nb

July 31, 2019

1 (1) Python Basics

1.1 Types 1

[1]:
```
a = True
type(a)
```

[1]:
```
bool
```

[2]:
```
a = 1
type(a)
```

[2]:
```
int
```

[3]:
```
a = 1.0
type(a)
```

[3]:
```
float
```

1.2 Operators 1

[4]:
```
1 + 3 - 2 * 3 / 4.0
```

[4]:
```
2.5
```

[5]:
```
1 == 1.0
```

[5]:
```
True
```

[6]:
```
a = True
not a
```

[6]:
```
False
```

[7]:
```
a = True
a and a or not a
```

[7]:
```
True
```

1.3 Types 2

```
[8]: a = "hello"
     type(a)
```

```
[8]: str
```

```
[9]: a = [1, 2, 3]
     type(a)
```

```
[9]: list
```

```
[10]: a = {1, 2, 3}
      type(a) # unique elements
```

```
[10]: set
```

```
[11]: a = (1, 2, 3)
      type(a) # immutable
```

```
[11]: tuple
```

```
[12]: a = {"a": 1, "b": 2}
      type(a)
```

```
[12]: dict
```

1.4 Operators 2

```
[13]: "Hel" + "lo"
```

```
[13]: 'Hello'
```

```
[14]: [1, 2, 3] + [5, 4]
```

```
[14]: [1, 2, 3, 5, 4]
```

```
[15]: [1, 2] * 2
```

```
[15]: [1, 2, 1, 2]
```

```
[16]: {1, 2, 3} | {2, 3, 4}
```

```
[16]: {1, 2, 3, 4}
```

```
[17]: {1, 2, 3} & {2, 3, 4}
```

```
[17]: {2, 3}
```

```
[18]: a = {"a": 1, "b": 2}
      a.update({"b": 3, "c": 4})
      a
```

```
[18]: {'a': 1, 'b': 3, 'c': 4}
```

```
[19]: a = [1, 2, 3]
      b = [1, 2, 3]
```

```python

a is b

a is a

a == b

b = a

a is b
```

[19]:
```
True
```

1.5 Indexing

[20]:
```python
[1, 2, 3, 4, 5][1]
```

[20]:
```
2
```

[21]:
```python
[1, 2, 3, 4, 5][-2]
```

[21]:
```
4
```

[22]:
```python
[1, 2, 3, 4, 5][:2] # slicing
```

[22]:
```
[1, 2]
```

[23]:
```python
[1, 2, 3, 4, 5][:-1]
```

[23]:
```
[1, 2, 3, 4]
```

[24]:
```python
[1, 2, 3, 4, 5][-1:]
```

[24]:
```
[5]
```

[25]:
```python
[1, 2, 3, 4, 5][::-1] # reverting
```

[25]:
```
[5, 4, 3, 2, 1]
```

[26]:
```python
a = {"a": 1, "b": 2}
a["b"]
```

[26]:
```
2
```

[27]:
```python
a = {"a": 1, "b": 2}
a["b"] = 3
a
```

[27]:
```
{'a': 1, 'b': 3}
```

1.6 Flow control and iteration

[28]:
```python
a = 2

if a == 2:
    print("a")
elif a > 2:
    print("b")
else:
    print("c")
```

```
a
```

[29]:
```python
for list_el in [3, 1, 2]:
    print(list_el)
```

```
3
1
2
```

[30]:
```python
for list_el in [3, 1, 2]:
    if list_el == 2:
        break
    print(list_el)
```

```
3
1
```

[31]:
```python
for list_el in range(2, 10):
    print(list_el)
```

```
2
3
4
5
6
7
8
9
```

1.7 list iteration/comprehenshion

[32]:
```python
[i for i in range(10)]
```

[32]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[33]:
```python
[i for i in range(10) if i % 2 == 0]
```

[33]:
```
[0, 2, 4, 6, 8]
```

[34]:
```
[i if i % 2 == 0 else -i for i in range(10)]
```

[34]:
```
[0, -1, 2, -3, 4, -5, 6, -7, 8, -9]
```

1.8 Functions

[35]:
```
def fun():
    pass
fun()
```

[36]:
```
def fun():
    return 1
fun()
```

[36]:
```
1
```

[37]:
```
def fun(a):
    return a
fun(2)
```

[37]:
```
2
```

[38]:
```
# optional key word arguments
def fun(a, b=1):
    return a + b
print(fun(2))
print(fun(2, 3))
```

```
3
5
```

[39]:
```
# optional number of arguments (args) and keyword arguments (kwargs)
def fun(*args, **kwargs):
    return args, kwargs

print(fun(1))
print(fun(1, 2))
print(fun(a=1))
print(fun(1, 2, a=2, b=3))
```

```
((1,), {})
((1, 2), {})
((), {'a': 1})
((1, 2), {'a': 2, 'b': 3})
```

[40]:
```
def fun(list_):
    list_[0] = 4
```

```
a = [1, 2, 3]
fun(a)
a
```

[40]:

```
[4, 2, 3]
```

1.9 Module imports

[41]:

```
import numpy
numpy
```

[41]:

```
<module 'numpy' from '/Users/davidlassner/Envs/wh2/lib/python3.7/site-
packages/numpy/__init__.py'>
```

[42]:

```
import numpy as np
np
```

[42]:

```
<module 'numpy' from '/Users/davidlassner/Envs/wh2/lib/python3.7/site-
packages/numpy/__init__.py'>
```

[43]:

```
import numpy.random
numpy.random
```

[43]:

```
<module 'numpy.random' from '/Users/davidlassner/Envs/wh2/lib/python3.7/site-
packages/numpy/random/__init__.py'>
```

[44]:

```
from numpy import random
random
```

[44]:

```
<module 'numpy.random' from '/Users/davidlassner/Envs/wh2/lib/python3.7/site-
packages/numpy/random/__init__.py'>
```

[45]:

```
from numpy import random as rng
rng
```

[45]:

```
<module 'numpy.random' from '/Users/davidlassner/Envs/wh2/lib/python3.7/site-
packages/numpy/random/__init__.py'>
```

2 (2) Numerical Python

[46]:

```
import numpy as np
```

[47]:

```
print(np.__doc__[:186])
```

```


NumPy
=====

Provides
  1. An array object of arbitrary homogeneous items
```

2. Fast mathematical operations over arrays
3. Linear Algebra, Fourier Transforms, Random Number Generation

2.1 Basic Operations

[48]:

```
a = np.array([1, 2, 3])
print(a)
print(a[:2])
print(a.shape)
```

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

[49]:

```
a = np.array(range(9)).reshape(3,3)
print(a)
print(a.sum()) # summation
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
36
```

[50]:

```
a = np.array(range(9)).reshape(3,3)
# summation over specific axis
print(a.sum(axis=0))
print(a.sum(axis=1))
```

```
[ 9 12 15]
[ 3 12 21]
```

[51]:

```
a = np.array(range(9)).reshape(3,3)
print(a)
# transpose
print(a.T)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
[[0 3 6]
 [1 4 7]
 [2 5 8]]
```

[52]:

```
a = np.array(range(9)).reshape(3,3)
np.multiply(a,a) # element wise multiplication
```

```
[52]: array([[ 0,  1,  4],
              [ 9, 16, 25],
              [36, 49, 64]])
```

```
[53]: a = np.array(range(9)).reshape(3,3)
      a.dot(a) # dot product
```

```
[53]: array([[ 15,  18,  21],
              [ 42,  54,  66],
              [ 69,  90, 111]])
```

```
[54]: a = rng.randint(1,10,(3,3))
      np.linalg.inv(a) # inverse
```

```
[54]: array([[-0.7826087 ,  0.26086957,  1.        ],
              [-1.39130435,  0.13043478,  2.        ],
              [ 1.52173913, -0.17391304, -2.        ]])
```

```
[55]: a = np.array(range(27)).reshape(3,3,3) # array not limited to 2 dimensions
      a.shape
```

```
[55]: (3, 3, 3)
```

2.2 Various indexing methods

```
[56]: np.random.rand?
```

```
[57]: mask = np.random.rand(9) > .5
      mask
```

```
[57]: array([False, False,  True, False,  True,  True, False, False,  True])
```

```
[58]: np.arange(9)[mask]
```

```
[58]: array([2, 4, 5, 8])
```

```
[59]: np.arange(81).reshape(9,9)
```

```
[59]: array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8],
              [ 9, 10, 11, 12, 13, 14, 15, 16, 17],
              [18, 19, 20, 21, 22, 23, 24, 25, 26],
              [27, 28, 29, 30, 31, 32, 33, 34, 35],
              [36, 37, 38, 39, 40, 41, 42, 43, 44],
              [45, 46, 47, 48, 49, 50, 51, 52, 53],
              [54, 55, 56, 57, 58, 59, 60, 61, 62],
              [63, 64, 65, 66, 67, 68, 69, 70, 71],
              [72, 73, 74, 75, 76, 77, 78, 79, 80]])
```

```
[60]: np.arange(81).reshape(9,9)[:,mask]
```

```
[60]: array([[ 2,  4,  5,  8],
              [11, 13, 14, 17],
              [20, 22, 23, 26],
              [29, 31, 32, 35],
              [38, 40, 41, 44],
              [47, 49, 50, 53],
              [56, 58, 59, 62],
              [65, 67, 68, 71],
              [74, 76, 77, 80]])
```

[61]:
```
np.arange(81).reshape(9,9)[mask,mask]
```

[61]:
```
array([20, 40, 50, 80])
```

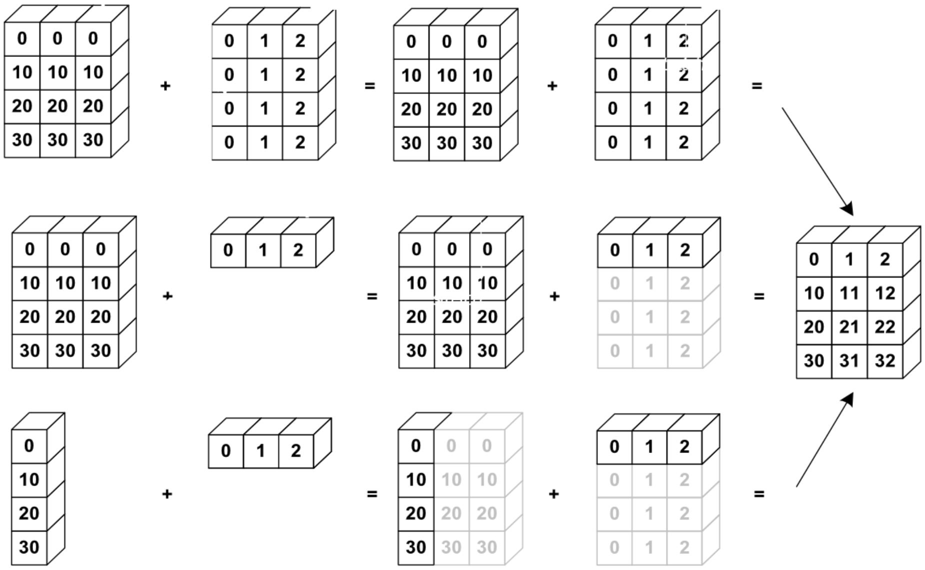
[62]:
```
np.arange(81).reshape(9,9)[[0,1,2]]
```

[62]:
```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8],
       [ 9, 10, 11, 12, 13, 14, 15, 16, 17],
       [18, 19, 20, 21, 22, 23, 24, 25, 26]])
```

[ ]:
```

```

2.3 Broadcasting



Numpy broadcasting

[63]:
```
a = np.ones((3, 1))
print(a)
print(a.shape)
print ("--")
b = np.ones((1,2))
```

```
print(b)
print(b.shape)
print ("--")

print(a + b)
print((a + b).shape)
```

```
[[1.]
 [1.]
 [1.]]
(3, 1)
--
[[1. 1.]]
(1, 2)
--
[[2. 2.]
 [2. 2.]
 [2. 2.]]
(3, 2)
```

3 (3) IO

3.1 writing and reading files

[64]:

```
with open("dat.tsv", "w") as fout:
    fout.write("col1\tcol2\tcol3\n")
    for row in np.arange(9).reshape(3,3):
        fout.write("\t".join([str(el) for el in row]) + "\n")
```

[65]:

```
with open("dat.tsv", "r") as fin:
    header = fin.readline()
    cols = header.strip().split("\t")
    data = []
    for line in fin:
        data.append(line.strip().split("\t"))

data = np.array(data, dtype="float32")

print(header)
print(data)
```

```
col1    col2    col3

[[0. 1. 2.]
 [3. 4. 5.]
 [6. 7. 8.]]
```

3.2 Plotting

[66]:
```
import matplotlib.pyplot as plt
plt
```

[66]:
```
<module 'matplotlib.pyplot' from
'/Users/davidlassner/Envs/wh2/lib/python3.7/site-packages/matplotlib/pyplot.py'>
```

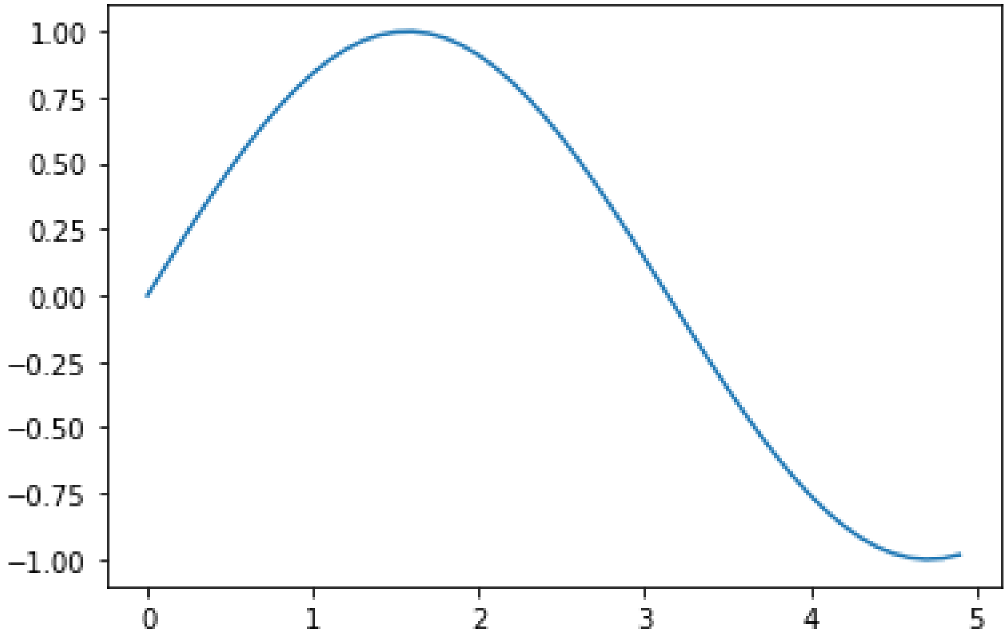
[ ]:
```
```

[67]:
```
# to make plots visible in ipython notebook
%matplotlib inline
```

3.2.1 Line plot

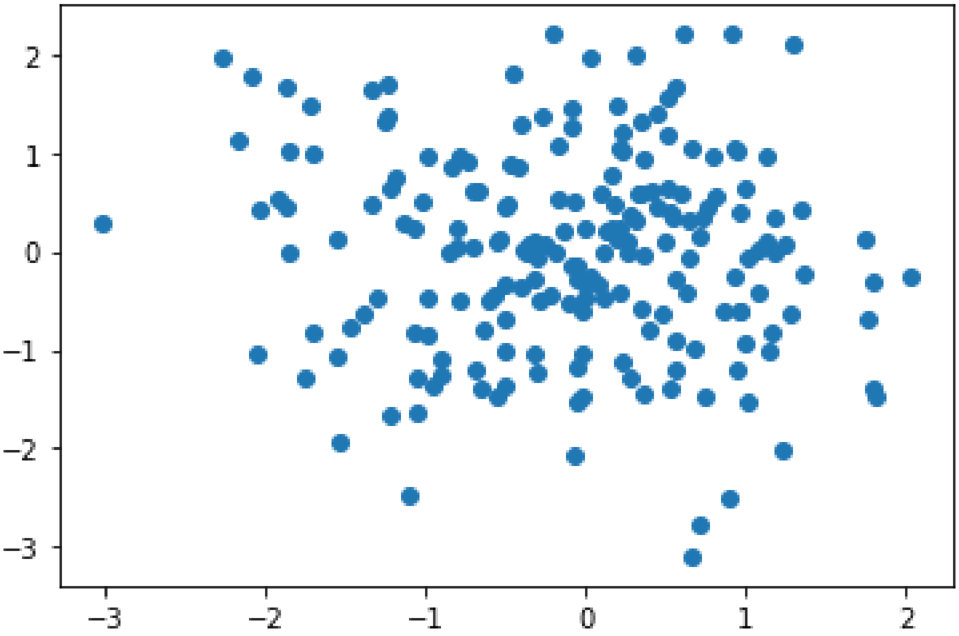
[68]:
```
x = np.arange(0, 5, 0.1);
y = np.sin(x)
plt.plot(x, y)
```

[68]:
```
[<matplotlib.lines.Line2D at 0x107922850>]
```



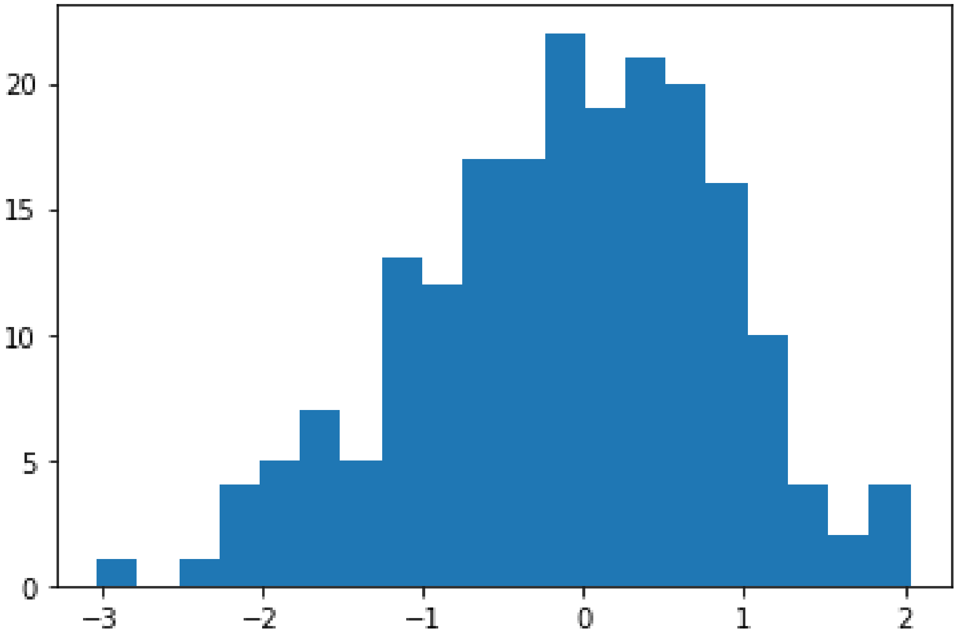
3.2.2 scatter plot

[69]:
```
a = np.random.randn(200)
b = np.random.randn(200)
_ = plt.scatter(a, b)
```



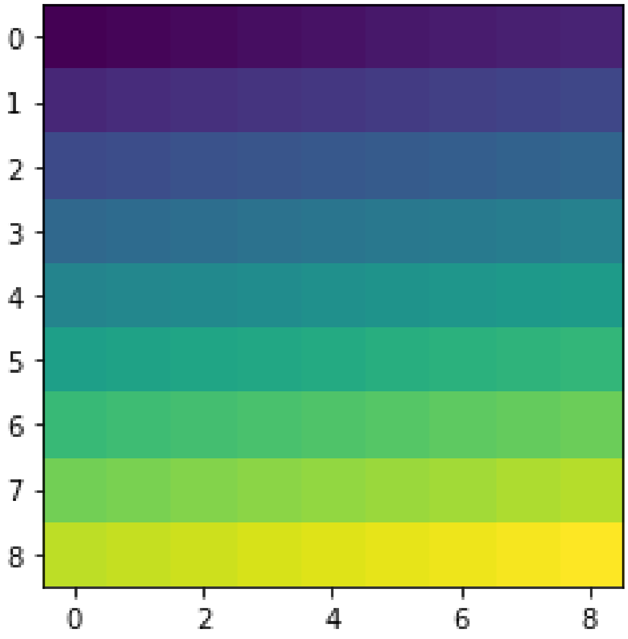
3.2.3 histogram

[70]:
```
_ = plt.hist(a, bins=20)
```



3.2.4 plot an image

[71]:
```
_ = plt.imshow(np.arange(81).reshape(9,9))
```



4 (4) Object oriented

[72]:
```
class A():

    def __init__(self):
        pass
```

[73]:
```
a = A()
a
```

[73]:
```
<__main__.A at 0x11524c850>
```

[74]:
```
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html
class NDArray(list):

    def __init__(self, data):
        super(NDArray, self).__init__(data)

    def summation(self):
        sum_ = 0
        for n in self:
            for d in n:
                sum_ += d
        return sum_

    def __getitem__(self,i):
        return super(NDArray, self).__getitem__(i[0])[i[1]]
```

[75]:

```
X = NDArray([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
])

X.summation()

for row in X:
    print(row)

X[2, 0]
```

```
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

[75]: 7

[76]:

```
# http://scikit-learn.org/stable/developers/contributing.
 ↪html#rolling-your-own-estimator
from sklearn.base import BaseEstimator, ClassifierMixin

class RandomClassifier(BaseEstimator, ClassifierMixin):

    def __init__(self, distribution=None):
        if distribution is not None:
            self.distribution = distribution
        else:
            self.distribution = lambda x: rng.randint(0,10,len(x))


    def fit(self, X, y):
        return self

    def predict(self, X):
        return self.distribution(X)
```

[77]:

```
N, D = 100, 10

X_train = rng.randn(N,D)
```

```
X_test = rng.randn(N,D)
Y_train = rng.randint(0,5,N)


clf = RandomClassifier()

clf.fit(X_train, Y_train)

clf.predict(X_test)
```

[77]:
```
array([8, 2, 3, 0, 8, 4, 4, 6, 4, 8, 6, 6, 7, 1, 8, 4, 8, 4, 4, 5, 5, 3,
       3, 8, 0, 2, 3, 8, 1, 1, 9, 9, 7, 0, 6, 2, 8, 3, 7, 3, 4, 9, 4, 8,
       5, 1, 3, 7, 5, 6, 6, 4, 7, 3, 6, 7, 8, 0, 5, 4, 0, 7, 0, 7, 8, 8,
       9, 4, 4, 7, 1, 0, 7, 4, 7, 2, 4, 1, 3, 1, 9, 3, 9, 3, 6, 8, 7, 7,
       1, 4, 1, 5, 6, 9, 9, 4, 7, 3, 3, 0])
```

[78]:
```
clf.get_params()
```

[78]:
```
{'distribution': <function
__main__.RandomClassifier.__init__.<locals>.<lambda>(x)>}
```

5 (5) Debugging

[80]:
```
import pdb

for i in range(100):
    if i == 23:
        pdb.set_trace() # try "l", "bt", "s", "n", "c" and printing variables
```

6 Further reading

6.1 General style recommendations

- https://www.python.org/dev/peps/pep-0008/
- https://www.python.org/dev/peps/pep-0020/

6.2 Scientific python and ML

- http://www.scipy-lectures.org/
- http://scikit-learn.org/stable/
- https://pytorch.org/

6.3 Some important things that could not be covered

- (object oriented python) https://docs.python.org/3/tutorial/classes.html
- (functional python) https://docs.python.org/3/howto/functional.html
- (performance profiling) https://docs.python.org/2/library/profile.html

- (Cython) http://docs.cython.org/en/latest/src/tutorial/cython_tutorial.html
- (multiprocessing) https://docs.python.org/2/library/multiprocessing.html
- ...