Python Introduction

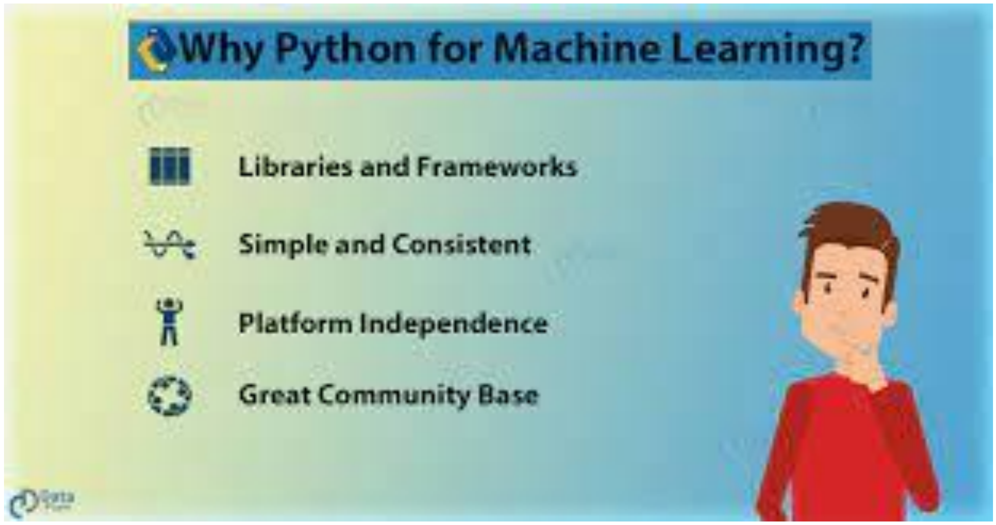
Deep Learning (DS-5006)
Dr. Adeel Mumtaz
Lec 2
Fall, 2022

National University
Of Computer and Emerging Sciences

Contents

Introduction (Python)

- https://www.python.org
- “Monty Python's Flying Circus”, a BBC comedy series from the 1970s
- Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands in December 1989
  - Python 2.0 was released on 16 October 2000
  - Python 3.0 released on 3 December 2008
  - Latest version Python 3.11
- Applications
  - Web and Internet Development
  - Database Access
  - Desktop GUIs
  - Scientific & Numeric
  - Education
  - Network Programming
  - Software & Game Development
- Fast, open and runs everywhere
- https://docs.python.org/3/
  - Python standard library



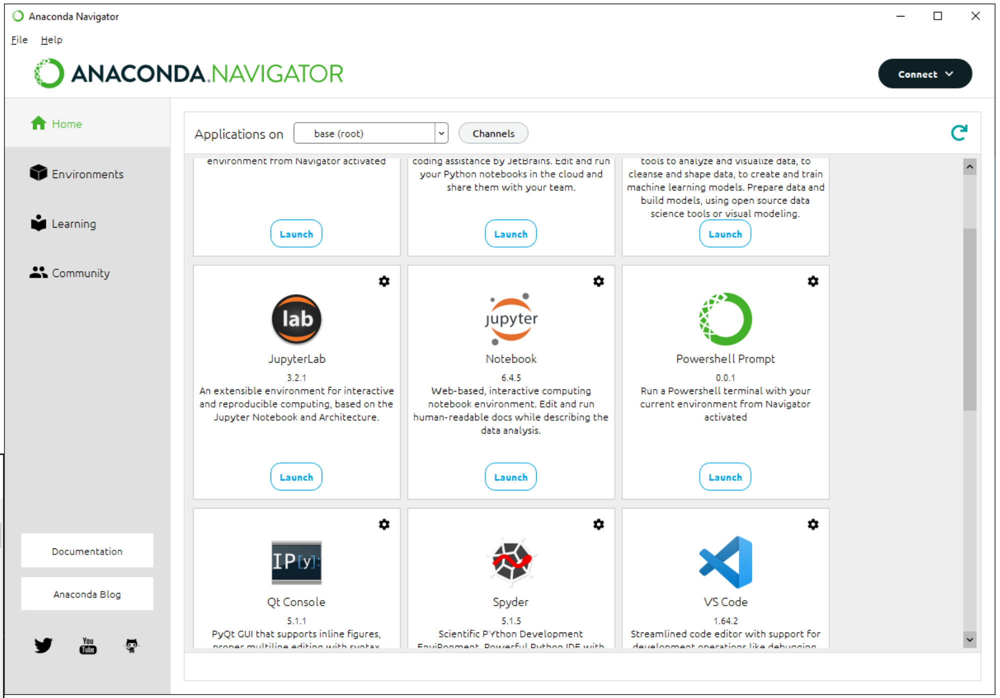
Why Python for Machine Learning?

- Libraries and Frameworks
- Simple and Consistent
- Platform Independence
- Great Community Base

PYTHON ENVIRONMENT

Anaconda

- Anaconda offers the easiest way to perform Python data science and machine learning
- Allows working with thousands of open-source packages and libraries.



environment from Navigator activated

coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

JupyterLab
3.2.1
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

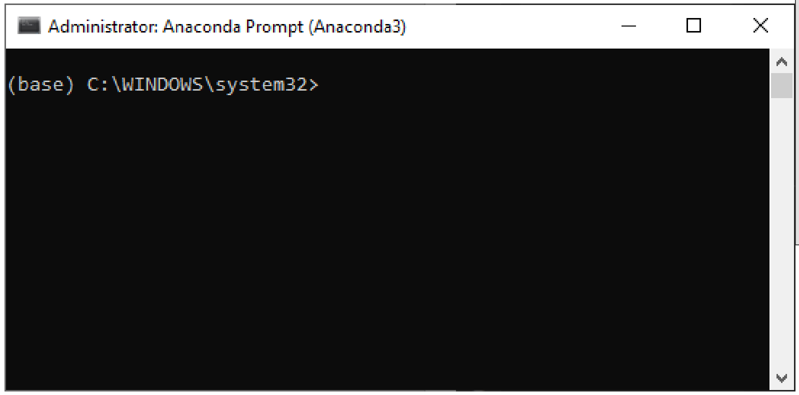
Notebook
6.4.5
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

Qt Console
5.1.1
PyQt GUI that supports inline figures, proper multiline editing with syntax

Spyder
5.1.5
Scientific PYthon Development EnviRonment. Powerful Python IDE with

VS Code
1.64.2
Streamlined code editor with support for development operations like debugging

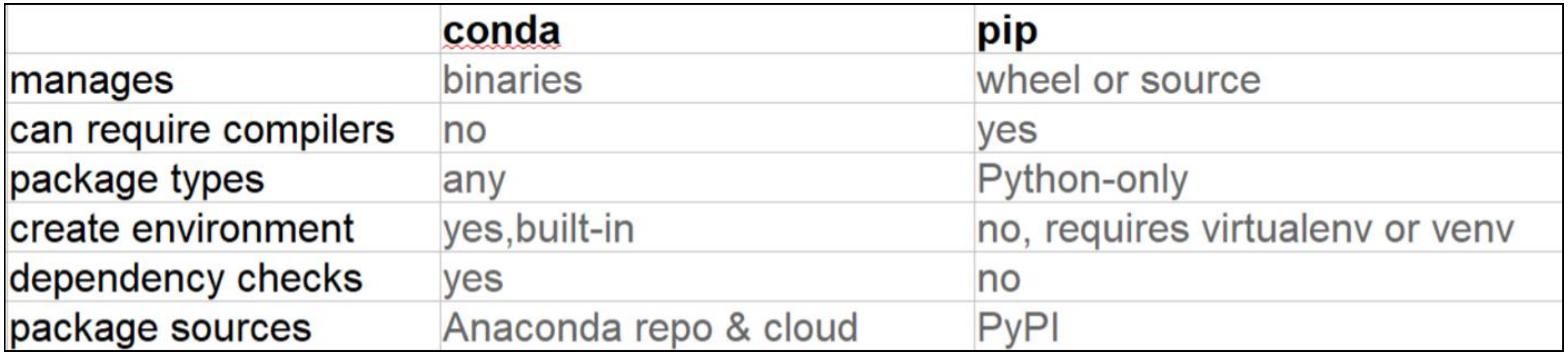


Administrator: Anaconda Prompt (Anaconda3)

```
(base) C:\WINDOWS\system32>
```

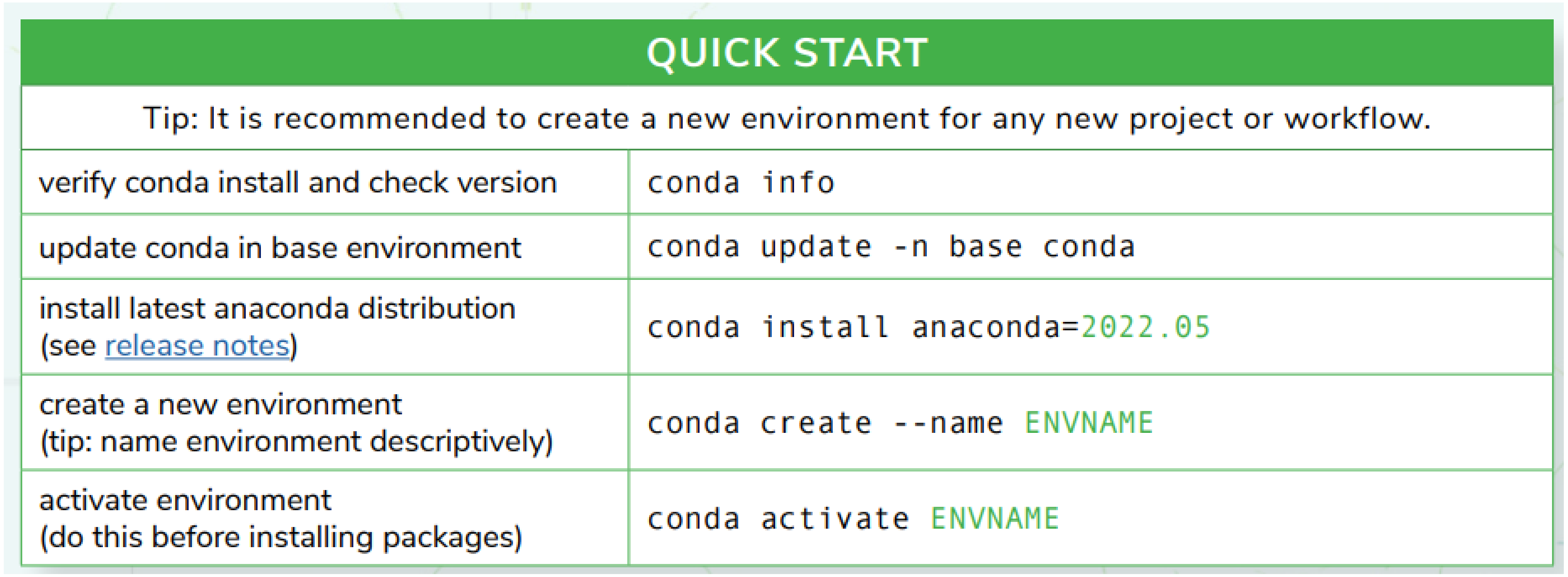
Conda

- Conda is a package manager
- Must know conda/pip commands to create a DL environment



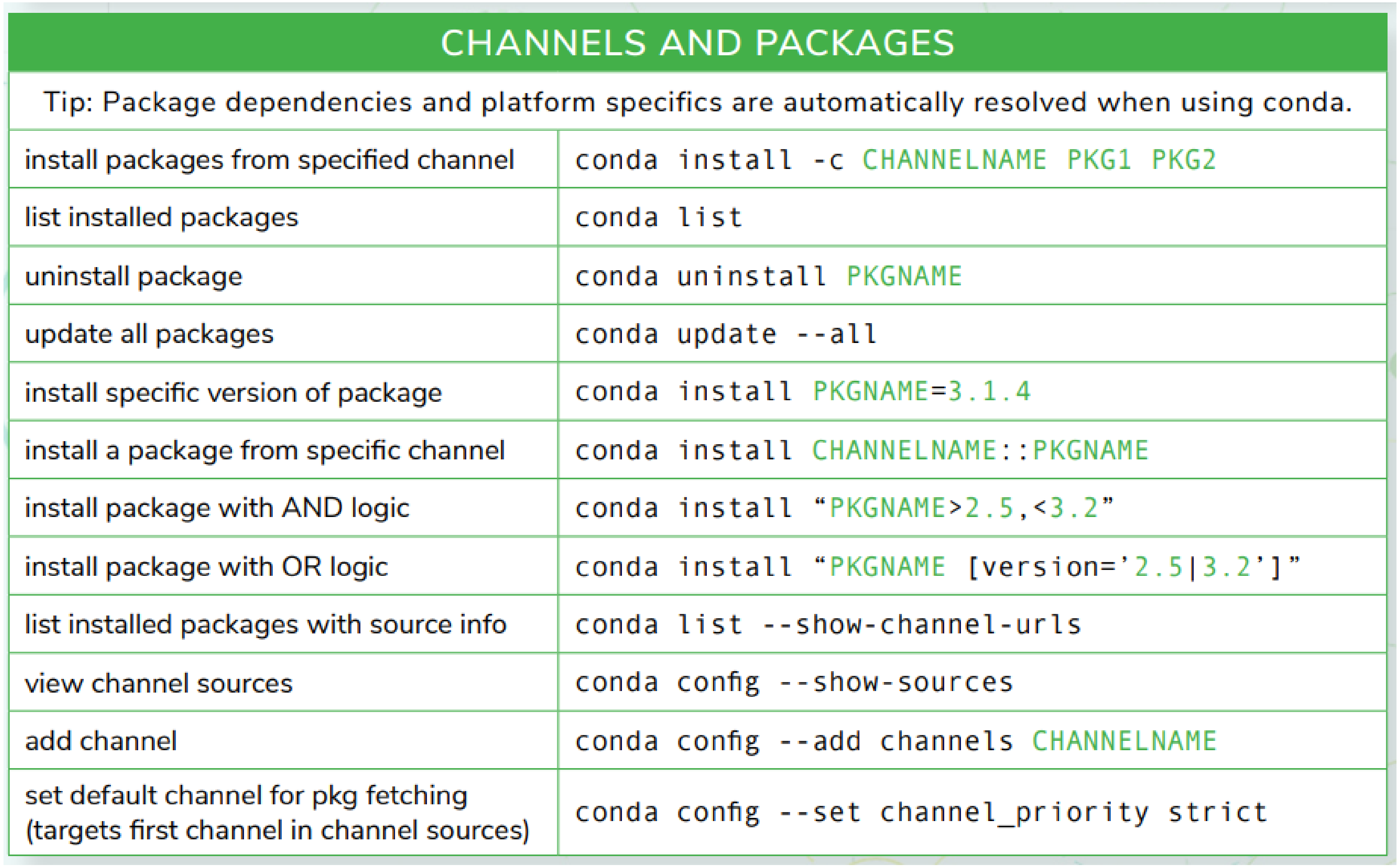
	conda	pip
manages	binaries	wheel or source
can require compilers	no	yes
package types	any	Python-only
create environment	yes,built-in	no, requires virtualenv or venv
dependency checks	yes	no
package sources	Anaconda repo & cloud	PyPI

Conda



QUICK START	
Tip: It is recommended to create a new environment for any new project or workflow.	
verify conda install and check version	`conda info`
update conda in base environment	`conda update -n base conda`
install latest anaconda distribution (see release notes)	`conda install anaconda=2022.05`
create a new environment (tip: name environment descriptively)	`conda create --name ENVNAME`
activate environment (do this before installing packages)	`conda activate ENVNAME`

Conda

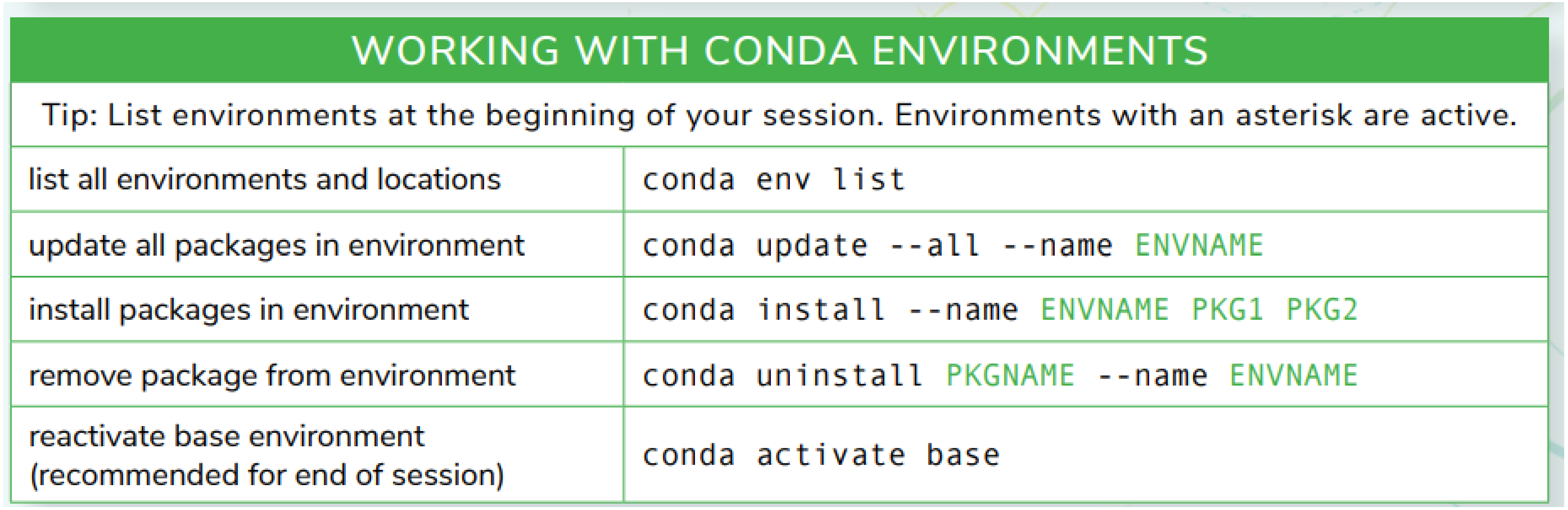


CHANNELS AND PACKAGES

Tip: Package dependencies and platform specifics are automatically resolved when using conda.

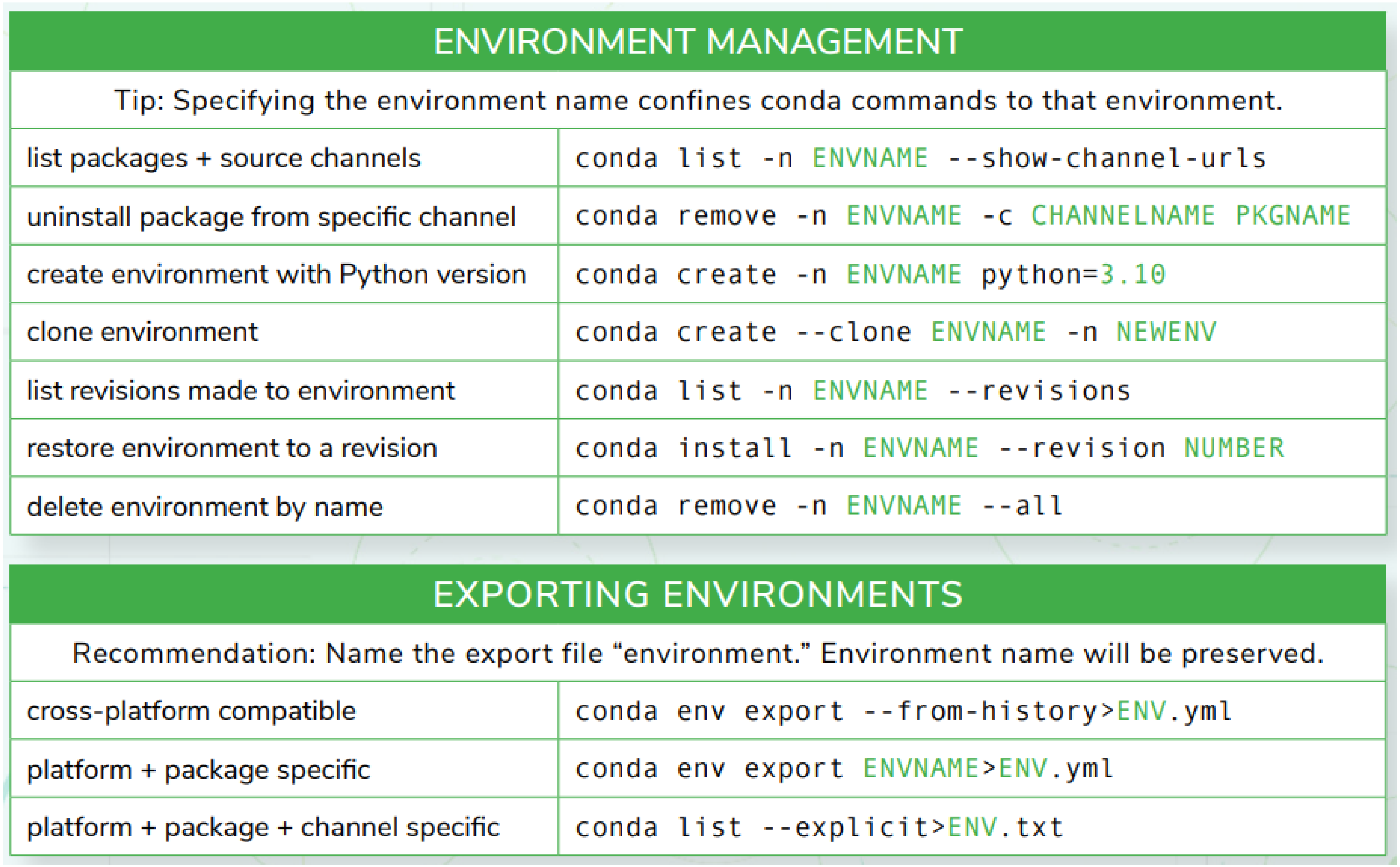
install packages from specified channel	`conda install -c CHANNELNAME PKG1 PKG2`
list installed packages	`conda list`
uninstall package	`conda uninstall PKGNAME`
update all packages	`conda update --all`
install specific version of package	`conda install PKGNAME=3.1.4`
install a package from specific channel	`conda install CHANNELNAME::PKGNAME`
install package with AND logic	`conda install “PKGNAME>2.5,<3.2”`
install package with OR logic	`conda install “PKGNAME [version=’2.5|3.2’]”`
list installed packages with source info	`conda list --show-channel-urls`
view channel sources	`conda config --show-sources`
add channel	`conda config --add channels CHANNELNAME`
set default channel for pkg fetching (targets first channel in channel sources)	`conda config --set channel_priority strict`

Conda



WORKING WITH CONDA ENVIRONMENTS	
Tip: List environments at the beginning of your session. Environments with an asterisk are active.	
list all environments and locations	`conda env list`
update all packages in environment	`conda update --all --name ENVNAME`
install packages in environment	`conda install --name ENVNAME PKG1 PKG2`
remove package from environment	`conda uninstall PKGNAME --name ENVNAME`
reactivate base environment (recommended for end of session)	`conda activate base`

Conda



ENVIRONMENT MANAGEMENT

Tip: Specifying the environment name confines conda commands to that environment.

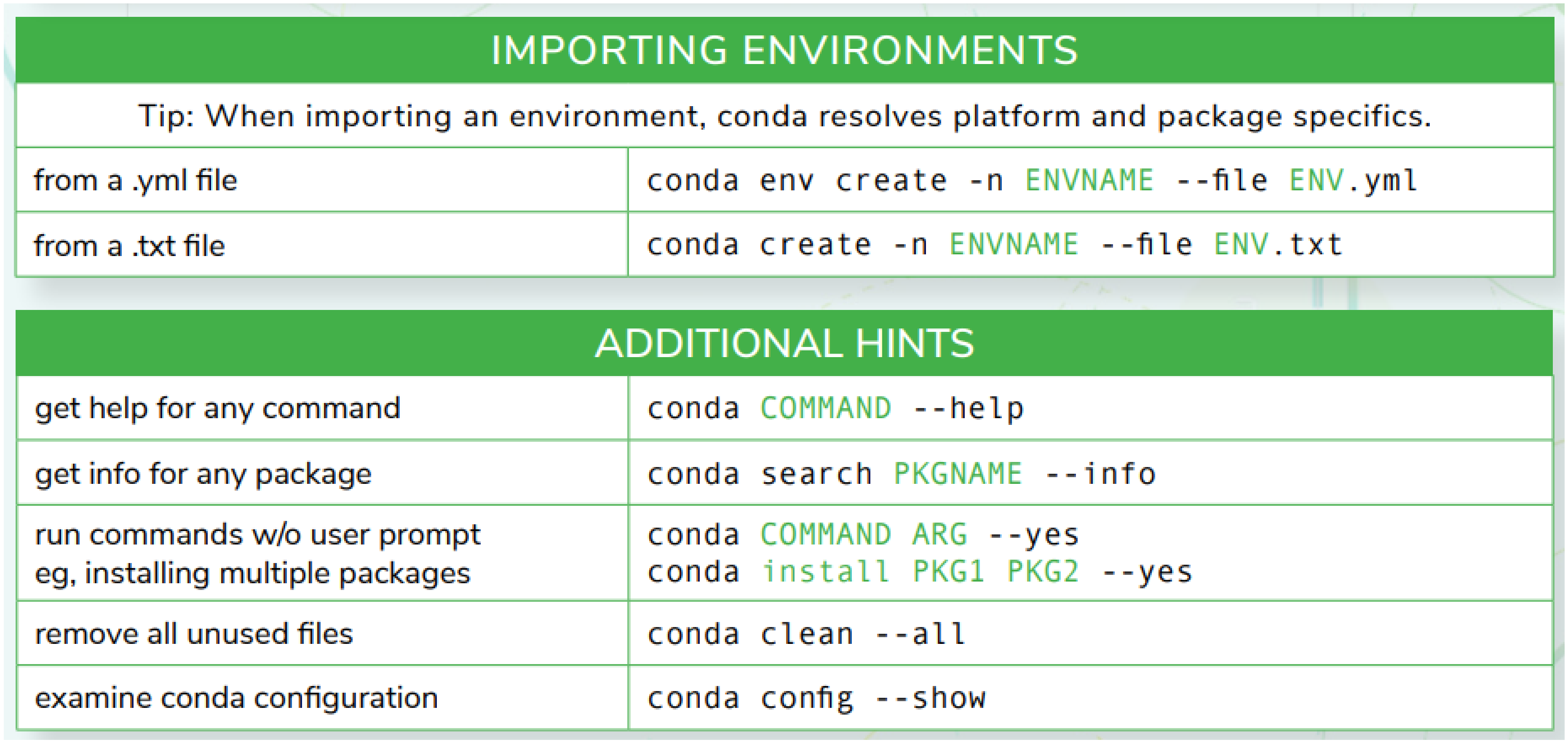
list packages + source channels	`conda list -n ENVNAME --show-channel-urls`
uninstall package from specific channel	`conda remove -n ENVNAME -c CHANNELNAME PKGNAME`
create environment with Python version	`conda create -n ENVNAME python=3.10`
clone environment	`conda create --clone ENVNAME -n NEWENV`
list revisions made to environment	`conda list -n ENVNAME --revisions`
restore environment to a revision	`conda install -n ENVNAME --revision NUMBER`
delete environment by name	`conda remove -n ENVNAME --all`

EXPORTING ENVIRONMENTS

Recommendation: Name the export file “environment.” Environment name will be preserved.

cross-platform compatible	`conda env export --from-history>ENV.yml`
platform + package specific	`conda env export ENVNAME>ENV.yml`
platform + package + channel specific	`conda list --explicit>ENV.txt`

Conda



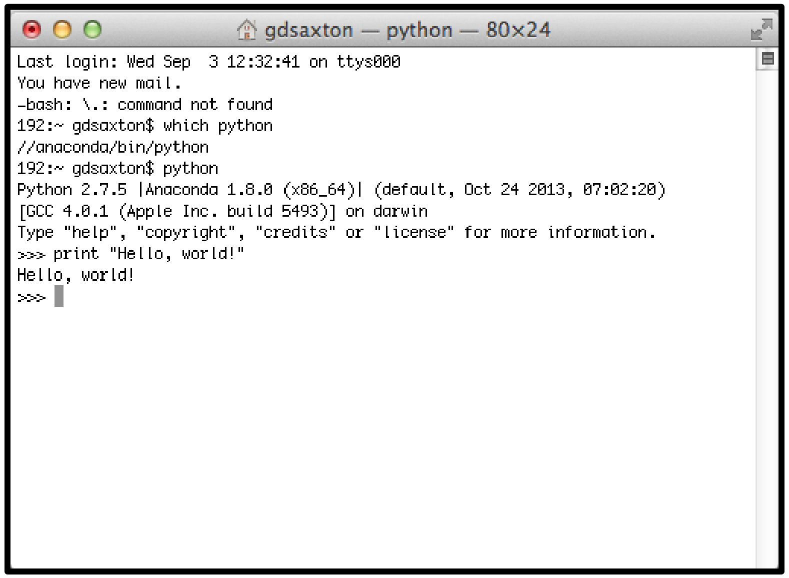
IMPORTING ENVIRONMENTS	
Tip: When importing an environment, conda resolves platform and package specifics.	
from a .yml file	`conda env create -n ENVNAME --file ENV.yml`
from a .txt file	`conda create -n ENVNAME --file ENV.txt`

ADDITIONAL HINTS	
get help for any command	`conda COMMAND --help`
get info for any package	`conda search PKGNAME --info`

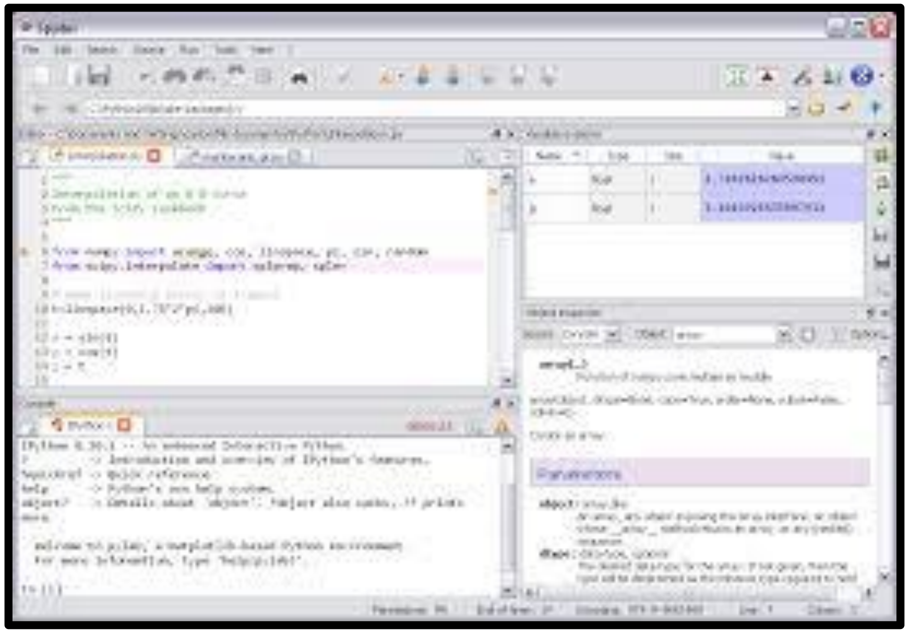
run commands w/o user prompt
eg, installing multiple packages
	`conda COMMAND ARG --yes`
	`conda install PKG1 PKG2 --yes`

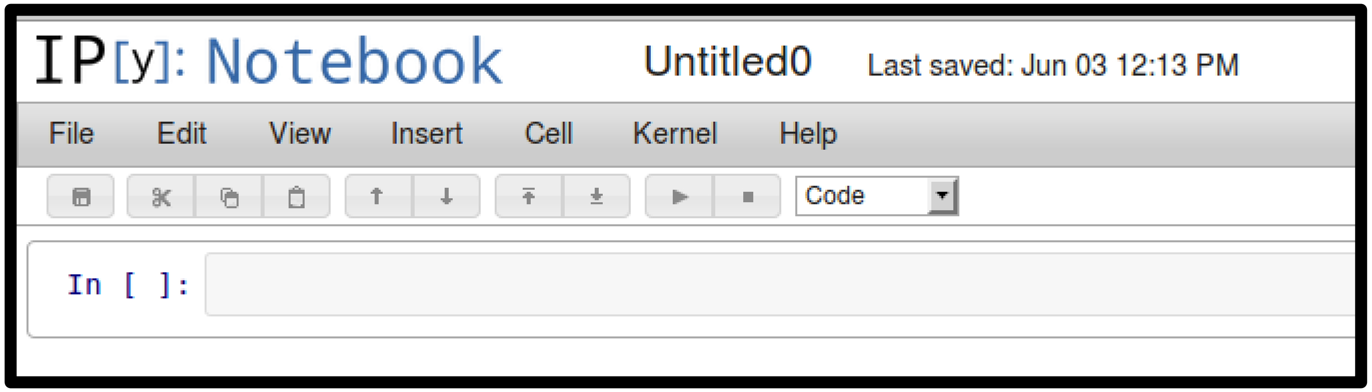
remove all unused files	`conda clean --all`
examine conda configuration	`conda config --show`

Different Ways to Run Python Code



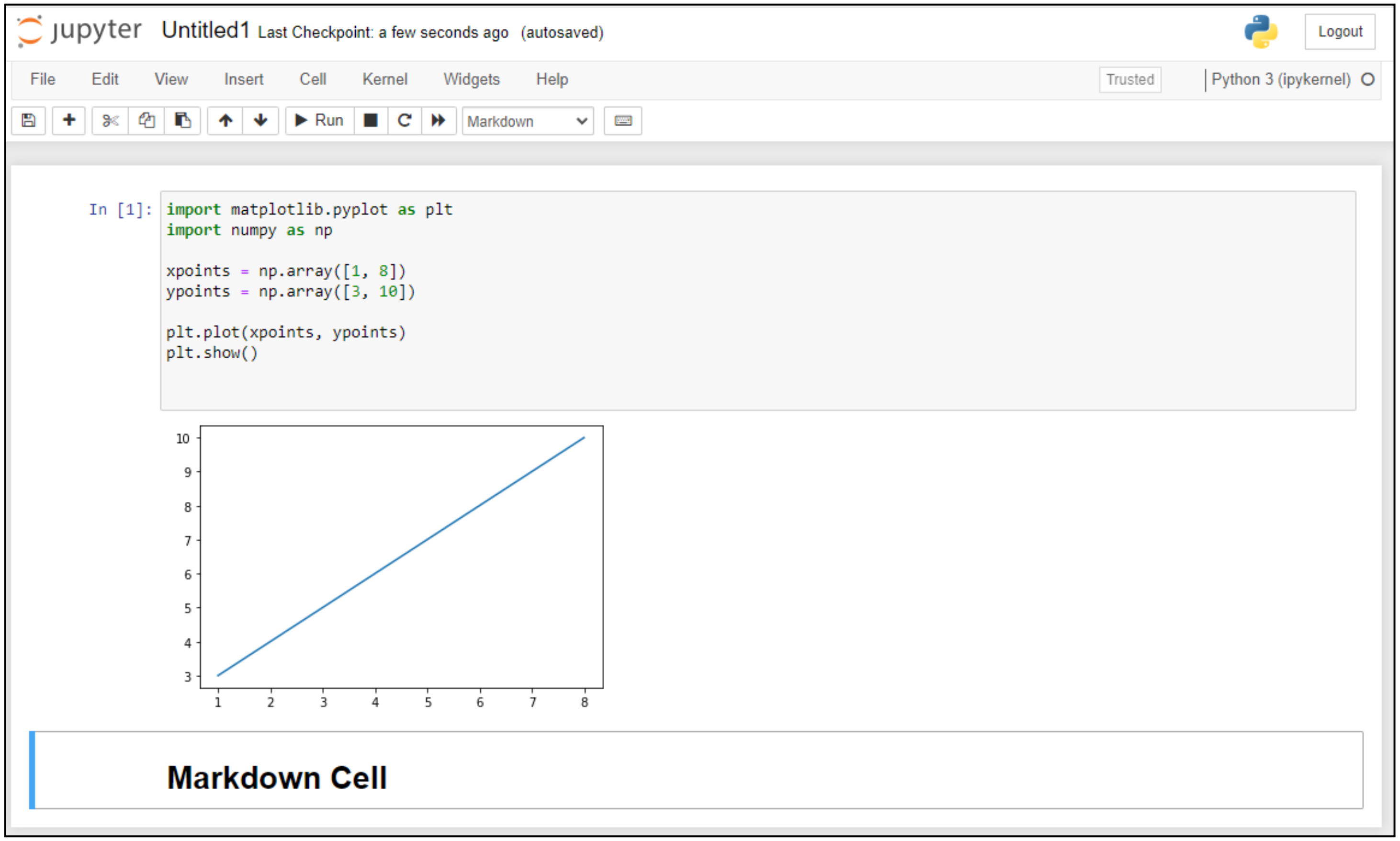
```
Last login: Wed Sep  3 12:32:41 on ttys000
You have new mail.
-bash: \.: command not found
192:~ gdsaxton$ which python
//anaconda/bin/python
192:~ gdsaxton$ python
Python 2.7.5 |Anaconda 1.8.0 (x86_64)| (default, Oct 24 2013, 07:02:20)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Hello, world!"
Hello, world!
>>> 
```





Jupyter Notebook

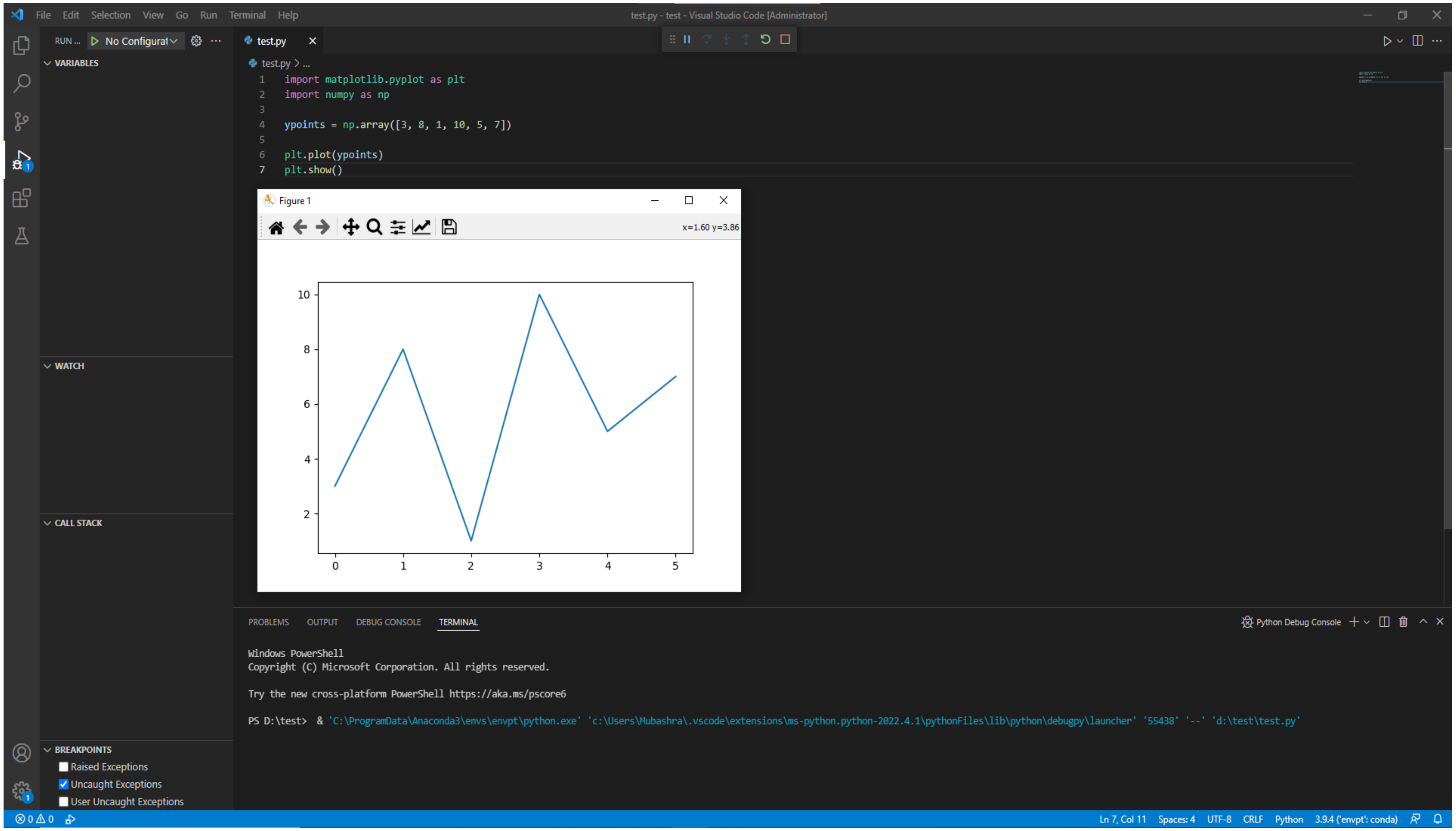
- How to start?
  - Command: jupyter notebook (start from a working directory)
  - Navigator



In [1]:
```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints)
plt.show()
```

Markdown Cell

VSCode

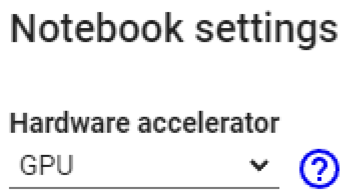


```python
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10, 5, 7])

plt.plot(ypoints)
plt.show()
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\test>  & 'C:\ProgramData\Anaconda3\envs\envpt\python.exe' 'c:\Users\Mubashra\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '55438' '--' 'd:\test\test.py'
```

Google Colaboratory

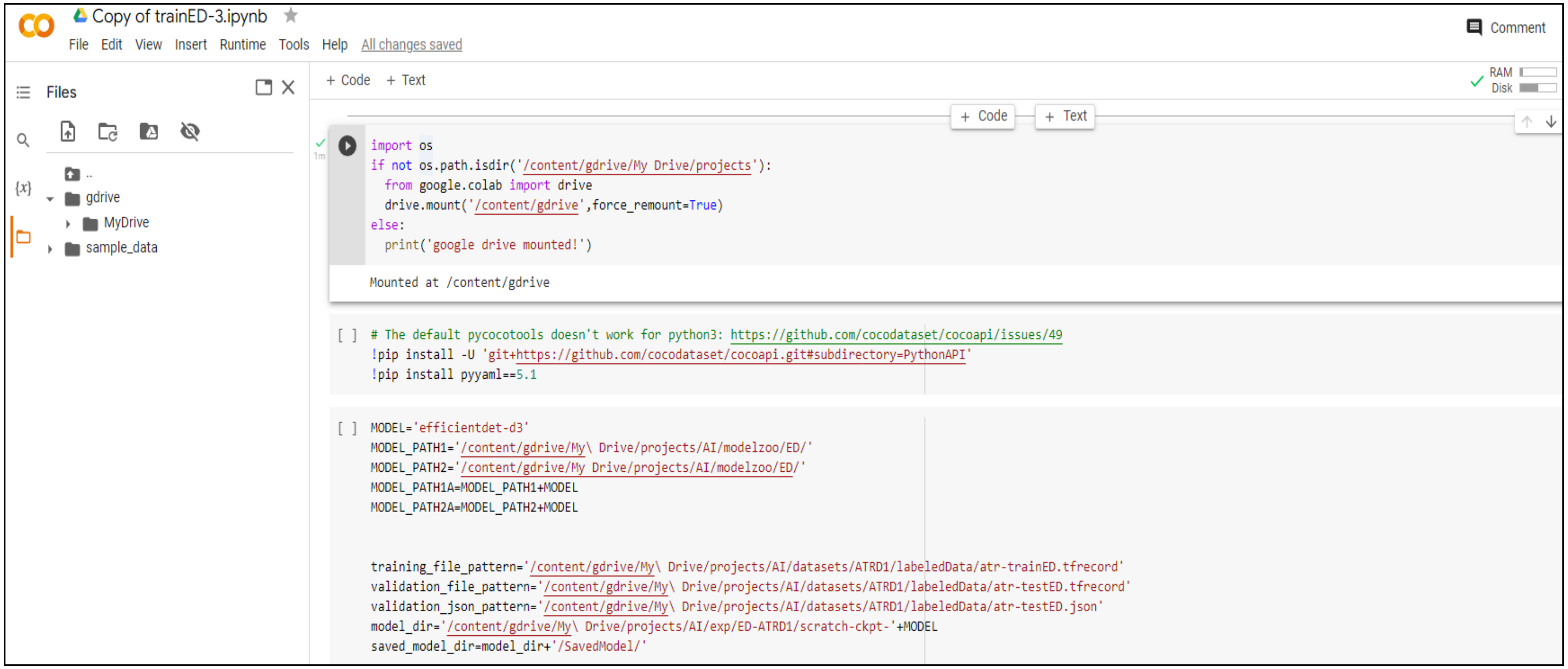
- Free Jupyter Notebook environment that runs in cloud
- Teams can work simultaneously
- Supports GPU
- Supports all common AI libraries
- Import/export google drive and GitHub
- https://colab.research.google.com/



Notebook settings

Hardware accelerator
GPU

Colab (Data + Models)

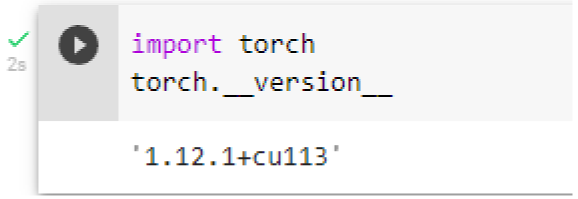


```
import os
if not os.path.isdir('/content/gdrive/My Drive/projects'):
  from google.colab import drive
  drive.mount('/content/gdrive',force_remount=True)
else:
  print('google drive mounted!')
```

```
Mounted at /content/gdrive
```

```
# The default pycocotools doesn't work for python3: https://github.com/cocodataset/cocoapi/issues/49
!pip install -U 'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'
!pip install pyyaml==5.1
```

```
MODEL='efficientdet-d3'
MODEL_PATH1='/content/gdrive/My\ Drive/projects/AI/modelzoo/ED/'
MODEL_PATH2='/content/gdrive/My Drive/projects/AI/modelzoo/ED/'
MODEL_PATH1A=MODEL_PATH1+MODEL
MODEL_PATH2A=MODEL_PATH2+MODEL


training_file_pattern='/content/gdrive/My\ Drive/projects/AI/datasets/ATRD1/labeledData/atr-trainED.tfrecord'
validation_file_pattern='/content/gdrive/My\ Drive/projects/AI/datasets/ATRD1/labeledData/atr-testED.tfrecord'
validation_json_pattern='/content/gdrive/My\ Drive/projects/AI/datasets/ATRD1/labeledData/atr-testED.json'
model_dir='/content/gdrive/My\ Drive/projects/AI/exp/ED-ATRD1/scratch-ckpt-'+MODEL
saved_model_dir=model_dir+'/SavedModel/'
```

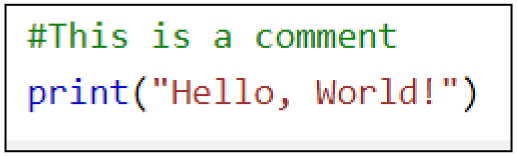


```
import torch
torch.__version__
```

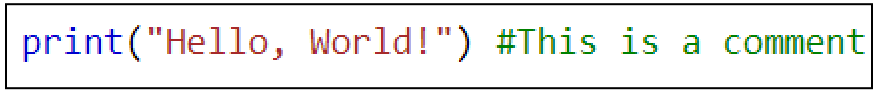
```
'1.12.1+cu113'
```

Python Comments

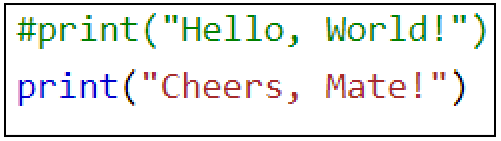
- Explain Python code
- Make the code more readable
- Prevent execution when testing code.



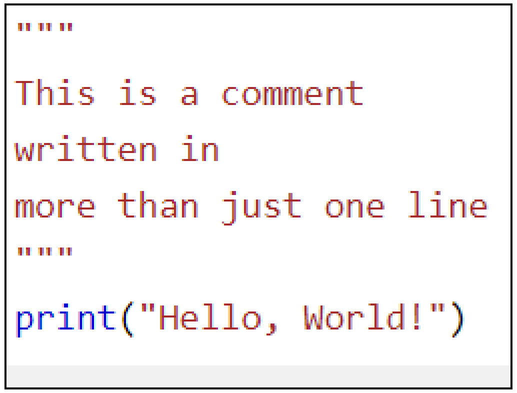
```
#This is a comment
print("Hello, World!")
```



```
print("Hello, World!") #This is a comment
```



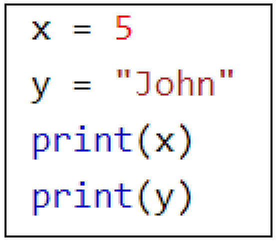
```
#print("Hello, World!")
print("Cheers, Mate!")
```



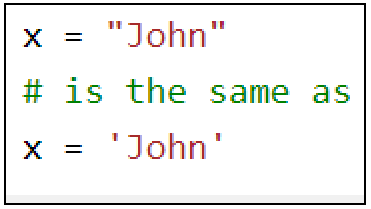
```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

Variables

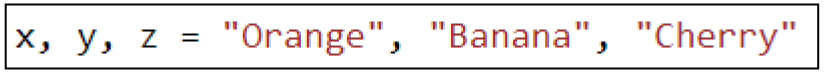
- Containers for storing data values
- Variables do not need to be declared with any particular type and can even change type after they have been set
- created the moment you first assign a value to it.



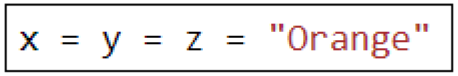
```python
x = 5
y = "John"
print(x)
print(y)
```



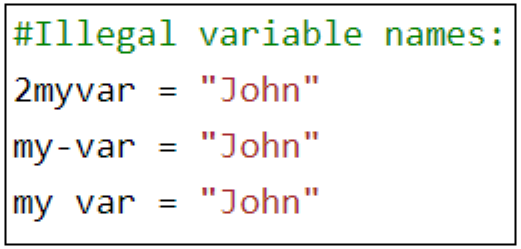
```python
x = "John"
# is the same as
x = 'John'
```



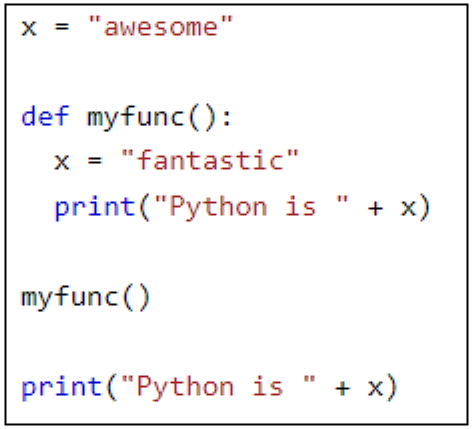
```python
x, y, z = "Orange", "Banana", "Cherry"
```



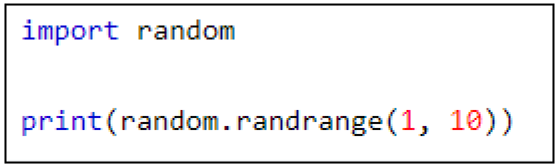
```python
x = y = z = "Orange"
```



```python
#Illegal variable names:
2myvar = "John"
my-var = "John"
my var = "John"
```



```python
x = "awesome"

def myfunc():
  x = "fantastic"
  print("Python is " + x)

myfunc()

print("Python is " + x)
```



```python
import random

print(random.randrange(1, 10))
```

Data Types

Python has the following data types built-in by default, in these categories:

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`

```
x = str("Hello World")
```

```
x = int(20)
```

```
x = float(20.5)
```

```
x = 5
print(type(x))
```

Strings

`'hello'` is the same as `"hello"`.

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

```
a = "Hello, World!"
print(a[1])
```

```
for x in "banana":
  print(x)
```

```
a = "Hello, World!"
print(len(a))
```

```
a = " Hello, World! "
print(a.strip()) # returns "Hello, World!"
```

```
txt = "The best things in life are free!"
print("free" in txt)
```

```
b = "Hello, World!"
print(b[2:5])
```

```
b = "Hello, World!"
print(b[:5])
```

```
a = "Hello, World!"
print(a.replace("H", "J"))
```

```
b = "Hello, World!"
print(b[2:])
```

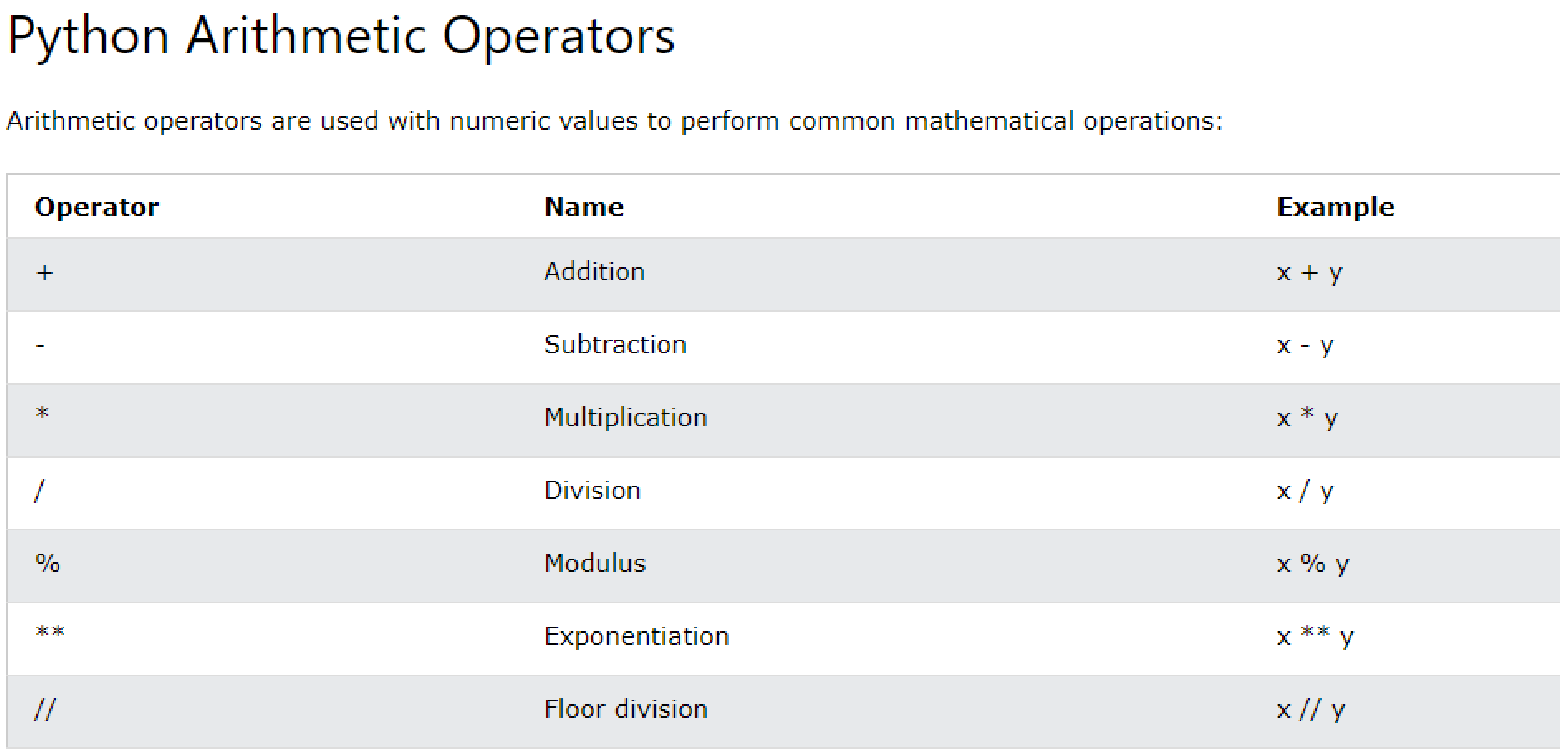
```
b = "Hello, World!"
print(b[-5:-2])
```

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

```
a = "Hello, World!"
print(a.split(",")) # returns ['Hello', ' World!']
```

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

Operators

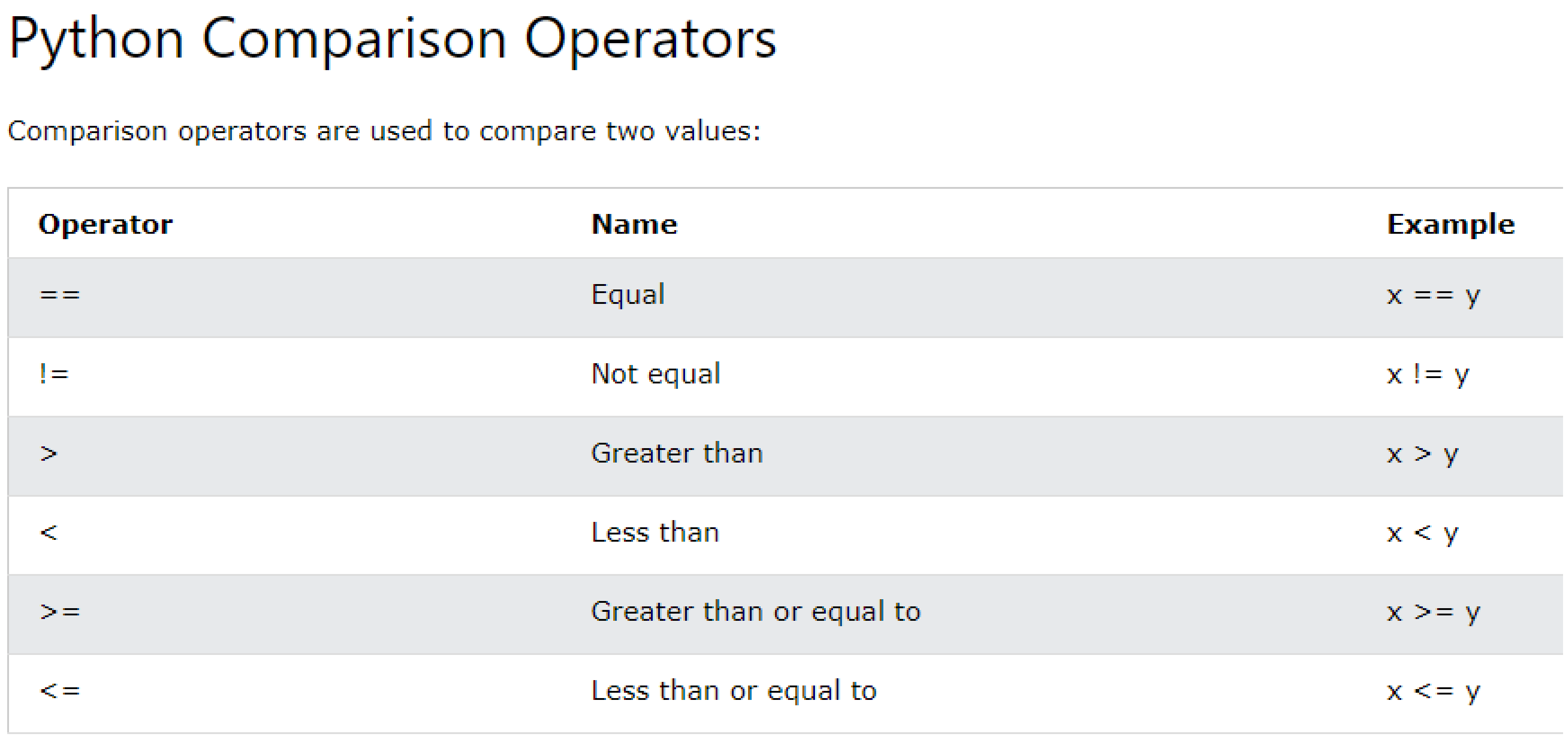


Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Operators

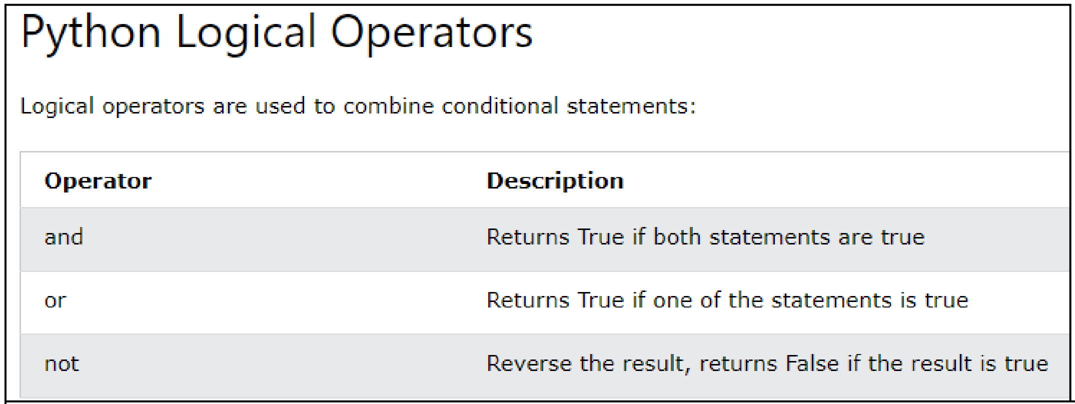


Python Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

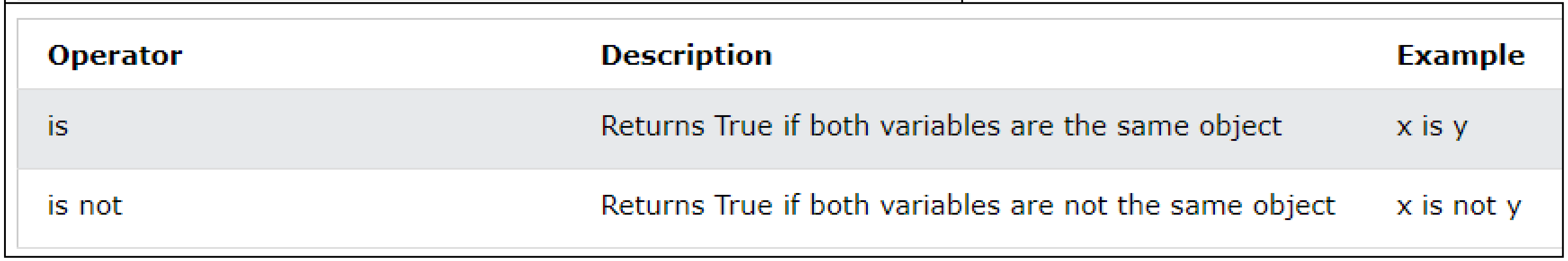
Operators



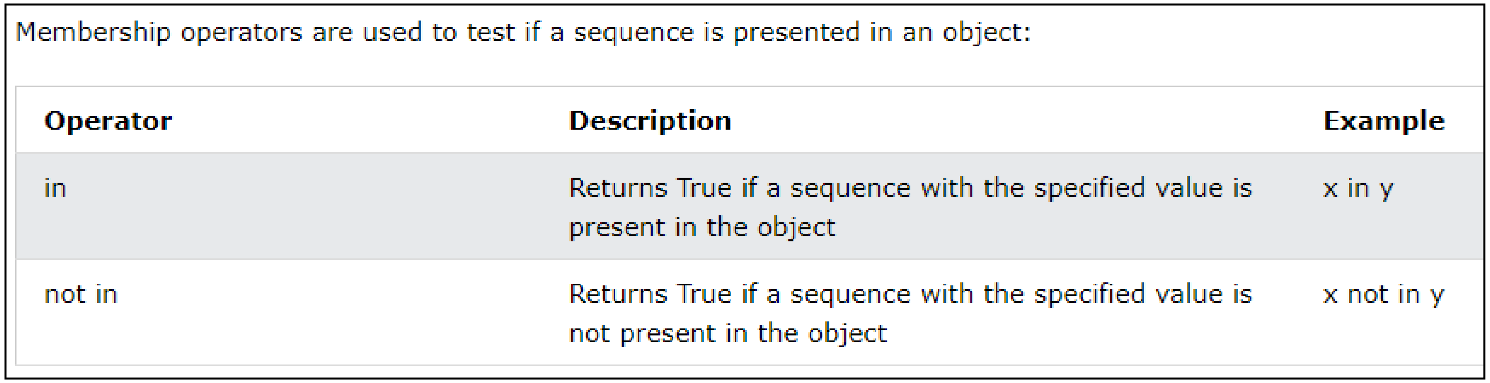
Python Logical Operators

Logical operators are used to combine conditional statements:

Operator	Description
and	Returns True if both statements are true
or	Returns True if one of the statements is true
not	Reverse the result, returns False if the result is true



Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y



Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Collections (List, Tuple, Set, Dictionary)

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

```
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
```

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

```
mytuple = ("apple", "banana", "cherry")
myit = iter(mytuple)

print(next(myit))
print(next(myit))
```

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon",
"mango"]
print(thislist[-4:-1])
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

```
list1 = ["abc", 34, True, 40, "male"]
```

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[-4:-1])
```

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
```

append, insert, extend,sort

```
x = thisdict["model"]
```

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```

```
for x, y in thisdict.items():
  print(x, y)
```

Conditionals & Loops (for, while, with)

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

```
for x in range(2, 30, 3):
  print(x)
```

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

Functions

```python
def my_function(x):
  return 5 * x
```

```python
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil", "Refsnes")
```

```python
def my_function(child3, child2, child1):
  print("The youngest child is " + child3)

my_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")
```

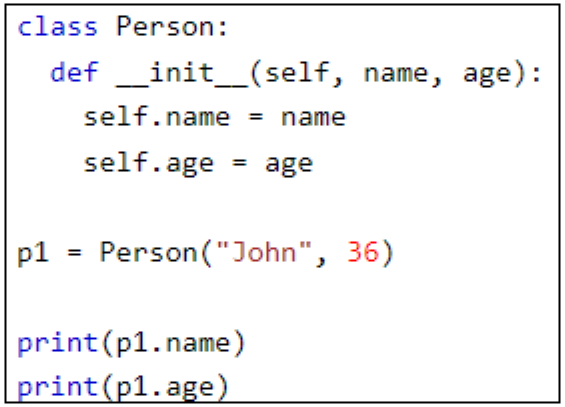
```python
def my_function(**kid):
  print("His last name is " + kid["lname"])

my_function(fname = "Tobias", lname = "Refsnes")
```

```python
def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Emil", "Tobias", "Linus")
```

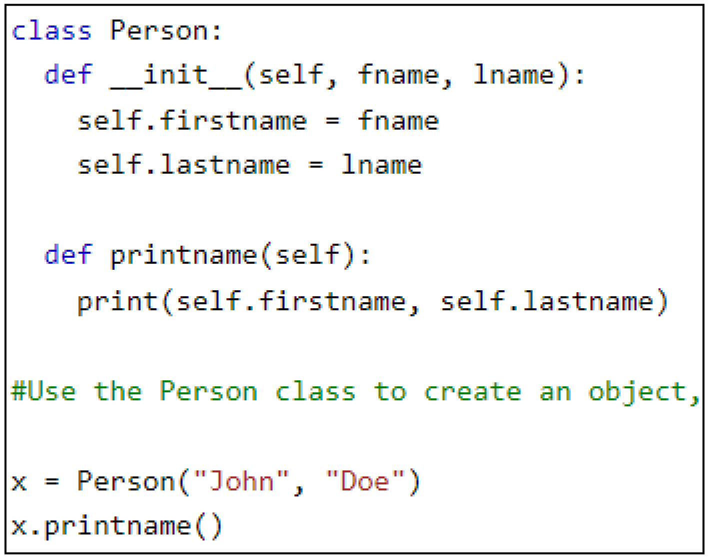
```python
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

```python
def my_function(food):
  food[0]=100

fruits = ["apple", "banana", "cherry"]

my_function(fruits)

print(fruits)
```

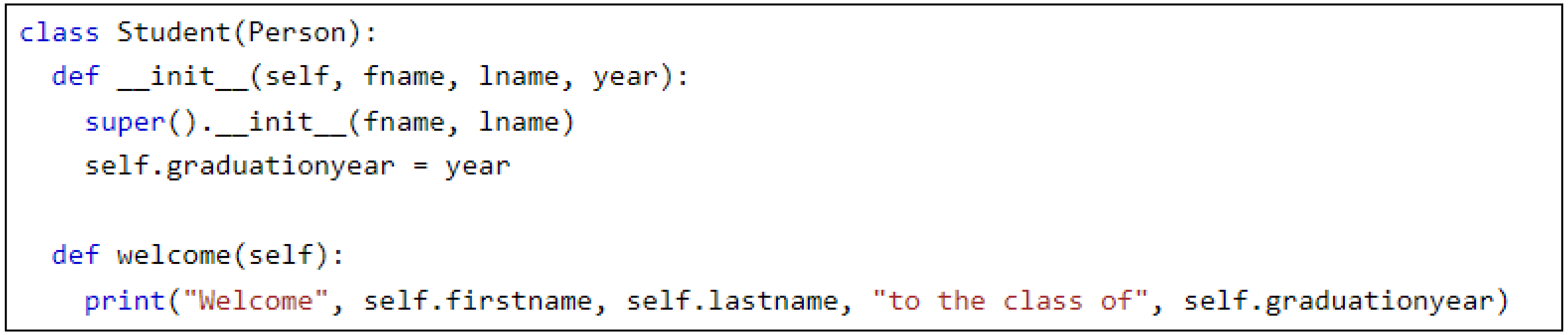
Classes & Inheritance



```python
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```



```python
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

#Use the Person class to create an object,

x = Person("John", "Doe")
x.printname()
```



```python
class Student(Person):
  def __init__(self, fname, lname, year):
    super().__init__(fname, lname)
    self.graduationyear = year

  def welcome(self):
    print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)
```

Modules, Packages and Libraries

The module named `mymodule` has one function and one dictionary:

```
def greeting(name):
  print("Hello, " + name)

person1 = {
  "name": "John",
  "age": 36,
  "country": "Norway"
}
```

```
import math

x = math.ceil(1.4)
y = math.floor(1.4)

print(x) # returns 2
print(y) # returns 1
```

```
import mymodule

mymodule.greeting("Jonathan")
```

```
from mymodule import person1

print (person1["age"])
```

```
import mymodule as mx

a = mx.person1["age"]
print(a)
```

```
import platform

x = dir(platform)
print(x)
```

1. Current directory
2. The list of directories in PYTHONPATH environment variable
3. Installation-dependent default path

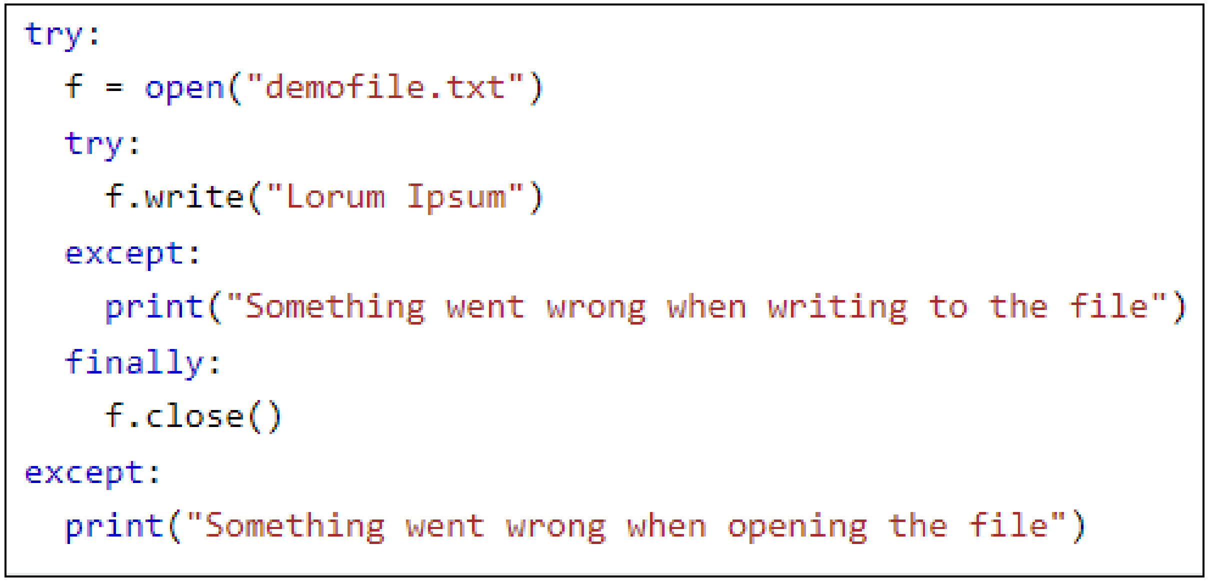
Python Packages

Directory structure of a Python package

```
vision/                                 Top-level package
       __init__.py                      Initialize vision package
       learning/                        Subpackage for learning
                 __init__.py
                 adaboost.py
                 svm.py
       tracking/                        Subpackage for tracking
                 __init__.py
                 kalman.py
       features/                        Subpackage for features
                 __init__.py
                 sift.py
                 harris.py
                 canny.py
```

```python
import vision.tracking.kalman as kf
kf.predict()
```

Python Try Except



```python
try:
  f = open("demofile.txt")
  try:
    f.write("Lorum Ipsum")
  except:
    print("Something went wrong when writing to the file")
  finally:
    f.close()
except:
  print("Something went wrong when opening the file")
```

File handling

demofile.txt

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

```python
f = open("demofile.txt", "r")
print(f.read(5))
```

```python
f = open("demofile.txt", "r")
print(f.readline())
```

```python
f = open("demofile.txt", "r")
for x in f:
  print(x)
```

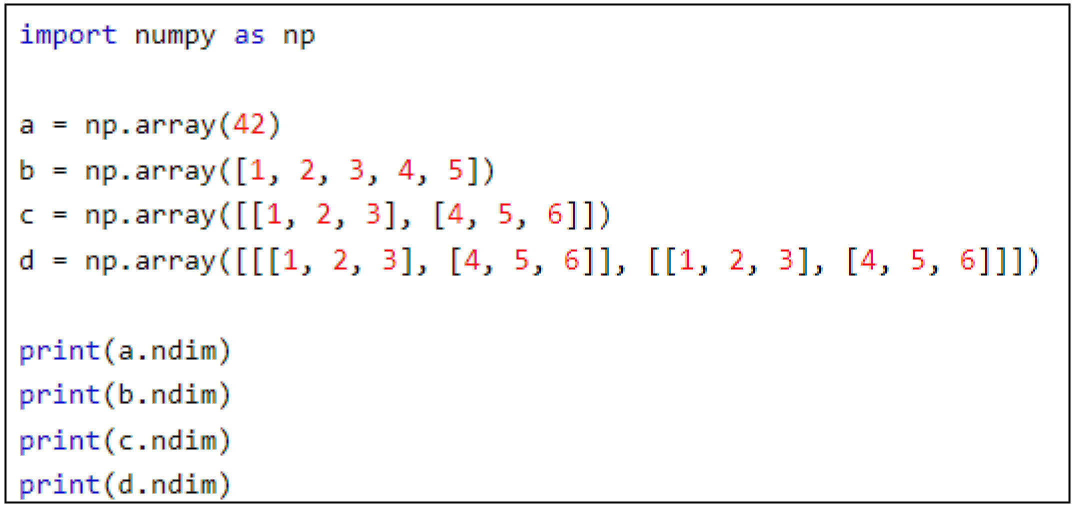
```python
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()
```

```python
import pickle

# take user input to take the amount of data
number_of_data = int(input('Enter the number of data : '))
data = []

# take input of the data
for i in range(number_of_data):
    raw = input('Enter data '+str(i)+' : ')
    data.append(raw)

# open a file, where you ant to store the data
file = open('important', 'wb')

# dump information to that file
pickle.dump(data, file)

# close the file
file.close()
```

```python
import pickle

# open a file, where you stored the pickled data
file = open('important', 'rb')

# dump information to that file
data = pickle.load(file)

# close the file
file.close()
```

Numpy

- 50x faster than traditional Python lists.
- ndarray object
- NumPy arrays are stored at one continuous place in memory



```python
import numpy as np

a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

Numpy

```
import numpy as np

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(arr[0, 1, 2])
```

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5:2])
```

```
import numpy as np

arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[0:2, 2])
```

```
newarr = arr.reshape(-1)
```

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

newarr = arr.reshape(4, 3)
```

```
import numpy as np

arr = np.array([1.1, 2.1, 3.1])

newarr = arr.astype('i')

print(newarr)
print(newarr.dtype)
```

```
arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
arr[0] = 42
```

```
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])

print(arr.shape)
```

```
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])

for idx, x in np.ndenumerate(arr):
  print(idx, x)
```

```
x = np.where(arr == 4)

arr = np.concatenate((arr1, arr2))
```

Matplotlib

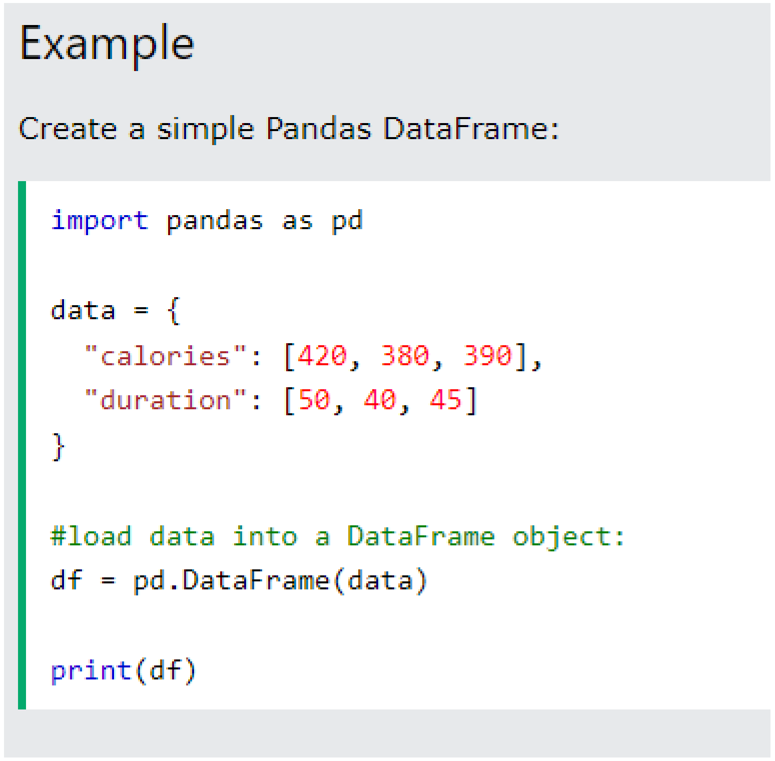
```
import matplotlib.pyplot as plt
import numpy as np

#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)

plt.show()
```

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])

plt.pie(y)
plt.show()
```

```
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.plot(x, y)

plt.grid()
```

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```

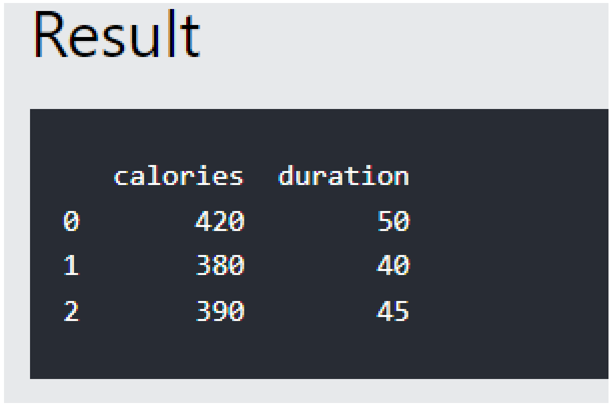
Pandas



Example

Create a simple Pandas DataFrame:

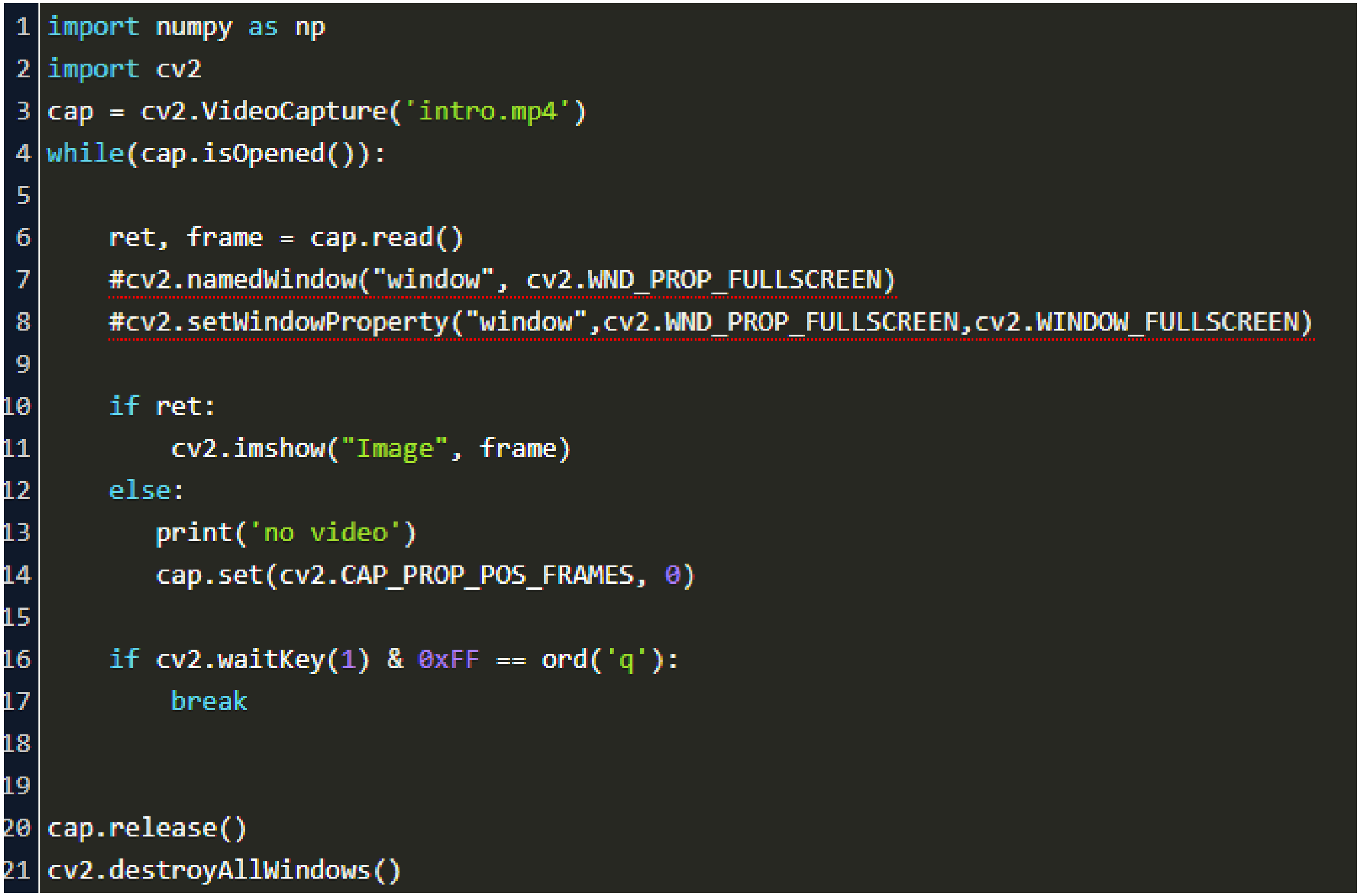
```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)

print(df)
```



Result

```
     calories  duration
0         420        50
1         380        40
2         390        45
```

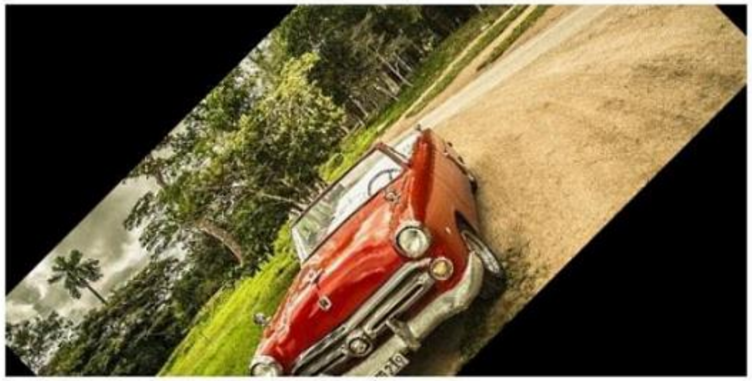
OpenCV



```python
import numpy as np
import cv2
cap = cv2.VideoCapture('intro.mp4')
while(cap.isOpened()):

    ret, frame = cap.read()
    #cv2.namedWindow("window", cv2.WND_PROP_FULLSCREEN)
    #cv2.setWindowProperty("window",cv2.WND_PROP_FULLSCREEN,cv2.WINDOW_FULLSCREEN)

    if ret:
        cv2.imshow("Image", frame)
    else:
       print('no video')
       cap.set(cv2.CAP_PROP_POS_FRAMES, 0)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


cap.release()
cv2.destroyAllWindows()
```

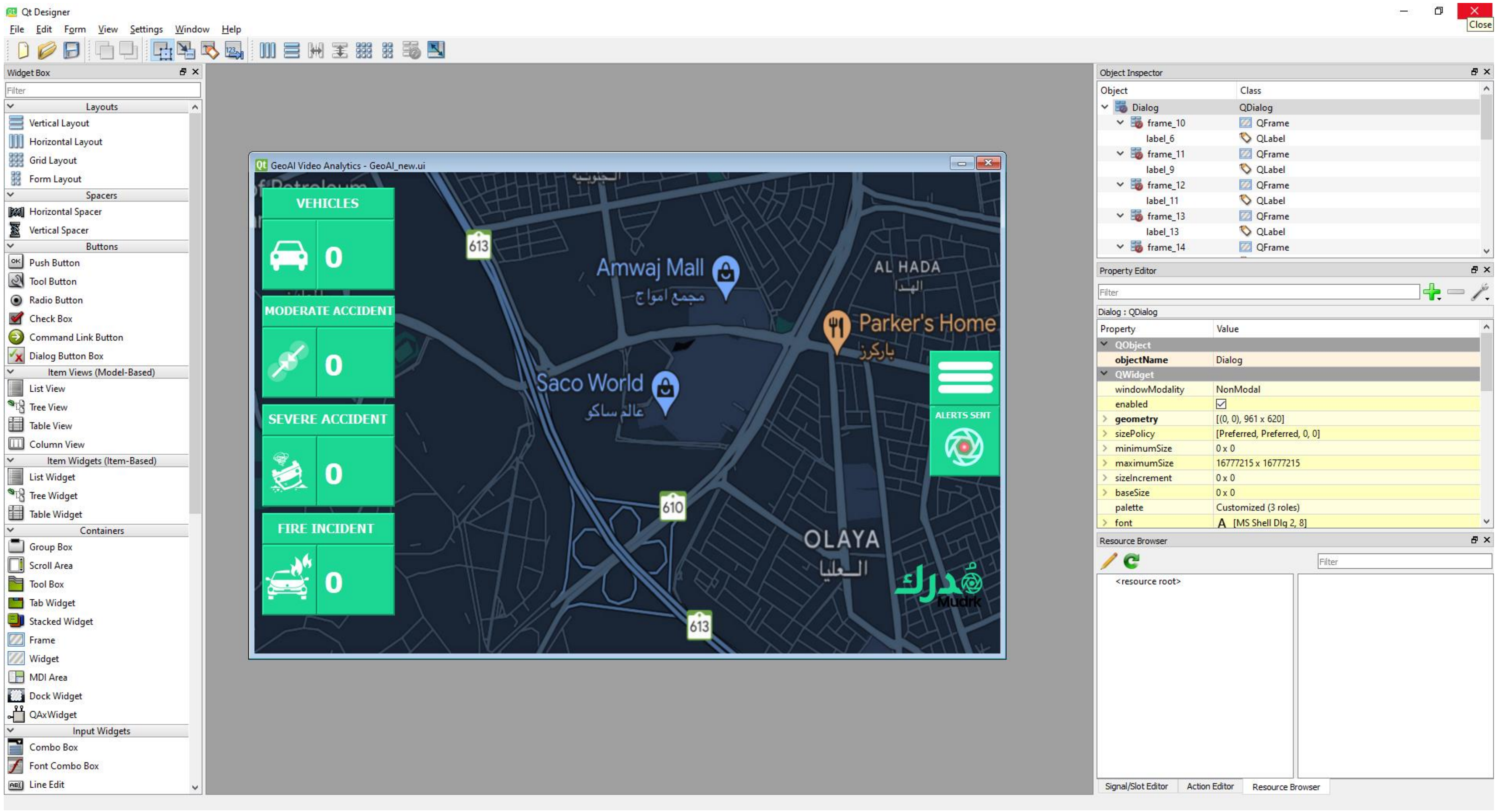
Pillow

```python
from PIL import Image
#Open image using Image module
im = Image.open("images/cuba.jpg")
#Show actual Image
im.show()
#Show rotated Image
im = im.rotate(45)
im.show()
```



Tkinter

GUI PyQT (Designer)



Object Inspector

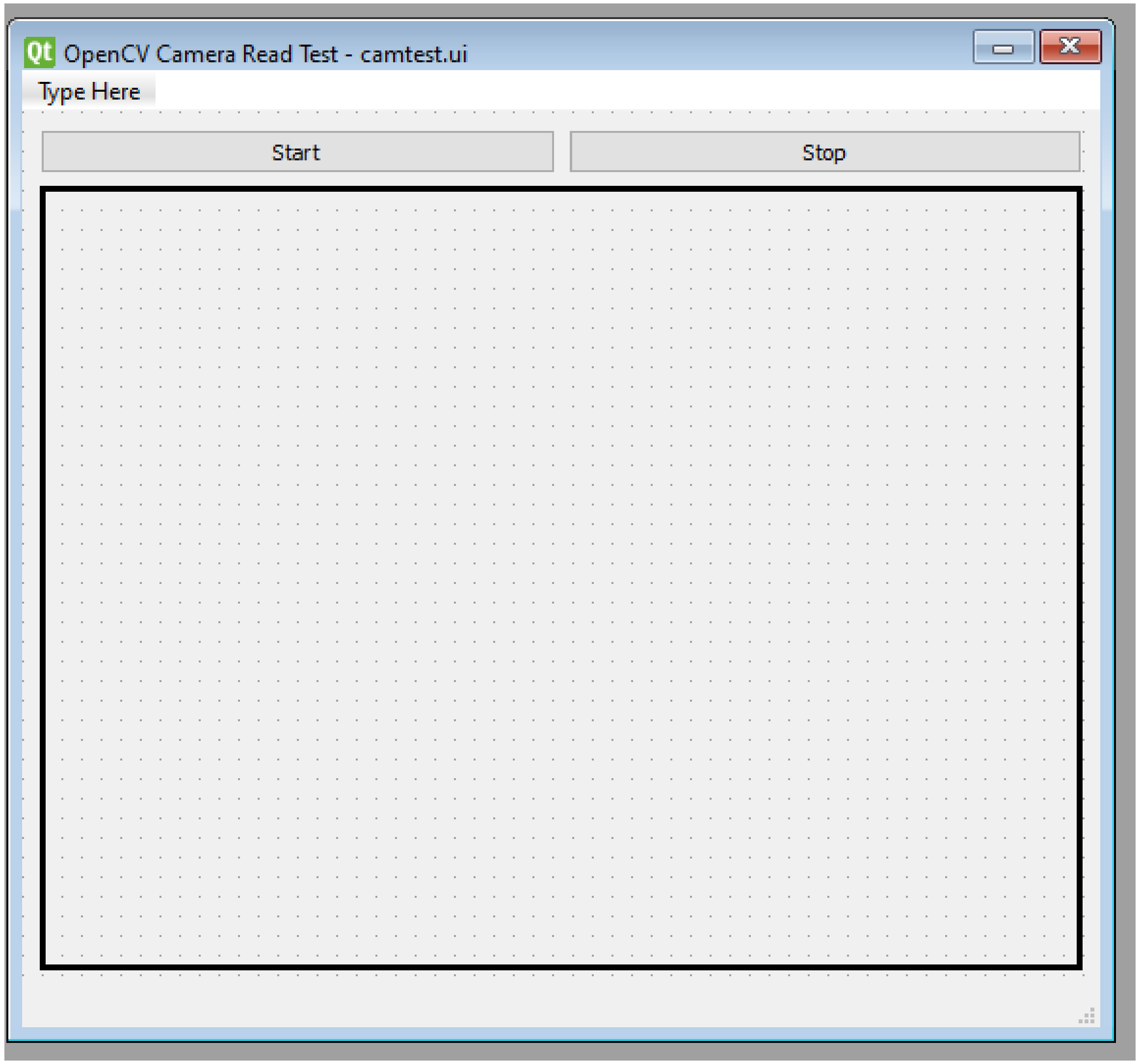
Object	Class
Dialog	QDialog
frame_10	QFrame
label_6	QLabel
frame_11	QFrame
label_9	QLabel
frame_12	QFrame
label_11	QLabel
frame_13	QFrame
label_13	QLabel
frame_14	QFrame

Property Editor

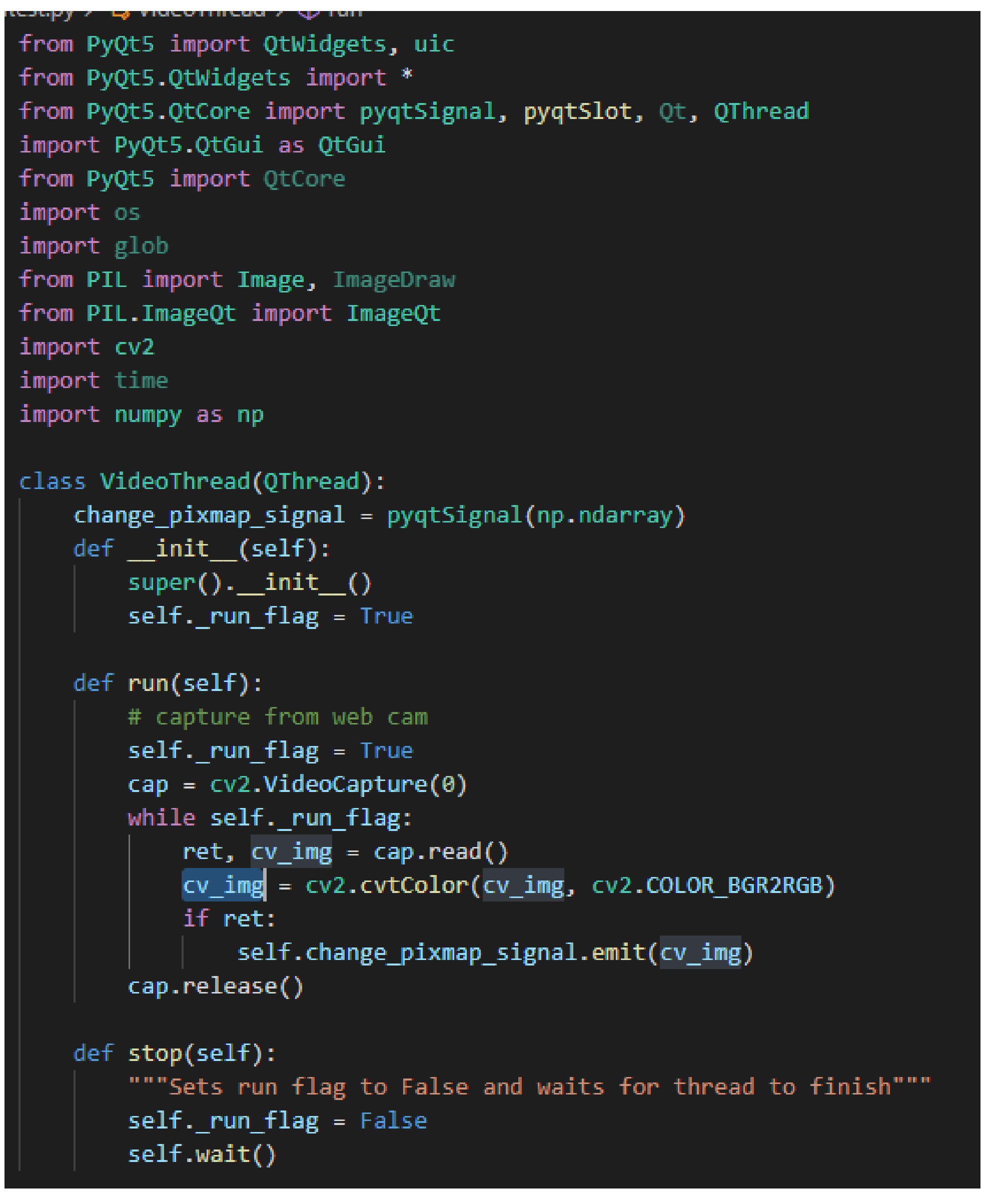
Dialog : QDialog

Property	Value
QObject	
objectName	Dialog
QWidget	
windowModality	NonModal
enabled	☑
geometry	[(0, 0), 961 x 620]
sizePolicy	[Preferred, Preferred, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Customized (3 roles)
font	A [MS Shell Dlg 2, 8]

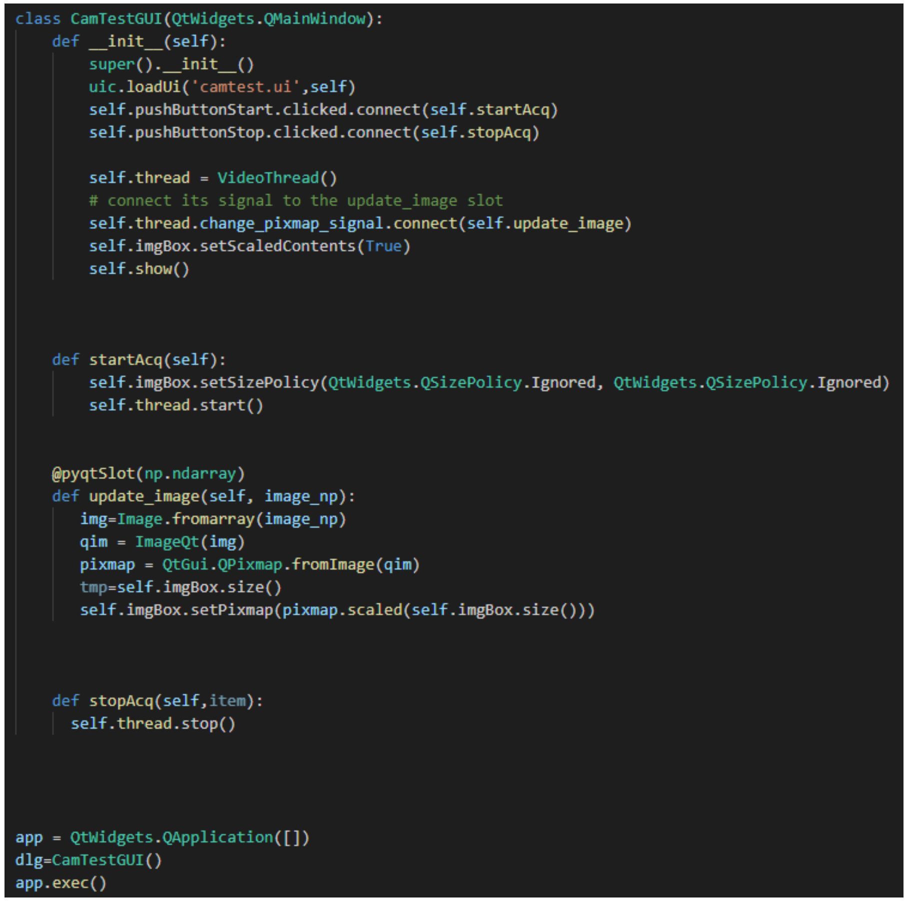
GUI PyQT (Code)



GUI PyQT, Threads (Code)

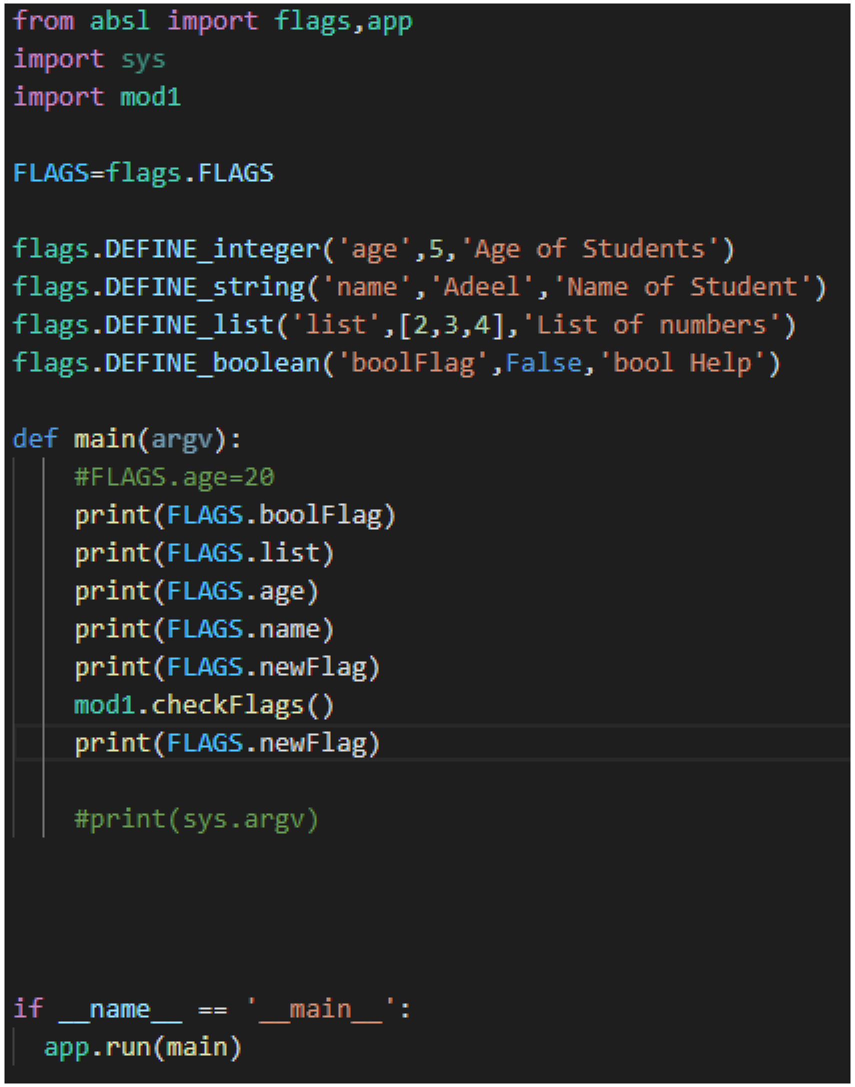


```python
from PyQt5 import QtWidgets, uic
from PyQt5.QtWidgets import *
from PyQt5.QtCore import pyqtSignal, pyqtSlot, Qt, QThread
import PyQt5.QtGui as QtGui
from PyQt5 import QtCore
import os
import glob
from PIL import Image, ImageDraw
from PIL.ImageQt import ImageQt
import cv2
import time
import numpy as np

class VideoThread(QThread):
    change_pixmap_signal = pyqtSignal(np.ndarray)
    def __init__(self):
        super().__init__()
        self._run_flag = True

    def run(self):
        # capture from web cam
        self._run_flag = True
        cap = cv2.VideoCapture(0)
        while self._run_flag:
            ret, cv_img = cap.read()
            cv_img = cv2.cvtColor(cv_img, cv2.COLOR_BGR2RGB)
            if ret:
                self.change_pixmap_signal.emit(cv_img)
        cap.release()

    def stop(self):
        """Sets run flag to False and waits for thread to finish"""
        self._run_flag = False
        self.wait()
```

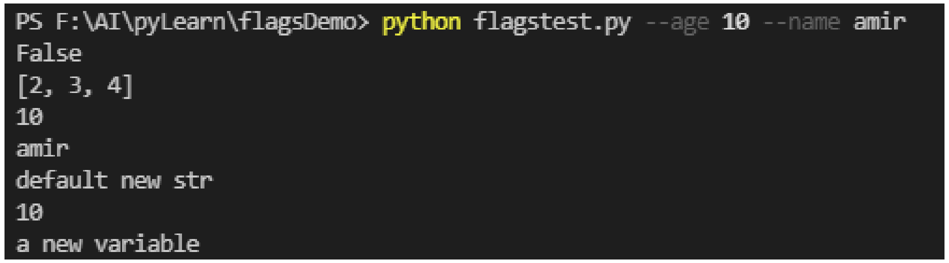


```python
class CamTestGUI(QtWidgets.QMainWindow):
    def __init__(self):
        super().__init__()
        uic.loadUi('camtest.ui',self)
        self.pushButtonStart.clicked.connect(self.startAcq)
        self.pushButtonStop.clicked.connect(self.stopAcq)

        self.thread = VideoThread()
        # connect its signal to the update_image slot
        self.thread.change_pixmap_signal.connect(self.update_image)
        self.imgBox.setScaledContents(True)
        self.show()



    def startAcq(self):
        self.imgBox.setSizePolicy(QtWidgets.QSizePolicy.Ignored, QtWidgets.QSizePolicy.Ignored)
        self.thread.start()


    @pyqtSlot(np.ndarray)
    def update_image(self, image_np):
       img=Image.fromarray(image_np)
       qim = ImageQt(img)
       pixmap = QtGui.QPixmap.fromImage(qim)
       tmp=self.imgBox.size()
       self.imgBox.setPixmap(pixmap.scaled(self.imgBox.size()))



    def stopAcq(self,item):
      self.thread.stop()




app = QtWidgets.QApplication([])
dlg=CamTestGUI()
app.exec()
```

Command Line



```python
from absl import flags,app
import sys
import mod1

FLAGS=flags.FLAGS

flags.DEFINE_integer('age',5,'Age of Students')
flags.DEFINE_string('name','Adeel','Name of Student')
flags.DEFINE_list('list',[2,3,4],'List of numbers')
flags.DEFINE_boolean('boolFlag',False,'bool Help')

def main(argv):
    #FLAGS.age=20
    print(FLAGS.boolFlag)
    print(FLAGS.list)
    print(FLAGS.age)
    print(FLAGS.name)
    print(FLAGS.newFlag)
    mod1.checkFlags()
    print(FLAGS.newFlag)

    #print(sys.argv)




if __name__ == '__main__':
  app.run(main)
```



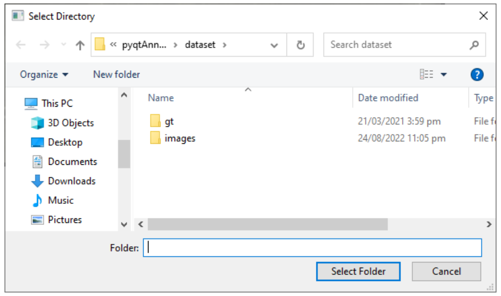
```
PS F:\AI\pyLearn\flagsDemo> python flagstest.py --age 10 --name amir
False
[2, 3, 4]
10
amir
default new str
10
a new variable
```

Programming Assignment 1

Annotation Visualization Utility

- Design an application in python where user can load a dataset directory having two sub-folders
  - Gt: The folder "ground truth" contains 650 separate text files and each one corresponds to an image in “images" folder. Each line of those text files defines a ground truth bounding box in the following format: (x1,y1),(x2,y2),a
    - where (x1,y1) denotes the top-left coordinate of the bounding box, (x2,y2) denotes the right-bottom coordinate of the bounding box, and a is the object class (1-airplane, 2-ship, 3-storage tank, 4-baseball diamond, 5-tennis court, 6-basketball court, 7-ground track field, 8-harbor, 9-bridge, 10-vehicle).
    - e.g: (563,478),(630,573),1
  - Images: 650 image files
- After selecting the dataset folder, all file names must be loaded in a list
- When user clicks a file name from the list, image must be loaded and displayed in a image box along with the bounding boxes of objects with name of object written on top.

Programming Assignment 1



Select Directory

« pyqtAnn... › dataset ›

Search dataset

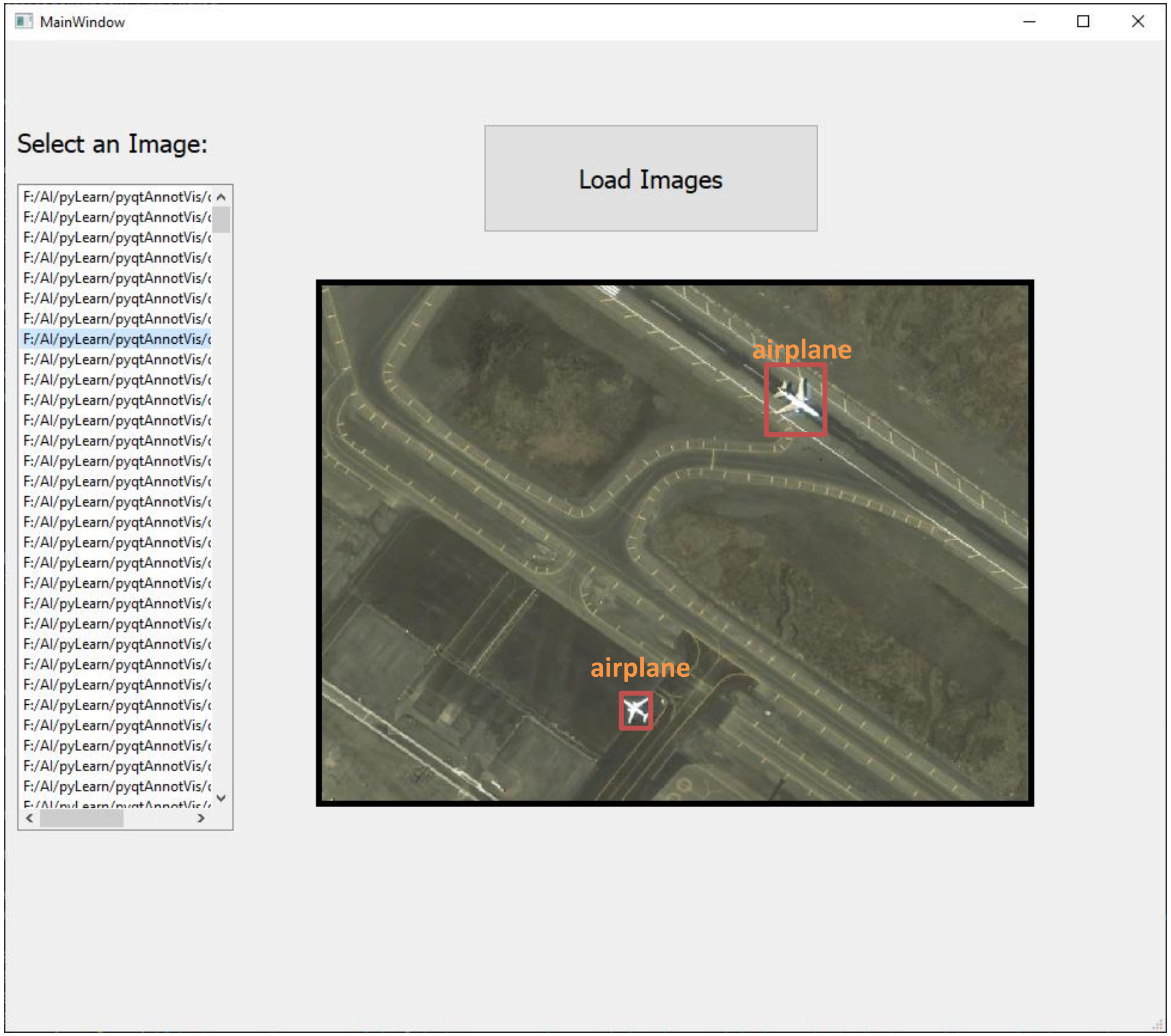
Organize ▾ New folder

Name	Date modified	Type
gt	21/03/2021 3:59 pm	File f
images	24/08/2022 11:05 pm	File f

This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures

Folder:

Select Folder Cancel



MainWindow

Select an Image:

F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/
F:/AI/pyLearn/pyqtAnnotVis/

Load Images