing - it's not going away any time soon.

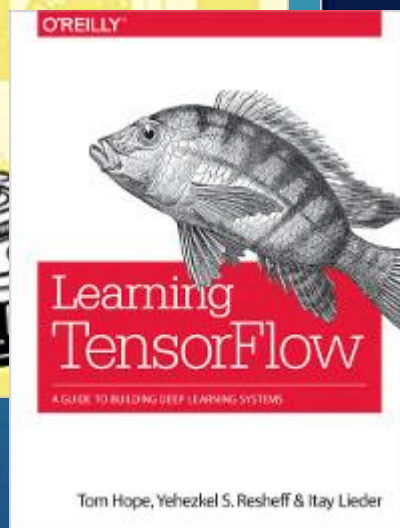
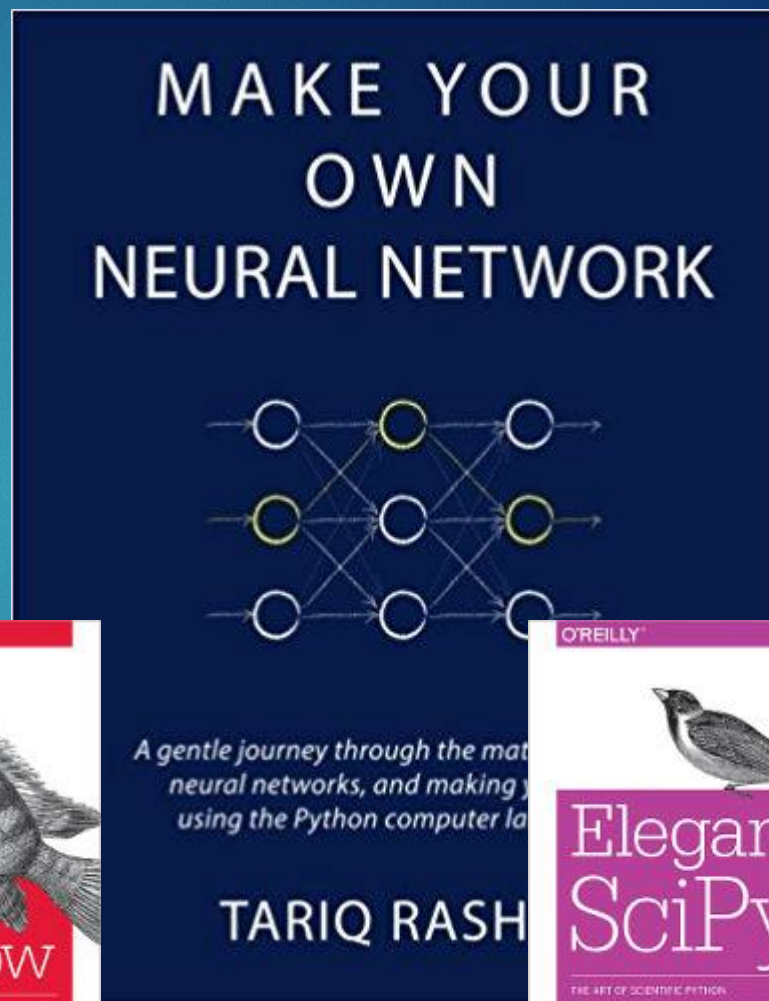
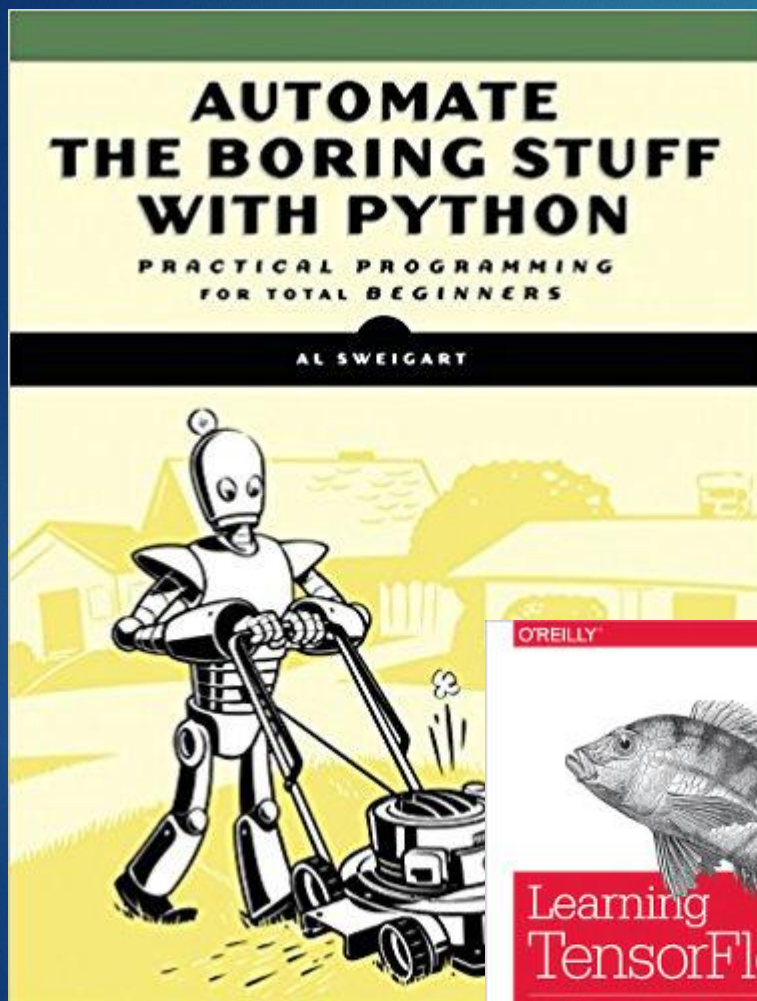
Utilized by the likes of Nokia, Google, and even NASA for it's easy syntax.

Its support of multiple programming paradigms, including object-oriented Python programming, functional Python programming, and parallel programming models makes it a highly adaptive choice.

Books

100





- ▶ Websites:

- ▶ <https://docs.python.org>

- ▶ <https://wiki.python.org>

- ▶ Books:

- ▶ Python Pocket Reference (5th ed) [Lutz 2014-02-09]

- ▶ Programming Computer Vision, Jan Erik Solem ,[2012 Jan]



Department of Computer Engineering, Shahid Bahonar University of Kerman

Milad Abolhasani

Mail: milad@eng.uk.ac.ir

Web: tuxgeek.ir

Github: [Github.com/Ravexina](https://github.com/Ravexina)

Stack: [Stackexchange.com/users/4177764](https://stackoverflow.com/users/4177764)

Asgar Jangah

Email : a_Jangah@eng.uk.ac.ir

Github: [Github.com/asgarjangah](https://github.com/asgarjangah)



Mr. Afzalipour

Within **Today**, Maybe tomorrow might be too **late** ...