ML Python Tutorial

Python from 0 to 1

Outline

- Installing and Environment
- Basic
- Numpy
- Matplotlib

Installing and Environment

Install Python 3.5+

Linux:

- apt-get (default python 3.x)

Mac:

- Homebrew

Windows:

- Official Website

pip / pip3

For Python 2.7.9+ or Python 3.4+

- you should already have pip or pip3 installed

Manual install

- Installing with get-pip.py from here

Difference between pip and pip3

- Stack Overflow
- Quora

Virtual Environment

“*virtualenv* is a tool to create isolated Python environments.”

- $ [sudo] pip install virtualenv

“pyenv” v.s. *“pyvenv”* v.s. *“virtualenv”*: check it out from here

Virtual Environment Wrapper

“virtualenvwrapper is a set of extensions to Ian Bicking’s virtualenv tool”

- $ [sudo] pip install virtualenvwrapper

Commands:

- $mkvirtualenv mynewenv
- (mynewenv) $ deactivate
- $

Other commands: rmvirtualenv, workon, etc.

-

Other Useful Learning Tools

IDLE

- IDLE is Python's Integrated Development and Learning Environment

IPython

- IPython is a command shell for interactive computing in multiple programming languages, originally developed for Python

Basic

Before Start...

- Interpreted language
- Identation matters (whitespace recommended)
- Dynamic type
- NOT backward compatible

Hello ML World

- $ vim test1.py
- print(‘Hello ML World!’)
- $ python3 test1.py
- Hello ML World!

Numeric Unary Operators

- x = 2
- print(x) # 2
- x += 3
- print(x) # 5
- x -= 0.5
- print(x) # 4.5

Numeric Binary Operators

- x = 5
- y = 2
- print(x - y) # 3
- print(x / y) # 2.5
- print(x // y) # 2
- print(x % y) # 1
- print(x ** y) # 25

String

- s1 = 'Hello World'
- s2 = 'ML'
- print(s1) # Hello World
- print(s1[0:4]) # Hell
- print(s1[:6] + s2) # Hello ML
- print(s1.replace('World' , 'Kitty')) # Hello Kitty

String (cont.)

- s1 = 'Hello World'
- print(s1.split()) # ['Hello', 'World']
- print(s1.upper()) # HELLO WORLD
- print(len(s1)) # 11
- print("Hello %s World %d" % ('ML' , 101)) # Hello ML World 101
- other methods: strip(), splitlines(), etc.
- coding style: single quotes v.s. double quotes in Python check out here

Python List

- myList = ['Hello', 'ML'] # ['Hello', 'ML']
- myList.append('World') # ['Hello', 'ML', 'World']
- myList.remove('ML') # ['Hello', 'World']
- myList.insert(1, 'Kitty') # ['Hello', 'Kitty', 'World']
- myList.pop(2) # ['Hello', 'Kitty']
- newList = [1,2,3]
- myList.extend(newList) # ['Hello', 'Kitty', 1, 2, 3]

Tuple

“A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Tuple (cont.)

```
- tup1 = ( 1,2,3 )
- tup2 = ( 4,5,6 )
- print( tup1 )                  # (1, 2, 3)
- print( len( tup1 ))            # 3
- print( tup1*2 )                # (1, 2, 3, 1, 2, 3)
- print( tup1 + tup2 )           # (1, 2, 3, 4, 5, 6)
- print( 3 in tup2 )             # False
- for x in tup2:
-     print( x )                  # 4  5  6
```

Set

“A set is an unordered collection with no duplicate elements”

- basket = {'apple', 'banana', 'guava'}
- print(basket) # {'banana', 'guava', 'apple'}
- print('apple' in basket) # True
- print('orange' in basket) # False

“note: to create an empty set you have to use set( ), NOT { }; the latter creates an empty dictionary”

Set (cont.)

- s1 = {'a', 'b', 'c', 'c', 'c', 'd', 'd'}
- s2 = {'a', 'e'}
- print(s1 - s2) # {'c', 'b', 'd'}
- print(s1 | s2) # {'e', 'c', 'b', 'a', 'd'}
- print(s1 & s2) # {'a'}
- print(s1 ^ s2) # {'b', 'c', 'e', 'd'}

Dictionary

“Dictionary is an unordered set of key : value pairs, with the requirement that the keys are unique within one dictionary.”

```
- d = {'GAI': 95, 'DMOB': 92, 'PG ONE': 95, 'Jony J': 90}
- print(d['GAI'])            # 95
- d['VAVA'] = 93             # {'GAI': 95, 'DMOB': 92, 'PG ONE': 95, 'Jony J':
- print(d)                       90, 'VAVA': 93}
- print(list(d.keys()))      # ['GAI', 'DMOB', 'PG ONE', 'Jony J', 'VAVA']
- print('PG ONE' in d)       # True
- print('2Real' in d)        # False
```

if ... elif ... else

```
d = {'GAI': 95, 'DMOB': 92, 'PG ONE': 95, 'Jony J': 90}
if d['GAI'] == d['PG ONE']:
    print('@@')
elif d['Jony J'] > d['GAI'] and d['Jony J'] > d['PG ONE']:
    print('WTF')
else:
    print('MC Jin')
```

for loop

- for i in range(5):
- print(i) # 0 1 2 3 4
- for i in reversed(range(5)):
- print(i) # 4 3 2 1 0
- for i, v in enumerate(['a' , 'b' , 'c']):
- print(i, v) # 0 a , 1 b , 2 c

for loop (cont.)

```
- number = { 'Kobe' : 24 , 'Lebron' : 23 }
- for k , n in number.items():                    # Kobe 24
-     print( k , n )                               # Lebron 23
- player = [ 'Duncan' , 'Dirk' ]
- team = [ 'Spurs' , 'Mavs' ]
- for p, t in zip( player , team ):               # Duncan from Spurs
-     print( '{0} from {1}'.format( p , t ) )     # Dirk from Mavs
```

Function

- def my_function(x , y):
- return x*2 , y**2 # identation
- p , q = my_function(10 , 20)
- print(p , q) # 20 400

Class

```
- class Teacher:
-     def __init__(self, name):
-         self.name = name
-     def update_name(self, name):
-         self.name = name
- teacher_list = []
- teacher_list.append( Teacher('MC HotDog') )
- teacher_list.append( Teacher('Kris Wu') )
- teacher_list.append( Teacher('Will Pan') )
- teacher_list[0].update_name('MC HotDog and Ayal Komod')
- for i in teacher_list:
-     print( i.name )
```

Output:

MC HotDog and Ayal Komod
Kris Wu
Will Pan

Import

- import sys
- sys.stdout.write('What\'s\n') # What's
- # stdout.write('Up!\n') NameError: name 'stdout' is not defined
- from sys import stdout
- stdout.write('Up!\n') # Up!

I/O

- print() # automatically change line
- sys.stdout.write() # do NOT change line without ‘\n’

I/O

- import sys
- if len(sys.argv) < 2:
- sys.stderr.write('Error: missing input argument\n')
- exit()
- with open(sys.argv[1], 'w') as of:
- of.write('check check 1 2\n')

```
$ python3 filename.py test.txt
$ cat test.txt
>>> check check 1 2
```

Relative Path v.s. Absolute Path

Absolute Path

- $ pwd
- /Users/Jason/Desktop/Tutorial

Relative Path

- Relative to directory Tutorial, the path of Desktop is “ ../ ”

Exceptions Handling

```
- def my_divide(x, y):
-     try:
-         x / y
-     except ZeroDivisionError:
-         print('Error: zero division')
-     else:
-         print('Result: ' + str(x/y) + ' (without error)')
-     finally:
-         print('Done')
- my_divide(10,5)          # Result: 2.0 (without error)      Done
- my_divide(10,0)          # Error: zero division             Done
```

NumPy

NumPy

“NumPy is a library for scientific computing in Python, providing high-performance multidimensional array objects and tools for working with these arrays.”

Install NumPy: check out here

NumPy - Array Creation

- import numpy as np
- a = np.array([2,3,4])
- print(a) # [2 3 4]
- print(a.dtype) # int64
- print(a.shape) # (3,)
- print(a.ndim) # 1
- print(a.size) # 3

- b = np.array([[1.5, 2.3] , [1.9, 3.7]])
- print(b) # [[1.5 2.3]
  [1.9 3.7]]
- print(b.dtype) # float64
- print(b.shape) # (2, 2)
- print(b.ndim) # 2
- print(b.size) # 4

NumPy - Array Creation (cont.)

- import numpy as np
- print(np.zeros((2,1))) # [[0.]
  [0.]]
- print(np.ones(3)) # [1. 1. 1.]
- print(np.arange(0,1,0.3)) # [0. 0.3 0.6 0.9]

NumPy - Reshape

- import numpy as np
- print(np.arange(24).reshape((2,3,4)))

```
# [ [ [ 0  1  2  3 ]
    [ 4  5  6  7 ]
    [ 8  9 10 11 ] ]
  [ [12 13 14 15 ]
    [16 17 18 19 ]
    [20 21 22 23 ] ] ]
```

NumPy - Basic Operations - 1

- import numpy as np
- a = np.array([5,10,15])
- b = np.arange(3)
- print(b) # [0 1 2]
- print(a-b) # [5 9 13]
- print(b**2) # [0 1 4]
- print(a>12) # [False False True]

NumPy - Basic Operations - 2

- import numpy as np
- A = np.array([[1,0] , [0,1]])
- B = np.array([[1,2] , [3,4]])
- print(A*B)

```
# [ [ 1 0 ]
#   [ 0 4 ] ]
```

- print(A.dot(B))

```
# [ [ 1 2 ]
#   [ 3 4 ] ]
```

NumPy - Basic Operations - 3

- import numpy as np
- a = np.array([[1,2] , [3,4]])
- print(a.sum()) # 10
- print(a.min()) # 1
- print(a.max(axis=0)) # [3 4]
- print(a.max(axis=1)) # [2 4]
- print(np.sqrt(a)) # [[1. 1.41421356]
  [1.73205081 2.]]

NumPy - Basic Operations - 4

```
- import numpy as np
- a = np.arange(12)
- print(a)                                  # [ 0  1  2  3  4  5  6  7  8  9 10 11]
- print(a[3])                               # 3
- print(a[1:5])                             # [1 2 3 4]
- c = a.reshape(2,-1)
- print(c)                                  # [[ 0  1  2  3  4  5  ]
                                            #  [ 6  7  8  9 10 11 ]]
- b = np.array([[100,100]])
- print(np.concatenate((c,b.T), axis=1))    # [[  0   1   2   3   4   5   100 ]
                                            #  [  6   7   8   9  10  11 100 ]]
```

Matplotlib

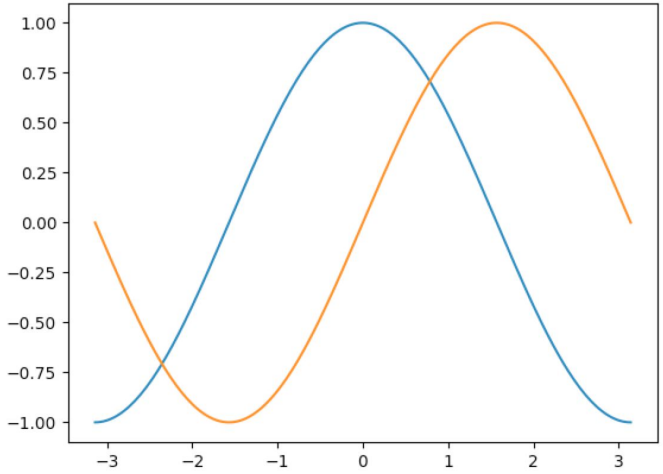
Matplotlib

“Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.”

- Install Matplotlib: check out here
- matplotlib.pyplot: provides a MATLAB-like plotting framework

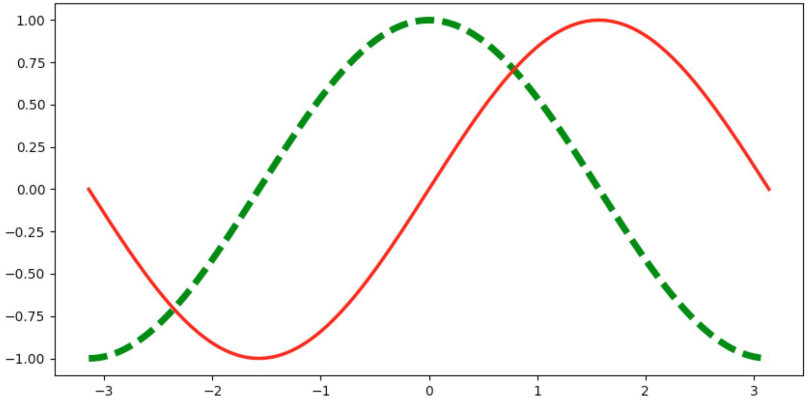
Matplotlib - First Glance

- import numpy as np
- import matplotlib.pyplot as plt
-
- X = np.linspace(-np.pi, np.pi, 200)
- C, S = np.cos(X), np.sin(X)
-
- plt.plot(X,C)
- plt.plot(X,S)
-
- plt.show()



Matplotlib - Style

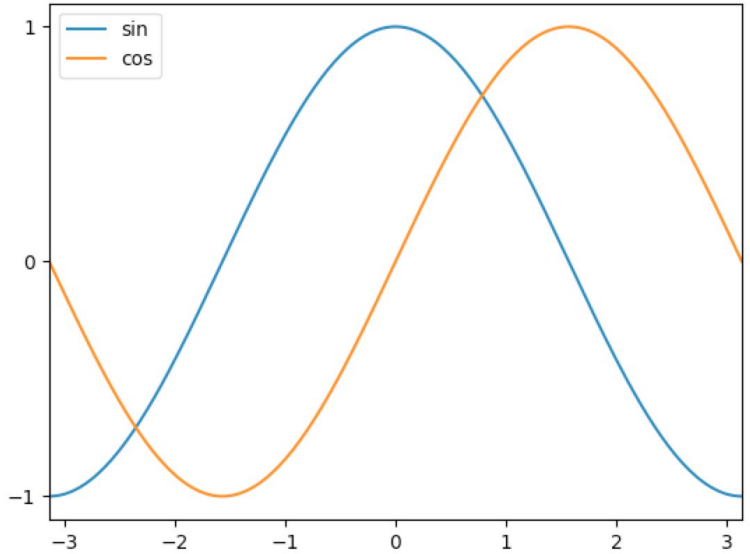
- import numpy as np
- import matplotlib.pyplot as plt
-
- X = np.linspace(-np.pi, np.pi, 200)
- C, S = np.cos(X), np.sin(X)
-
- plt.figure(figsize=(10,5))
- plt.plot(X,C, color="green", linewidth=5.0, linestyle="--")
- plt.plot(X,S, color="red", linewidth=2.5, linestyle="-")
-
- plt.show()
-



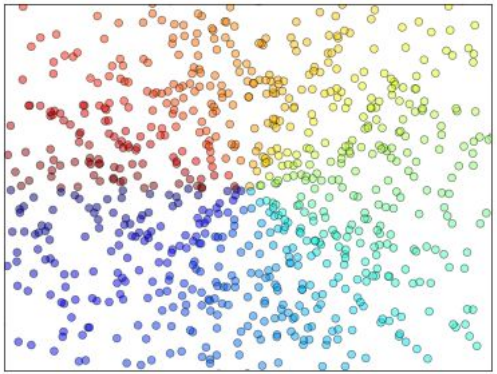
Matplotlib - Style (cont.)

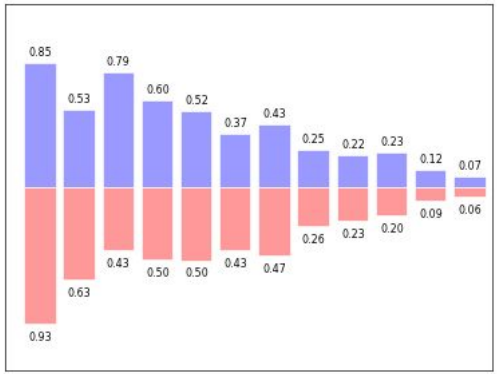
- import numpy as np
- import matplotlib.pyplot as plt

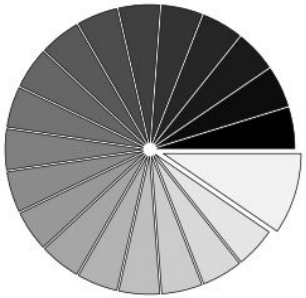
- X = np.linspace(-np.pi, np.pi, 200)
- C, S = np.cos(X), np.sin(X)
- plt.plot(X,C, label='sin')
- plt.plot(X,S, label='cos')
- plt.xlim(X.min(), X.max())
- plt.yticks([-1, 0, +1])
- plt.legend(loc='upper left')
- plt.show()

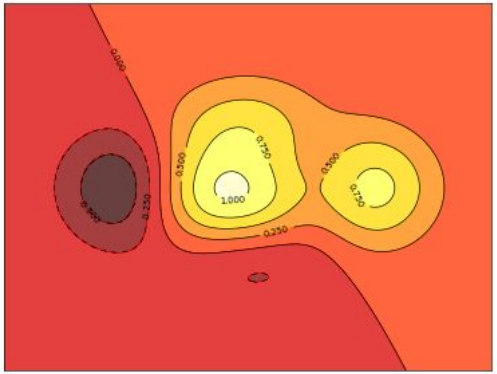


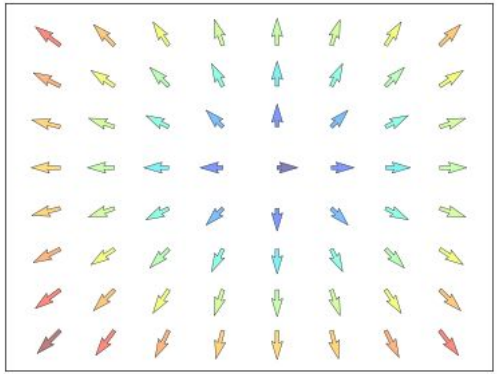
Matplotlib - Showcases

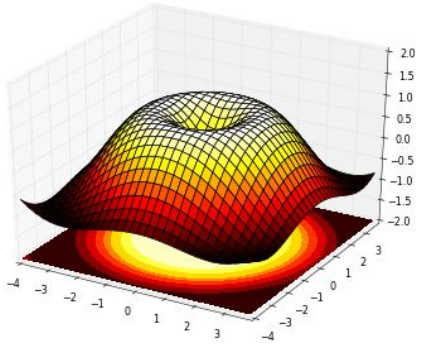












Thank You