

# **Big Data and Data Mining**

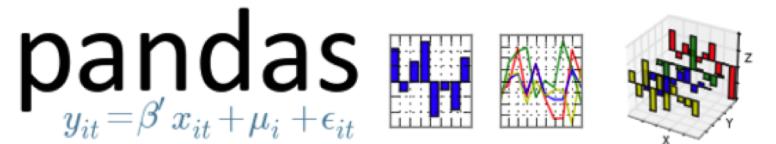
## 02. Introduction to Python



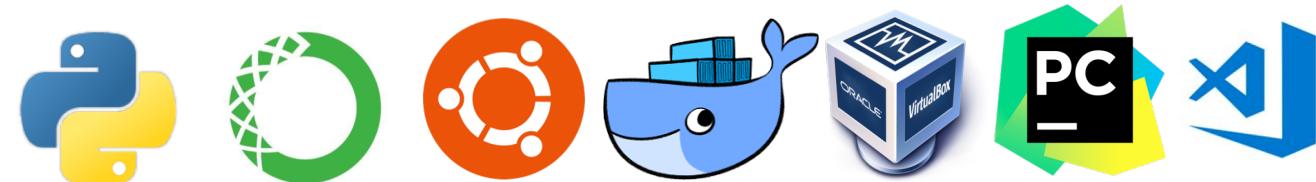
# Why Python for Data Science?



**IP[y]:** IPython  
Interactive Computing



# Requirement



- Operation System
  - Linux OS: Ubuntu distribution is highly recommended.
  - macOSX is also a decent platform
  - Windows 10 with native WSL Ubuntu
- Python Programming Environment
  - Interpreter, Library etc.
- Editor or Integrated Development Environment
  - Syntax Highlight, Easy Configuration etc.
- Option For Enterprise Production
  - Any Laptop with Text Editor (VSCode) and SSH toolchain
  - IDEs, such as *PyCharm Professional*, are always handy
  - Access credential to the docker, virtual machine on remote DEV server

a) <https://docs.microsoft.com/en-us/windows/wsl/install-win10>



# Python 2 or Python 3



| PYTHON 2                                                            |   | PYTHON 3                                                                      |
|---------------------------------------------------------------------|---|-------------------------------------------------------------------------------|
| ← Legacy                                                            | > | Future →                                                                      |
| It is still entrenched in the software at certain companies         |   | It will take over Python 2 by 2020                                            |
| Folder icon with '2'                                                | ≠ | Folder icon with '3'                                                          |
| Many older libraries built for Python 2 are not forwards-compatible |   | Many today's developers are creating libraries strictly for use with Python 3 |
| 0100<br>0001 ASCII                                                  | + | 0000<br>0000 Unicode                                                          |
| Strings are stored as ASCII by default                              |   | Text strings are Unicode by default                                           |
| ≈ 5/2=2                                                             | ≠ | 5/2=2.5 =                                                                     |
| It rounds your calculation down to the nearest whole number         |   | The expression 5 / 2 will return the expected result                          |
| print "hello"                                                       | ≠ | print ("hello")                                                               |
| Python 2 print statement                                            |   | The print statement has been replaced with a print () function                |



LEARNTOCODEWITH.ME

- a) <https://dev.to/kostassar/python-2-vs-python-3-2f03>  
b) <https://learntocodewith.me/programming/python/python-2-vs-python-3/>



# Official Resource

Practice is the key to learning any programming language!

The screenshot shows the Python.org homepage. The top navigation bar includes links for 'Donate', 'Search', 'GO', and 'Socialize'. Below this is a main menu with 'About' (selected), 'Downloads', 'Documentation', 'Community', 'Success Stories', 'News', and 'Events'. On the left, a sidebar lists 'All releases', 'Source code', 'Windows', 'Mac OS X', 'Other Platforms', 'License', and 'Alternative Implementations'. The 'Downloads' section for Mac OS X features a large button for 'Python 3.7.2' and a note about using Python on other systems. The 'Documentation' section highlights the standard documentation and offers links to 'Python 3.x Docs' and 'Python 2.x Docs'. A large banner at the bottom states 'Python is and in'.

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

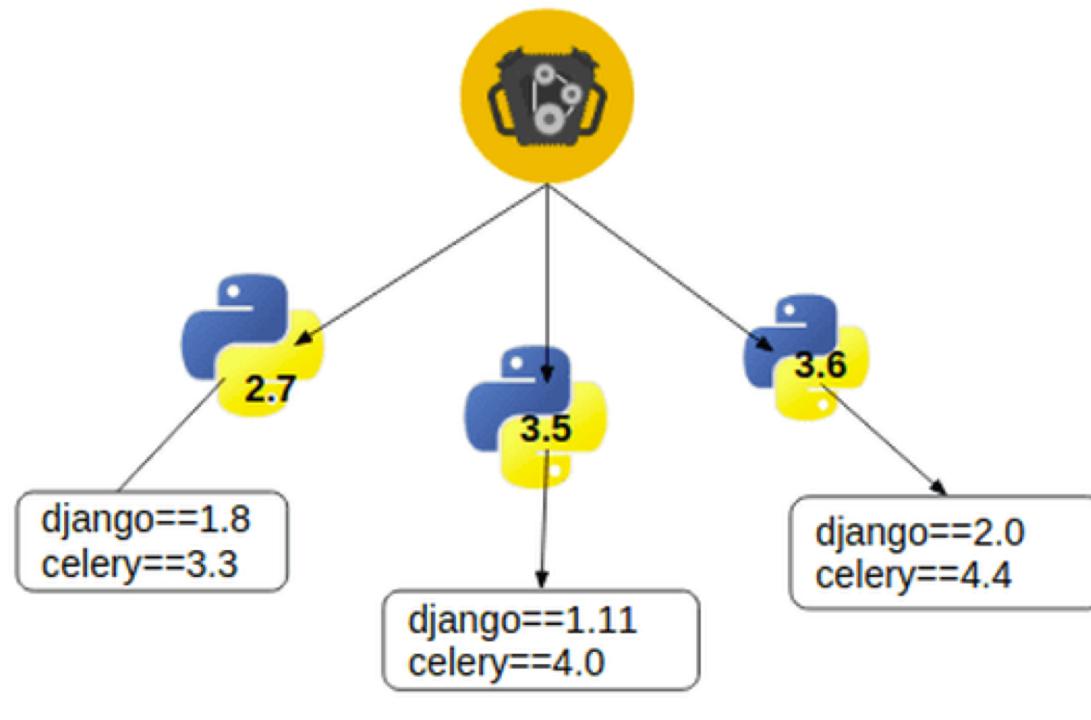


# Installation – Basic

- Interpreter and Standard Library
  - Official Python Distribution: Highly recommended for professional
    - Lightweight distribution with essential components
    - For Windows & macOS: Binary Installer
    - For Linux: *apt-get install ...* or other package manager
    - *pip* as library management
  - Anaconda Distribution: If you are new to python, check this out
    - Fully packed but with redundant packages
    - *conda* as library management
    - *pip* as library management
    - Third-party library repositories
- Official Doc is the best material for learning python as always



# Installation – Advance



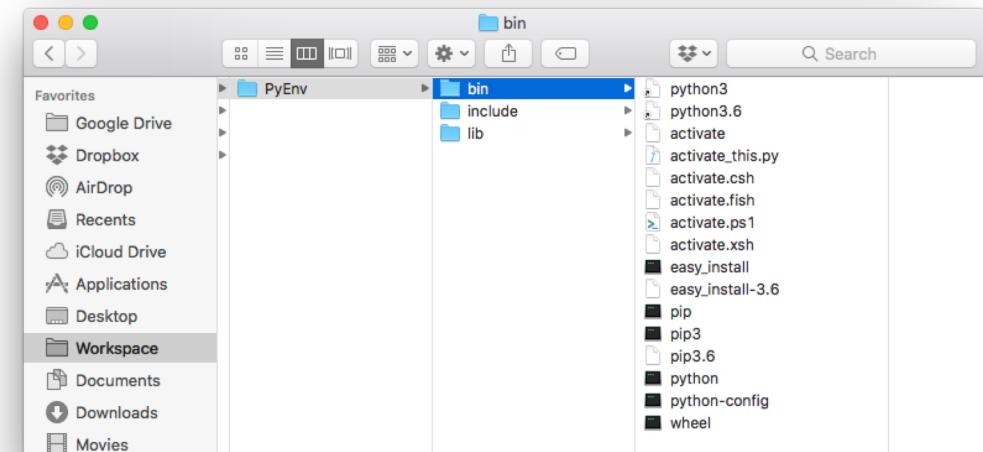
a) <https://virtualenv.pypa.io/en/latest/>

```

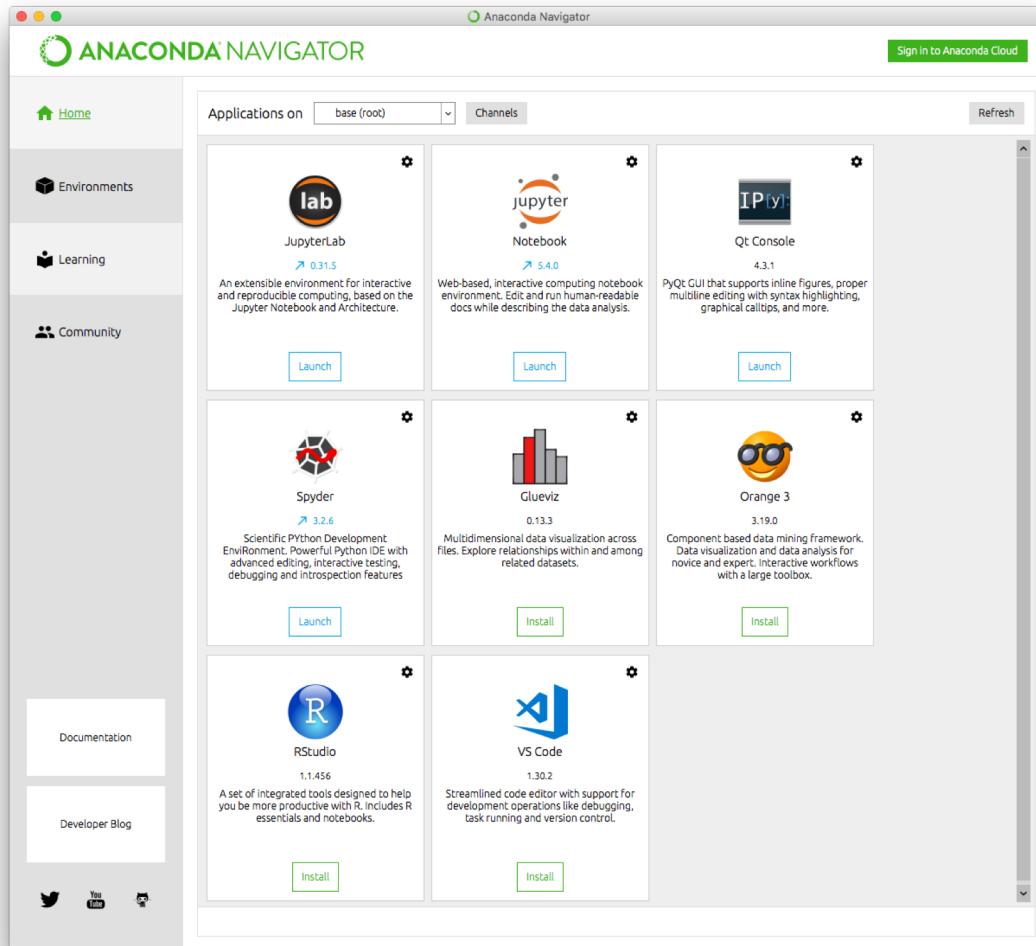
1. bash
CSJDs-iMac:~ jjdengjj$ virtualenv --version
-bash: virtualenv: command not found
CSJDs-iMac:~ jjdengjj$
CSJDs-iMac:~ jjdengjj$ pip install virtualenv
Collecting virtualenv
  Downloading https://files.pythonhosted.org/packages/8f/f1/c0b069ca6cb44f968171
5232e6d3d65c75866dd231c5e4a88e80a46634bb/virtualenv-16.3.0-py2.py3-none-any.whl
(2.0MB)
  100% |████████████████████████████████| 2.0MB 8.2MB/s
Requirement already satisfied: setuptools>=18.0.0 in ./anaconda3/lib/python3.6/
site-packages (from virtualenv) (38.4.0)
Installing collected packages: virtualenv
Successfully installed virtualenv-16.3.0
CSJDs-iMac:~ jjdengjj$ 
  
```

```

1. bash
CSJDs-iMac:Workspace jjdengjj$ mkdir BDDM
CSJDs-iMac:Workspace jjdengjj$ virtualenv ./BDDM/PyEnv
Using base prefix '/Users/jjdengjj/anaconda3'
New python executable in /Users/jjdengjj/Workspace/BDDM/PyEnv/bin/python
Installing setuptools, pip, wheel...
done.
CSJDs-iMac:Workspace jjdengjj$ 
  
```



# Anaconda for Newbie



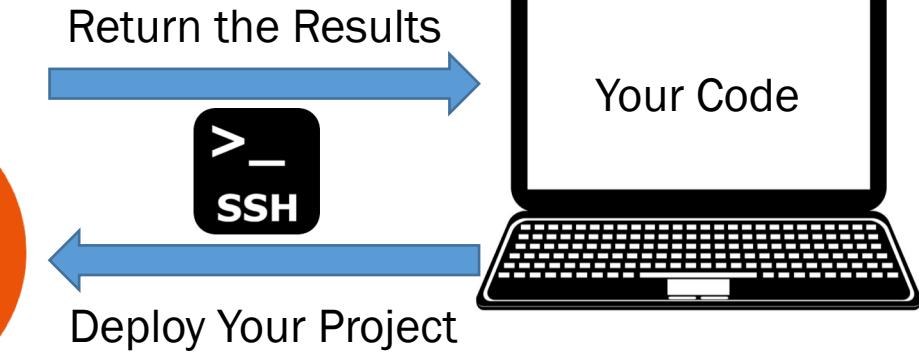
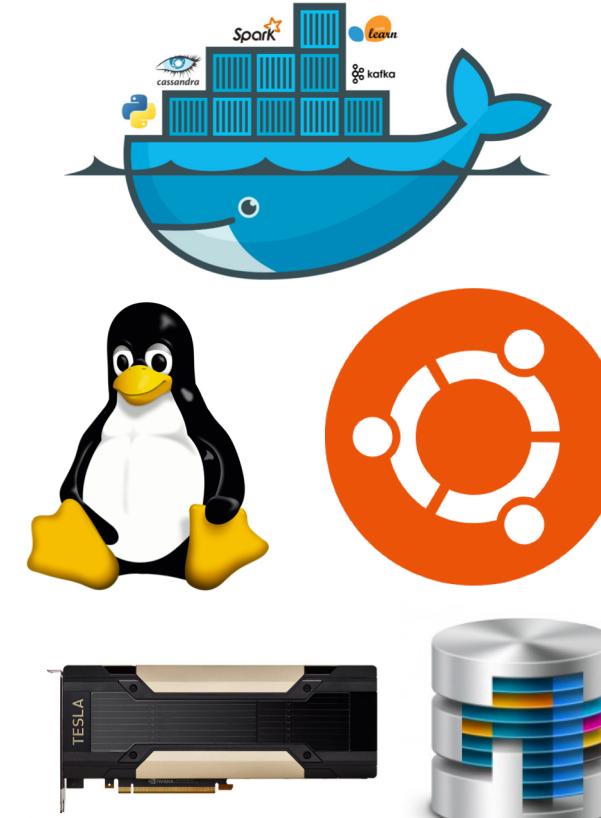
The screenshot shows the Anaconda Navigator application window. The main area is titled "base (root)" and displays a list of installed packages. At the top right are buttons for "Search Environments", "Installed", "Channels", "Update index...", and "Search Packages". Below is a table of packages:

| Name                             | Description                                                                                                                   | Version   |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------|-----------|
| <code>_tfflow_1100_select</code> | Absell python common libraries, see <a href="https://github.com/absell/_absell-py">https://github.com/absell/_absell-py</a> . | 0.0.2     |
| <code>absl-py</code>             | Read, rewrite, and write python absts nicely                                                                                  | 0.5.0     |
| <code>astor</code>               |                                                                                                                               | 0.7.1     |
| <code>blas</code>                |                                                                                                                               | 1.0       |
| <code>bz2</code>                 | High-quality data compressor                                                                                                  | 1.0.6     |
| <code>ca-certificates</code>     | Certificates for use with other packages.                                                                                     | 2018.0... |
| <code>cairo</code>               | Cairo is a 2d graphics library with support for multiple output devices.                                                      | 1.14.12   |
| <code>certifi</code>             | Python package for providing mozilla's ca bundle.                                                                             | 2018.8.24 |
| <code>cloudpickle</code>         | Extended pickling support for python objects                                                                                  | 0.5.6     |
| <code>cycler</code>              | Composable style cycles.                                                                                                      | 0.10.0    |
| <code>dash-core</code>           | Parallel python with task scheduling                                                                                          | 0.19.2    |
| <code>decorator</code>           | Better living through python with decorators.                                                                                 | 4.3.0     |
| <code>ffmpeg</code>              | Cross-platform solution to record, convert and stream audio and video.                                                        | 4.0       |
| <code>fontconfig</code>          | A library for configuring and customizing font access.                                                                        | 2.13.0    |
| <code>freetype</code>            | A free, high-quality, and portable font engine                                                                                | 2.9.1     |
| <code>gast</code>                | Python ast that abstracts the underlying python version                                                                       | 0.2.0     |
| <code>gettext</code>             | Internationalization package.                                                                                                 | 0.19.8.1  |
| <code>glib</code>                | Provides core application building blocks for libraries and applications written in C.                                        | 2.56.2    |
| <code>graphite2</code>           | A "smart font" system that handles the complexities of lesser-known languages of the world.                                   | 1.3.12    |
| <code>grpcio</code>              | Http/2-based rpc framework                                                                                                    | 1.12.1    |
| <code>harfbuzz</code>            | An opentype text shaping engine.                                                                                              | 1.8.8     |
| <code>hdfs</code>                | HDFS is a data model, library, and file format for storing and managing data                                                  | 1.10.2    |

82 packages available

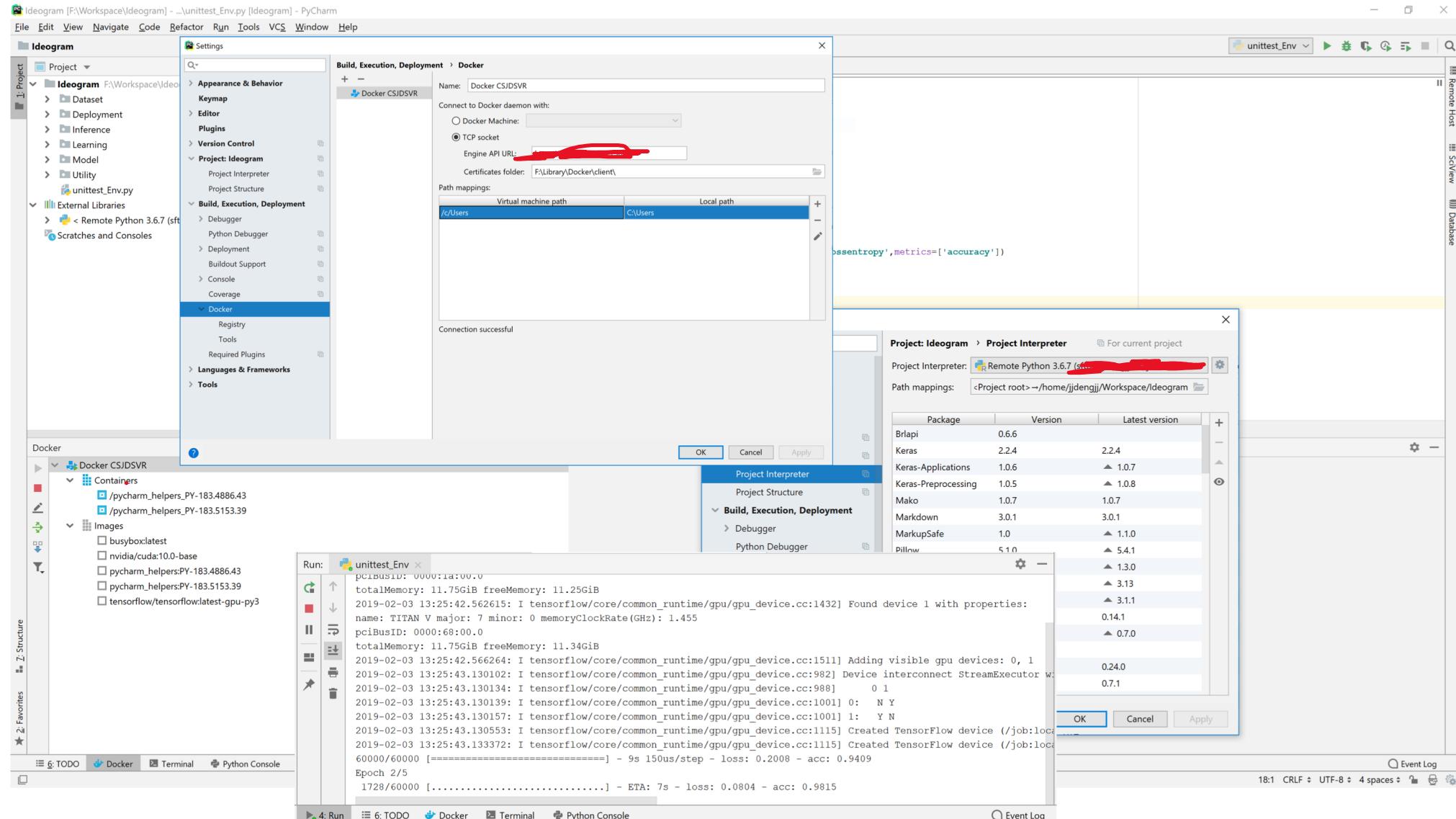


# Remote Server or Docker for Professional



a) Welcome to my office for viewing those lovely pieces.

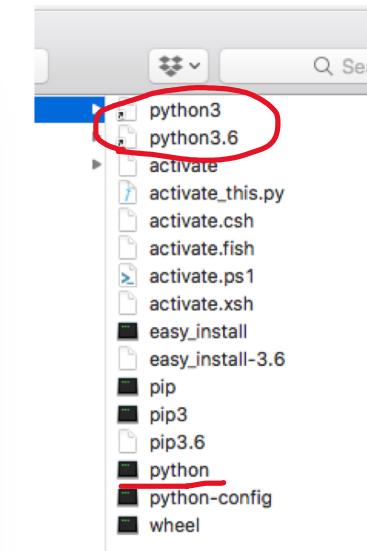
# Developing with PyCharm



# Using Python Interpreter

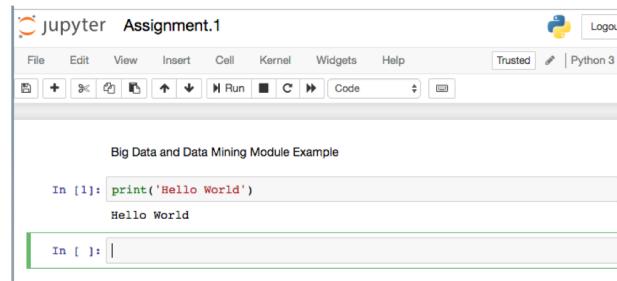
- The binary distribution of *python* or *python.exe* is the python language interpreter, similar to *java* or *java.exe*.

```
CSJDs-iMac:~ jjdengjj$ python
Python 3.6.8 |Anaconda custom (64-bit)| (default, Dec 29 2018, 19:04:46)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello World')
Hello World
>>> exit()
CSJDs-iMac:~ jjdengjj$ python -c "print('Hello World')"
Hello World
CSJDs-iMac:~ jjdengjj$ python Desktop/src.py
Hello World
CSJDs-iMac:~ jjdengjj$
```

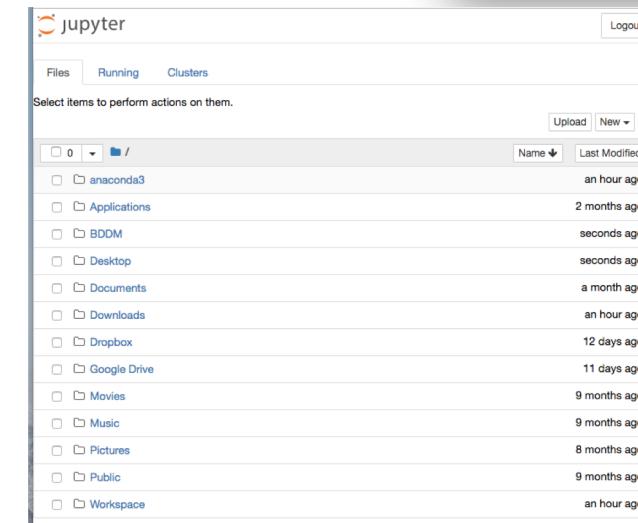


# Using Jupyter Environment

- Web-based Python programming environment
  - Ideally for python beginner
- Installation: *pip install jupyter*
- Get Started: *jupyter notebook*
  - Running a small web server on your local machine
  - <http://localhost:8888/tree>
- Stop Jupyter:
  - Press “**CTRL+C**” in the terminal



a) <https://jupyter.org/>

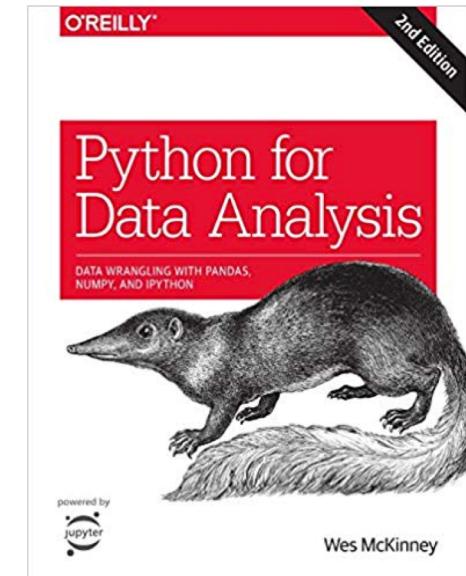
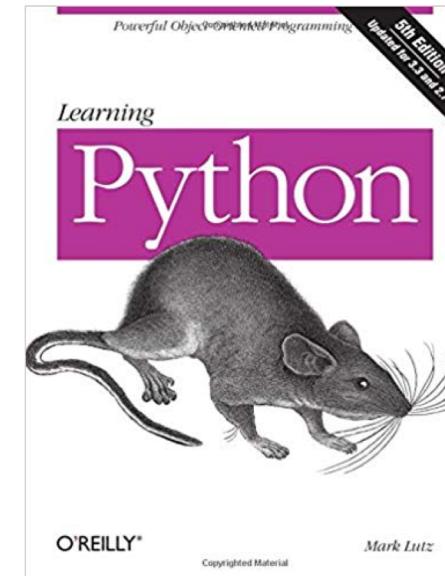


```
1. bash
Last login: Sun Feb  3 13:18:37 on ttys001
CSJDs-iMac:~ jjdengjj$ jupyter notebook
[I 13:48:46.212 NotebookApp] JupyterLab beta preview loaded from /Users/jjdengjj/anaconda3/lib/python3.6/site-packages/jupyterlab
[I 13:48:46.213 NotebookApp] JupyterLab application directory is /Users/jjdengjj/anaconda3/share/jupyter/lab
[I 13:48:46.224 NotebookApp] Serving notebooks from local directory: /Users/jjdengjj
[I 13:48:46.224 NotebookApp] 0 active kernels
[I 13:48:46.224 NotebookApp] The Jupyter Notebook is running at:
[I 13:48:46.224 NotebookApp] http://localhost:8888/
[I 13:48:46.224 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).
^C[I 13:53:34.900 NotebookApp] interrupted
Serving notebooks from local directory: /Users/jjdengjj
0 active kernels
The Jupyter Notebook is running at:
http://localhost:8888/
Shutdown this notebook server (y/[n])? y
[C 13:53:36.429 NotebookApp] Shutdown confirmed
[I 13:53:36.430 NotebookApp] Shutting down 0 kernels
CSJDs-iMac:~ jjdengjj$
```

# Essential Programming

- Logic and Numerical Operation
  - i.e. ==, =, +, -, \*, / etc.
- Data Structure
  - i.e. *List, Dictionary, String* etc.
- Programme Structure
  - i.e. *Condition, Loop, Function* etc.
- Project Structure and OOP
  - i.e. *Module Import, Class* etc.
- Standard Library
  - i.e. built-in function, basic IO etc.
- External Library and **More...**

This is NOT another programming module, so we are NOT going to cover all language features, but we will review the essential knowledge to get you started in this lecture.



# Logic and Numerical Operations

| Operation            | Result                                                                       |
|----------------------|------------------------------------------------------------------------------|
| <code>x or y</code>  | if <code>x</code> is false, then <code>y</code> , else <code>x</code>        |
| <code>x and y</code> | if <code>x</code> is false, then <code>x</code> , else <code>y</code>        |
| <code>not x</code>   | if <code>x</code> is false, then <code>True</code> , else <code>False</code> |

| Operation           | Meaning                 |
|---------------------|-------------------------|
| <code>&lt;</code>   | strictly less than      |
| <code>&lt;=</code>  | less than or equal      |
| <code>&gt;</code>   | strictly greater than   |
| <code>&gt;=</code>  | greater than or equal   |
| <code>==</code>     | equal                   |
| <code>!=</code>     | not equal               |
| <code>is</code>     | object identity         |
| <code>is not</code> | negated object identity |

| Operation                    | Result                                                                                                               |
|------------------------------|----------------------------------------------------------------------------------------------------------------------|
| <code>x + y</code>           | sum of <code>x</code> and <code>y</code>                                                                             |
| <code>x - y</code>           | difference of <code>x</code> and <code>y</code>                                                                      |
| <code>x * y</code>           | product of <code>x</code> and <code>y</code>                                                                         |
| <code>x / y</code>           | quotient of <code>x</code> and <code>y</code>                                                                        |
| <code>x // y</code>          | floored quotient of <code>x</code> and <code>y</code>                                                                |
| <code>x % y</code>           | remainder of <code>x / y</code>                                                                                      |
| <code>-x</code>              | <code>x</code> negated                                                                                               |
| <code>+x</code>              | <code>x</code> unchanged                                                                                             |
| <code>abs(x)</code>          | absolute value or magnitude of <code>x</code>                                                                        |
| <code>int(x)</code>          | <code>x</code> converted to integer                                                                                  |
| <code>float(x)</code>        | <code>x</code> converted to floating point                                                                           |
| <code>complex(re, im)</code> | a complex number with real part <code>re</code> , imaginary part <code>im</code> . <code>im</code> defaults to zero. |
| <code>c.conjugate()</code>   | conjugate of the complex number <code>c</code>                                                                       |
| <code>divmod(x, y)</code>    | the pair <code>(x // y, x % y)</code>                                                                                |
| <code>pow(x, y)</code>       | <code>x</code> to the power <code>y</code>                                                                           |
| <code>x ** y</code>          | <code>x</code> to the power <code>y</code>                                                                           |



# Logic and Numerical Operations

```
In [ ]: a=10
a.
```

bit\_length  
conjugate  
denominator  
from\_bytes  
imag  
numerator  
real  
to\_bytes

```
In [7]: a=10
print(a.bit_length())
print(bin(a))

4
0b1010
```

| Operation    | Result                                     |
|--------------|--------------------------------------------|
| $x \mid y$   | bitwise <i>or</i> of $x$ and $y$           |
| $x \wedge y$ | bitwise <i>exclusive or</i> of $x$ and $y$ |
| $x \& y$     | bitwise <i>and</i> of $x$ and $y$          |
| $x \ll n$    | $x$ shifted left by $n$ bits               |
| $x \gg n$    | $x$ shifted right by $n$ bits              |
| $\sim x$     | the bits of $x$ inverted                   |

```
In [6]: a=10
print(bin(a))
b=14
print(bin(b))
c=a^b
print(c)
print(bin(c))

0b1010
0b1110
4
0b100
```

a) More standard numerical operations/functions are defined in math etc. modules.

# Data Structure – Sequence Types

- List – Mutable
  - Using a pair of square brackets to denote the empty list: []
  - Using square brackets, separating items with commas: [a], [a, b, c]
- Tuple – Immutable
  - Using a pair of parentheses to denote the empty tuple: ()
  - Using a trailing comma for a singleton tuple: a, or (a,)
- Range – Immutable
  - The *range* type represents an immutable sequence of numbers and is commonly used for looping a specific number of times in *for* loops.
- An object with a fixed value. Immutable objects include numbers, strings and tuples. Such an object cannot be altered. A new object has to be created if a different value has to be stored.



# Data Structure – Sequence Types

| Operation                         | Result                                                                                                        |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------|
| <code>x in s</code>               | True if an item of <i>s</i> is equal to <i>x</i> , else False                                                 |
| <code>x not in s</code>           | False if an item of <i>s</i> is equal to <i>x</i> , else True                                                 |
| <code>s + t</code>                | the concatenation of <i>s</i> and <i>t</i>                                                                    |
| <code>s * n or n * s</code>       | equivalent to adding <i>s</i> to itself <i>n</i> times                                                        |
| <code>s[i]</code>                 | <i>i</i> th item of <i>s</i> , origin 0                                                                       |
| <code>s[i:j]</code>               | slice of <i>s</i> from <i>i</i> to <i>j</i>                                                                   |
| <code>s[i:j:k]</code>             | slice of <i>s</i> from <i>i</i> to <i>j</i> with step <i>k</i>                                                |
| <code>len(s)</code>               | length of <i>s</i>                                                                                            |
| <code>min(s)</code>               | smallest item of <i>s</i>                                                                                     |
| <code>max(s)</code>               | largest item of <i>s</i>                                                                                      |
| <code>s.index(x[, i[, j]])</code> | index of the first occurrence of <i>x</i> in <i>s</i> (at or after index <i>i</i> and before index <i>j</i> ) |
| <code>s.count(x)</code>           | total number of occurrences of <i>x</i> in <i>s</i>                                                           |

Note: brackets and square brackets

| Operation                          | Result                                                                                                            | Notes |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------|-------|
| <code>s[i] = x</code>              | item <i>i</i> of <i>s</i> is replaced by <i>x</i>                                                                 |       |
| <code>s[i:j] = t</code>            | slice of <i>s</i> from <i>i</i> to <i>j</i> is replaced by the contents of the iterable <i>t</i>                  |       |
| <code>del s[i:j]</code>            | same as <code>s[i:j] = []</code>                                                                                  |       |
| <code>s[i:j:k] = t</code>          | the elements of <code>s[i:j:k]</code> are replaced by those of <i>t</i>                                           | (1)   |
| <code>del s[i:j:k]</code>          | removes the elements of <code>s[i:j:k]</code> from the list                                                       |       |
| <code>s.append(x)</code>           | appends <i>x</i> to the end of the sequence (same as <code>s[len(s):len(s)] = [x]</code> )                        |       |
| <code>s.clear()</code>             | removes all items from <i>s</i> (same as <code>del s[:]</code> )                                                  | (5)   |
| <code>s.copy()</code>              | creates a shallow copy of <i>s</i> (same as <code>s[:]</code> )                                                   | (5)   |
| <code>s.extend(t) or s += t</code> | extends <i>s</i> with the contents of <i>t</i> (for the most part the same as <code>s[len(s):len(s)] = t</code> ) |       |
| <code>s *= n</code>                | updates <i>s</i> with its contents repeated <i>n</i> times                                                        | (6)   |
| <code>s.insert(i, x)</code>        | inserts <i>x</i> into <i>s</i> at the index given by <i>i</i> (same as <code>s[i:i] = [x]</code> )                |       |
| <code>s.pop([i])</code>            | retrieves the item at <i>i</i> and also removes it from <i>s</i>                                                  | (2)   |
| <code>s.remove(x)</code>           | remove the first item from <i>s</i> where <code>s[i]</code> is equal to <i>x</i>                                  | (3)   |
| <code>s.reverse()</code>           | reverses the items of <i>s</i> in place                                                                           | (4)   |

Mutable type only



# Data Structure – Sequence Types

- class range(stop)
- class range(start, stop[, step])
  - For a positive step, the contents of a range r are determined by the formula  $r[i] = start + step * i$  where  $i \geq 0$  and  $r[i] < stop$ .
  - For a negative step, the contents of the range are still determined by the formula  $r[i] = start + step * i$ , but the constraints are  $i \geq 0$  and  $r[i] > stop$ .

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(0, 30, 5))
[0, 5, 10, 15, 20, 25]
>>> list(range(0, 10, 3))
[0, 3, 6, 9]
>>> list(range(0, -10, -1))
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
>>> list(range(0))
[]
>>> list(range(1, 0))
[]
```

In [11]:

```
i = range(10)
print(i)
print(list(i))

range(0, 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Note: *range* gives an object not a list of value

# Data Structure – Dictionary

- A mapping object maps hashable values to arbitrary objects. Mappings are mutable objects. There is currently only one standard mapping type, the dictionary.
  - i.e. vehicle registration number

```
>>> a = dict(one=1, two=2, three=3)
>>> b = {'one': 1, 'two': 2, 'three': 3}
>>> c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
>>> d = dict([('two', 2), ('one', 1), ('three', 3)])
>>> e = dict({'three': 3, 'one': 1, 'two': 2})
>>> a == b == c == d == e
True
```

In [14]:

```
a = dict(one=1, two=2, three=3)
b = {'one': 1, 'two': 2, 'three': 3}
c = b
print(a==b)
print(a is b)
print(c is b)
c['one'] = 0
print(b)
```

```
True
False
True
{'one': 0, 'two': 2, 'three': 3}
```

# Data Structure – Set

- A set object is an unordered collection of distinct hashable objects. Common uses include membership testing, **removing duplicates from a sequence**, and computing mathematical operations such as intersection, union, difference, and symmetric difference.

```
>>> basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}>>>
>>> print(basket)                                     # show that duplicates have been removed
{'orange', 'banana', 'pear', 'apple'}
>>> 'orange' in basket                            # fast membership testing
True
>>> 'crabgrass' in basket
False

>>> # Demonstrate set operations on unique letters from two words
...
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                                              # unique letters in a
{'a', 'r', 'b', 'c', 'd'}
>>> a - b                                         # letters in a but not in b
{'r', 'd', 'b'}
>>> a | b                                         # letters in a or b or both
{'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
>>> a & b                                         # letters in both a and b
{'a', 'c'}
>>> a ^ b                                         # letters in a or b but not both
{'r', 'd', 'b', 'm', 'z', 'l'}
```



# Data Structure – String

A='100' v.s. A=100 ?

- Single quotes as str variable: ' allows embedded "double" quotes'
- Double quotes as str variable: " allows embedded 'single' quotes".
- Triple quoted: """Three single quotes""", """Three double quotes"""
  - Triple quoted is for multiple line comments.
  - Triple quoted strings may span multiple lines - all associated whitespace will be included in the string literal.
  - # is for single line comment.

```
def architecture(self, sizeCBlocks=_defaultCBlock, sizeDBlocks=_defaultDBlock, nonLinear='relu', onDropout=0, onBatchNorm=False):
    """
    Build LeNet Architecture
    :param sizeCBlocks: A List of Tuple of Convolutional Kernel Configuration, (nFilter, sizeKernel, sizePooling, sizeStride)
    :param sizeDBlocks: A List of Integer of Dense Layer Configuration, [nNeuros]
    :param nonLinear: Name of Activation Function, See https://keras.io/activations/
    :param onDropout: 0 = No Random Dropout, Positive Non-Zero Value (0, 1) = Percentage
    :param onBatchNorm: Perform Batch Normalization After Activation
    :return: self._model
    """

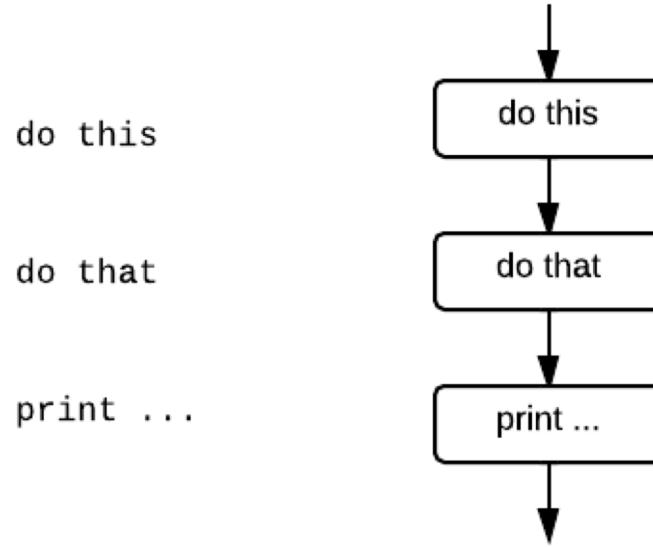
```

```
=====
# Copyright 2017 The DLPractice Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
=====
# Jingjing Deng (jjdengjj@gmail.com)
=====
```

a) More str operations/functions can be found in python reference document.

```
>>> table = {'Sjoerd': 4127, 'Jack': 4098, 'Dcab': 7678}
>>> for name, phone in table.items():
...     print(f'{name}:10 ==> {phone:10d}')
...
Sjoerd    ==>    4127
Jack      ==>    4098
Dcab      ==>    7678
```

# Programme Structure – Condition Control



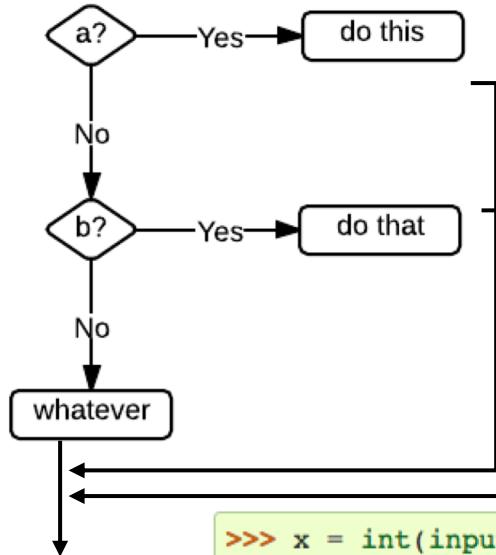
do this

do that

print ...

```

if a:
    do this
elif b:
    do that
else:
    whatever
  
```



**Don't forget the Colon.**

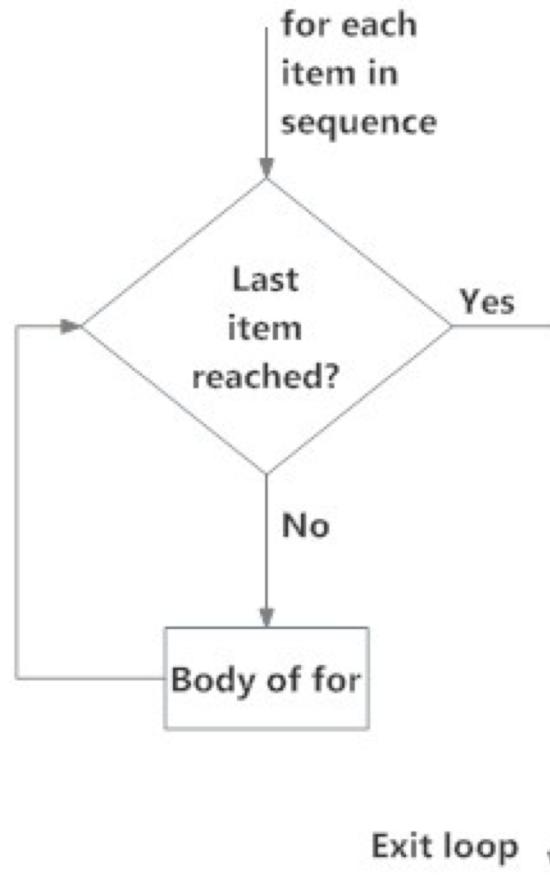
```

>>> x = int(input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Single')
... else:
...     print('More')
...
More
  
```

Note: Common logic mistake is that the conditions are not exclusive between each others.

a) [https://www.codecademy.com/en/forum\\_questions/51684a3d4ce76309b4001b9c](https://www.codecademy.com/en/forum_questions/51684a3d4ce76309b4001b9c)

# Programme Structure – For Loop Flow



```

>>> a = ['Mary', 'had', 'a', 'little', 'lamb']
>>> for i in range(len(a)):
...     print(i, a[i])
...
0 Mary
1 had
2 a
3 little
4 lamb
  
```

Don't forget the Colon.

```

>>> # Measure some strings:
... words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print(w, len(w))
...
cat 3
window 6
defenestrate 12
  
```

# Programme Structure – Loop with else

- Loop statements may have an else clause; it is executed when the loop terminates through exhaustion of the list (with for) or when the condition becomes false (with while), but not when the loop is terminated by a break statement.
- *break* and *continue*

```
>>> for n in range(2, 10):
...     for x in range(2, n):
...         if n % x == 0:
...             print(n, 'equals', x, '*', n//x)
...             break
...     else:
...         # loop fell through without finding a factor
...         print(n, 'is a prime number')
...
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

Don't forget the Colon.

```
>>> for num in range(2, 10):
...     if num % 2 == 0:
...         print("Found an even number", num)
...         continue
...     print("Found a number", num)
Found an even number 2
Found a number 3
Found an even number 4
Found a number 5
Found an even number 6
Found a number 7
Found an even number 8
Found a number 9
```



# Programme Structure – While Loop Flow

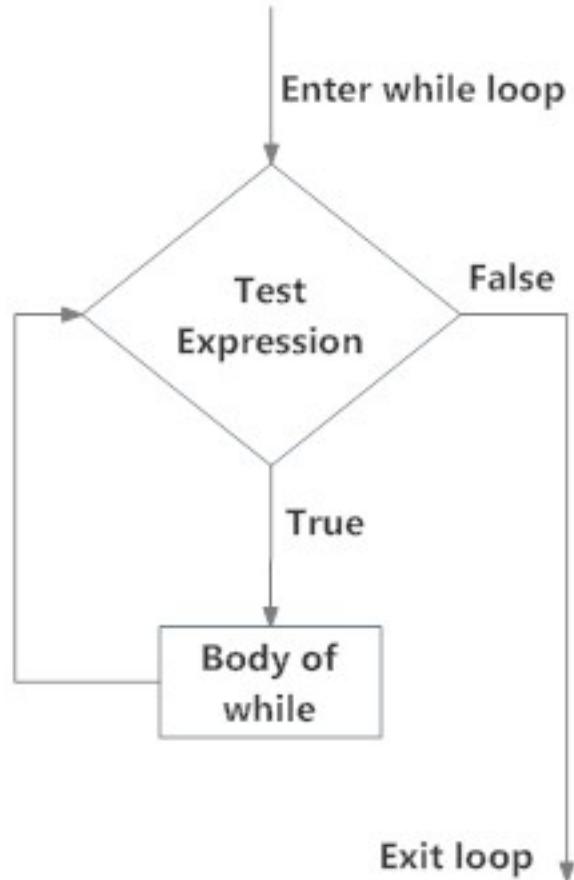


Fig: operation of while loop

**Don't forget the Colon.**

```

>>> def fib(n):      # write Fibonacci series up to n
...     """Print a Fibonacci series up to n."""
...     a, b = 0, 1
...     while a < n:
...         print(a, end=' ')
...         a, b = b, a+b
...     print()
...
>>> # Now call the function we just defined:
... fib(2000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
  
```

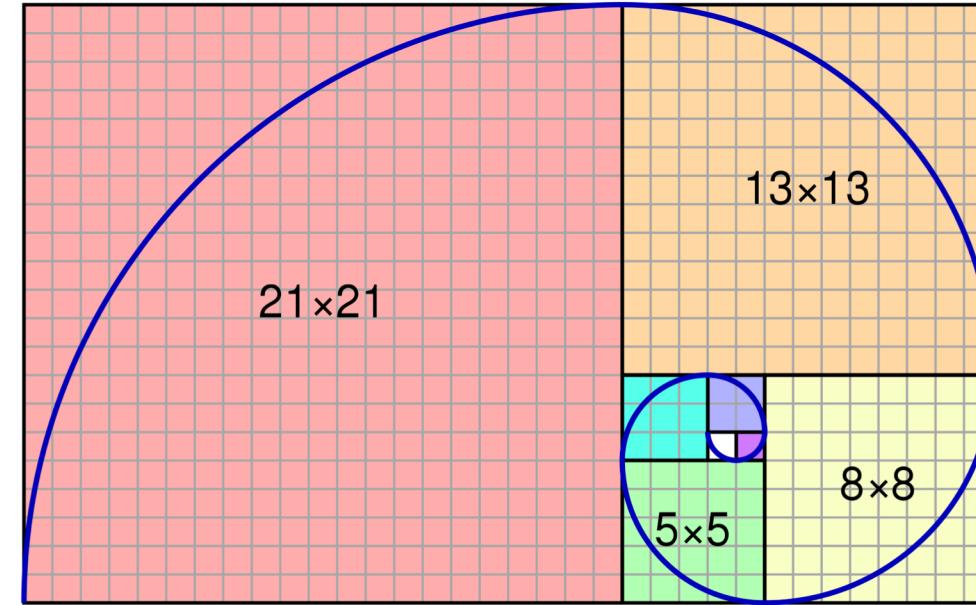
a) <https://www.datamentor.io>

# Programme Structure – Function

```
>>> def fib2(n): # return Fibonacci series up to n
...     """Return a list containing the Fibonacci series up to n."""
...     result = []
...     a, b = 0, 1
...     while a < n:
...         result.append(a)    # see below
...         a, b = b, a+b
...     return result
...
>>> f100 = fib2(100)    # call it
>>> f100                # write the result
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

The keyword *def* introduces a function definition. It must be followed by the function name and the parenthesized list of formal parameters. The statements that form the body of the function start at the next line, and must be indented.

**Don't forget the Colon.**

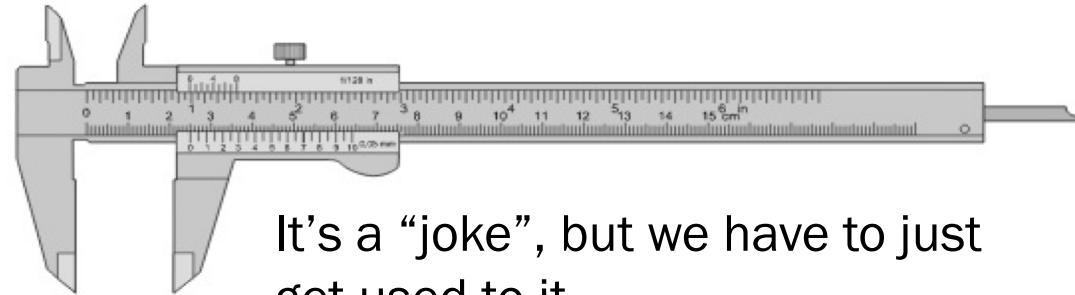


The Fibonacci spiral: an approximation of the golden spiral created by drawing circular arcs connecting the opposite corners of squares in the Fibonacci tiling



# Python Code Block – Indentation

- *if-elif-else* block
- *while* loop block
- *for* loop block
- function *def* block
- PEP 8 -- Style Guide for Python Code
  - PyCharm enforce this style and check your code again the rules by default.
- It includes Code layout, comment etc.
- It helps to improve the readability.



It's a “joke”, but we have to just get used to it.

a) <https://www.python.org/dev/peps/pep-0008/>

- [Code Lay-out](#)
  - [Indentation](#)
  - [Tabs or Spaces?](#)
  - [Maximum Line Length](#)
  - [Should a Line Break Before or After a Binary Operator?](#)
  - [Blank Lines](#)
  - [Source File Encoding](#)
  - [Imports](#)
  - [Module Level Dunder Names](#)
- [String Quotes](#)



# Project Structure – Module and Import

- If you quit from the Python interpreter and enter it again, the definitions you have made (functions and variables) are lost.
  - Use a text editor to prepare the input for the interpreter and running it with that file as input instead. This is known as creating a *script*.
  - Split it into several files for easier maintenance.
  - Reuse a handy function that you've written in several programs without copying its definition into each program.

```
# Fibonacci numbers module          fibo.py
def fib(n):      # write Fibonacci series up to n
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

def fib2(n):     # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result
```

```
>>> import fibo
>>> fibo.fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
>>> fibo.fib2(100)
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibo.__name__
'fibo'
>>> fib = fibo.fib
>>> fib(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

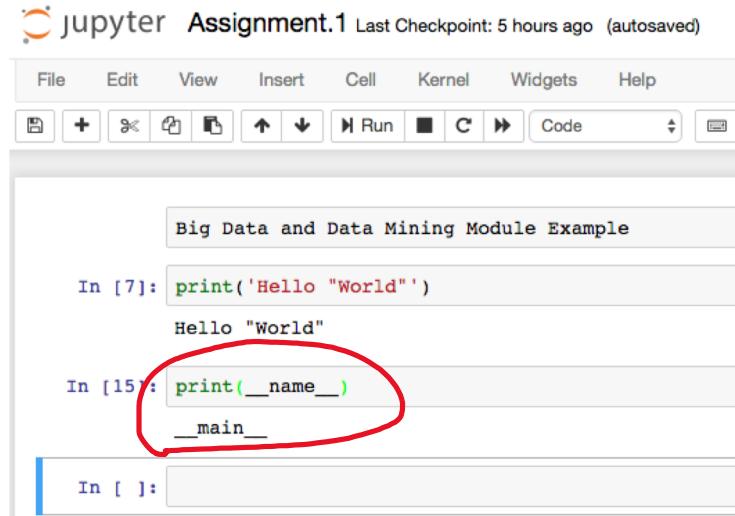
```
>>> from fibo import fib, fib2
>>> fib(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
>>> from fibo import *
>>> fib(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
>>> import fibo as fib
>>> fib.fib(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
>>> from fibo import fib as fibonacci
>>> fibonacci(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

# Running your module as a script

```
>>> import fibo

>>> fibo.fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fibo.fib2(100)
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibo.__name__
'fibo'

>>> fib = fibo.fib
>>> fib(500)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```



```
if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

Run as a script

Import as a module

```
$ python fibo.py 50
0 1 1 2 3 5 8 13 21 34
```

```
>>> import fibo
>>>
```



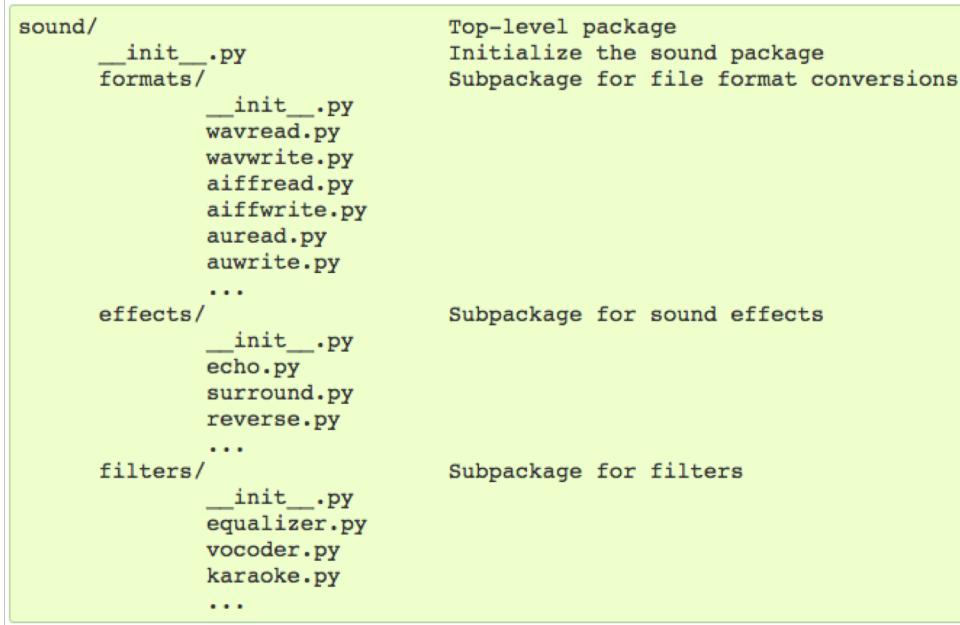
# Project Structure – Where is my module

- The Module Search Path
  - When a module named spam is imported, the interpreter first searches for a built-in module with that name.
  - If not found, it then searches for a file named xxxx.py in a list of directories given by the variable [sys.path](#).

```
In [24]: import sys
for x in sys.path:
    print(x)
    '''
/Users/jjdengjjj/anaconda3/lib/python36.zip
/Users/jjdengjjj/anaconda3/lib/python3.6
/Users/jjdengjjj/anaconda3/lib/python3.6/lib-dynload
/Users/jjdengjjj/anaconda3/lib/python3.6/site-packages
/Users/jjdengjjj/anaconda3/lib/python3.6/site-packages/aeosa
/Users/jjdengjjj/anaconda3/lib/python3.6/site-packages/IPython/extensions
/Users/jjdengjjj/.ipython
```



# Project Structure – Package or Library



- Import individual modules from the package
  - `import sound.effects.echo`
  - `from sound.effects import echo`
  - `echo.echofilter(input, output, delay=0.7, atten=4)`
- Import individual function from the module
  - `from sound.effects.echo import echofilter`
  - `echofilter(input, output, delay=0.7, atten=4)`
- Intra-package references
  - `from . import echo`
  - `from .. import formats`
  - `from ..filters import equalizer`



# Object Oriented Programming

```
In [26]: class Person:

    def __init__(self, first, last):
        self.firstname = first
        self.lastname = last

    def Name(self):
        return self.firstname + " " + self.lastname

class Employee(Person):

    def __init__(self, first, last, staffnum):
        Person.__init__(self,first, last)
        self.staffnumber = staffnum

    def GetEmployee(self):
        return self.Name() + ", " + self.staffnumber

x = Person("Marge", "Simpson")
y = Employee("Homer", "Simpson", "1007")

print(x.Name())
print(y.GetEmployee())

Marge Simpson
Homer Simpson, 1007
```

- Class Definition
- Initialisation Function
- Member Variable
- Member Function
- Inheritance
  - Initialise base class
- Object Instantiation
- Object Usage



# Standard Library – Basic File I/O

```
>>> f = open('workfile', 'w')
```

```
>>> f.close()
>>> f.read()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: I/O operation on closed file.
```



Always forget to close your file?  
Python has a good solution!



```
>>> with open('workfile') as f:
...     read_data = f.read()
>>> f.closed
True
```

**file.read([size])**

Reads at most size bytes from the file (less if the read hits EOF before obtaining size bytes).

**file.readline([size])**

Reads one entire line from the file. A trailing newline character is kept in the string.

**file.write(str)**

Writes a string to the file. There is no return value.

**file.writelines(sequence)**

Writes a sequence of strings to the file. The sequence can be any iterable object producing strings, typically a list of strings.

**file.seek(offset[, whence])**

Sets the file's current position

**file.flush()**

Flush the internal buffer, like stdio's fflush. This may be a no-op on some file-like objects.



# Standard Library – Built-in Function

| Built-in Functions         |                          |                           |                         |                             |
|----------------------------|--------------------------|---------------------------|-------------------------|-----------------------------|
| <code>abs()</code>         | <code>dict()</code>      | <code>help()</code>       | <code>min()</code>      | <code>setattr()</code>      |
| <code>all()</code>         | <code>dir()</code>       | <code>hex()</code>        | <code>next()</code>     | <code>slice()</code>        |
| <code>any()</code>         | <code>divmod()</code>    | <code>id()</code>         | <code>object()</code>   | <code>sorted()</code>       |
| <code>ascii()</code>       | <code>enumerate()</code> | <code>input()</code>      | <code>oct()</code>      | <code>staticmethod()</code> |
| <code>bin()</code>         | <code>eval()</code>      | <code>int()</code>        | <code>open()</code>     | <code>str()</code>          |
| <code>bool()</code>        | <code>exec()</code>      | <code>isinstance()</code> | <code>ord()</code>      | <code>sum()</code>          |
| <code>bytearray()</code>   | <code>filter()</code>    | <code>issubclass()</code> | <code>pow()</code>      | <code>super()</code>        |
| <code>bytes()</code>       | <code>float()</code>     | <code>iter()</code>       | <code>print()</code>    | <code>tuple()</code>        |
| <code>callable()</code>    | <code>format()</code>    | <code>len()</code>        | <code>property()</code> | <code>type()</code>         |
| <code>chr()</code>         | <code>frozenset()</code> | <code>list()</code>       | <code>range()</code>    | <code>vars()</code>         |
| <code>classmethod()</code> | <code>getattr()</code>   | <code>locals()</code>     | <code>repr()</code>     | <code>zip()</code>          |
| <code>compile()</code>     | <code>globals()</code>   | <code>map()</code>        | <code>reversed()</code> | <code>__import__()</code>   |
| <code>complex()</code>     | <code>hasattr()</code>   | <code>max()</code>        | <code>round()</code>    |                             |
| <code>delattr()</code>     | <code>hash()</code>      | <code>memoryview()</code> | <code>set()</code>      |                             |



# Standard Library – Overview

## 10. Brief Tour of the Standard Library

- 10.1. Operating System Interface
- 10.2. File Wildcards
- 10.3. Command Line Arguments
- 10.4. Error Output Redirection and Program Termination
- 10.5. String Pattern Matching
- 10.6. Mathematics
- 10.7. Internet Access
- 10.8. Dates and Times
- 10.9. Data Compression
- 10.10. Performance Measurement
- 10.11. Quality Control
- 10.12. Batteries Included

## 11. Brief Tour of the Standard Library — Part II

- 11.1. Output Formatting
- 11.2. Templating
- 11.3. Working with Binary Data Record Layouts
- 11.4. Multi-threading
- 11.5. Logging
- 11.6. Weak References
- 11.7. Tools for Working with Lists
- 11.8. Decimal Floating Point Arithmetic

Note: Always refer to the official documents, we cannot remember them all.

a) <https://docs.python.org/3.7/tutorial/index.html>



# External Library – For Data Mining

- *NumPy* and *SciPy*
  - Numerical computation
- *Matplotlib* and *OpenCV*
  - Data visualisation and computer vision
- *pandas* and *scikit-learn*
  - Data pre-process and traditional machine learning
- *BeautifulSoup*
  - Data scrapping from internet
- *Tensorflow* and *Keras*
  - Deep Neural Network

