python_basics

July 17, 2019

1 Python: basic features

https://www.python.org/

```
In [1]: print("Hello, World!")
```

```
Hello, World!
```

```
In [2]: a = 5
        b = 2

In [3]: a + b

Out[3]: 7

In [4]: 1 + a * b

Out[4]: 11

In [5]: a ** b

Out[5]: 25

In [6]: # different in python 3: a//b
        # for same behaviour run: from __future__ import division
        a / b

Out[6]: 2

In [7]: a / float(b)

Out[7]: 2.5

In [8]: a % b

Out[8]: 1

In [9]: min(a, b)
```

```
Out[9]: 2

In [10]: a == b

Out[10]: False

In [11]: a != b

Out[11]: True

In [12]: a += 3
         a

Out[12]: 8

In [13]: a = [1, "hello", 5.5]
         a

Out[13]: [1, 'hello', 5.5]

In [14]: len(a)

Out[14]: 3

In [15]: a[2]

Out[15]: 5.5

In [16]: a.append("how are you?")
         a

Out[16]: [1, 'hello', 5.5, 'how are you?']

In [17]: for x in a:
             print(x)

1
hello
5.5
how are you?


In [18]: for i, x in enumerate(a):
             print("element {}: {}".format(i, x))

element 0: 1
element 1: hello
element 2: 5.5
element 3: how are you?
```

```
In [19]: a[0] = 10
         a

Out[19]: [10, 'hello', 5.5, 'how are you?']

In [20]: b = (-1, "bye")
         b

Out[20]: (-1, 'bye')

In [21]: b[1]

Out[21]: 'bye'

In [22]: b[0] = 10
         b



        TypeErrorTraceback (most recent call last)

        <ipython-input-22-c58040f40f7e> in <module>()
    ----> 1 b[0] = 10
          2 b


        TypeError: 'tuple' object does not support item assignment


In [23]: x, y = b

In [24]: x

Out[24]: -1

In [25]: y

Out[25]: 'bye'

In [26]: a = {"name":"Mary", "age":23, "sign":"capricorn"}
         a

Out[26]: {'age': 23, 'name': 'Mary', 'sign': 'capricorn'}

In [27]: a["age"]

Out[27]: 23

In [28]: a["job"] = "student"
         a
```

```
Out[28]: {'age': 23, 'job': 'student', 'name': 'Mary', 'sign': 'capricorn'}
```

```
In [29]: def f(a, b=4, c=5):
             if a > 2 and b < 10:
                 return a
             elif c == 5:
                 return b
             else:
                 return a + b + c
```

```
In [30]: f(4)
```

```
Out[30]: 4
```

```
In [31]: f(4, 11)
```

```
Out[31]: 11
```

```
In [32]: f(4, c=6, b=11)
```

```
Out[32]: 21
```

2 NumPy: multi-dimensional arrays and scientific computing

https://www.numpy.org/

```
In [33]: import numpy as np
```

```
In [34]: a = np.array([0, 2, 4, 6, 8, 10, 12, 14, 16])
         a
```

```
Out[34]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16])
```

```
In [35]: a.ndim
```

```
Out[35]: 1
```

```
In [36]: a.shape
```

```
Out[36]: (9L,)
```

```
In [37]: a[2]
```

```
Out[37]: 4
```

```
In [38]: a[2:]
```

```
Out[38]: array([ 4,  6,  8, 10, 12, 14, 16])
```

```
In [39]: a[:4]
```

```
Out[39]: array([0, 2, 4, 6])

In [40]: a[2:7]

Out[40]: array([ 4,  6,  8, 10, 12])

In [41]: a[2:7:2]

Out[41]: array([ 4,  8, 12])

In [42]: a[-1]

Out[42]: 16

In [43]: a[::-1]

Out[43]: array([16, 14, 12, 10,  8,  6,  4,  2,  0])

In [44]: a[[0, 4, 5]]

Out[44]: array([ 0,  8, 10])

In [45]: b = a > 3
         b

Out[45]: array([False, False,  True,  True,  True,  True,  True,  True,  True], dtype=bool)

In [46]: a[b]

Out[46]: array([ 4,  6,  8, 10, 12, 14, 16])

In [47]: a = np.array([[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]])
         a

Out[47]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11]])

In [48]: a.ndim

Out[48]: 2

In [49]: a.shape

Out[49]: (3L, 4L)

In [50]: a[1, 2]

Out[50]: 6

In [51]: a[0]
```

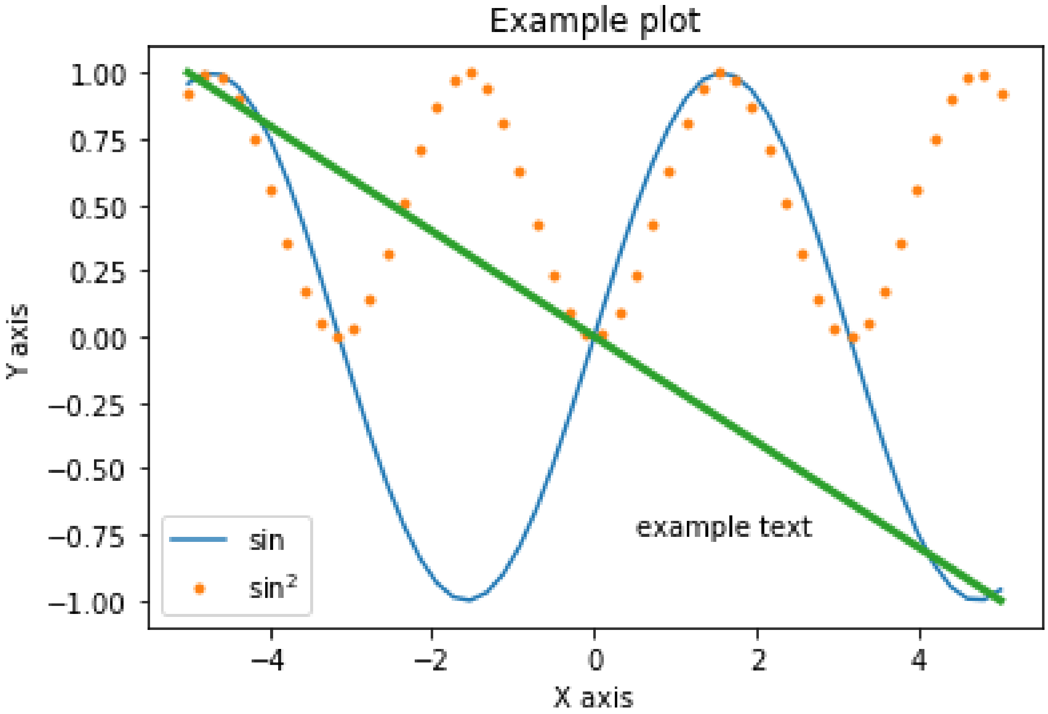
```
Out[51]: array([0, 1, 2, 3])

In [52]: a[:, 1:3]

Out[52]: array([[ 1,  2],
                [ 5,  6],
                [ 9, 10]])

In [53]: a.T

Out[53]: array([[ 0,  4,  8],
                [ 1,  5,  9],
                [ 2,  6, 10],
                [ 3,  7, 11]])

In [54]: a + 10

Out[54]: array([[10, 11, 12, 13],
                [14, 15, 16, 17],
                [18, 19, 20, 21]])

In [55]: a ** 2

Out[55]: array([[  0,   1,   4,   9],
                [ 16,  25,  36,  49],
                [ 64,  81, 100, 121]])

In [56]: a * [10, 20, 30, 40]

Out[56]: array([[  0,  20,  60, 120],
                [ 40, 100, 180, 280],
                [ 80, 180, 300, 440]])

In [57]: np.sin(a)

Out[57]: array([[ 0.        ,  0.84147098,  0.90929743,  0.14112001],
                [-0.7568025 , -0.95892427, -0.2794155 ,  0.6569866 ],
                [ 0.98935825,  0.41211849, -0.54402111, -0.99999021]])

In [58]: np.mean(a)

Out[58]: 5.5

In [59]: a.mean(axis=1)

Out[59]: array([ 1.5,  5.5,  9.5])

In [60]: np.max(a)

Out[60]: 11
```

```
In [61]: np.max(a, axis=1)

Out[61]: array([ 3,  7, 11])

In [62]: np.arange(10)

Out[62]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [63]: np.linspace(2, 4, 5)

Out[63]: array([ 2. ,  2.5,  3. ,  3.5,  4. ])

In [64]: np.zeros((2, 3))

Out[64]: array([[ 0.,  0.,  0.],
                [ 0.,  0.,  0.]])

In [65]: np.full((2, 3), 2.5)

Out[65]: array([[ 2.5,  2.5,  2.5],
                [ 2.5,  2.5,  2.5]])
```

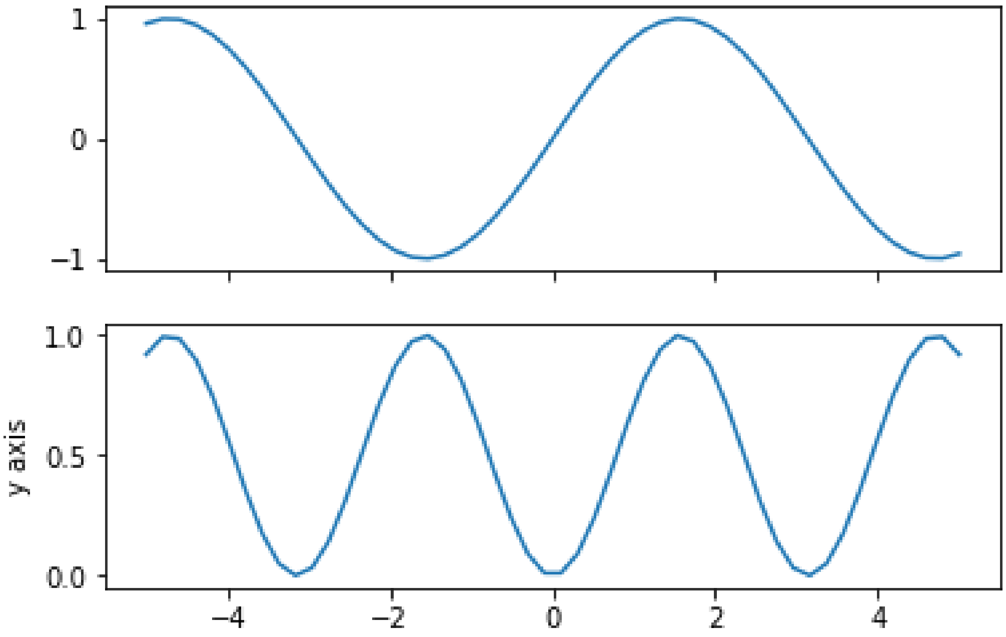
3 matplotlib: plotting

https://matplotlib.org/

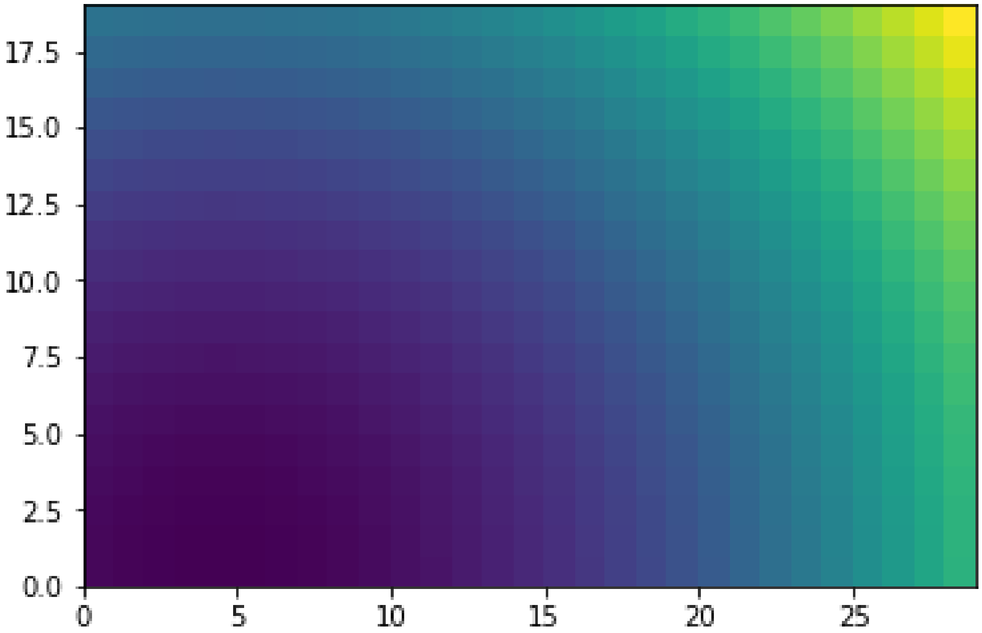
```
In [66]: import matplotlib.pyplot as plt

In [67]: #%matplotlib notebook
         #%matplotlib inline

In [68]: x = np.linspace(-5, 5, 50)
         y = np.sin(x)
         y2 = y ** 2
         y3 = -x / 5

In [69]: plt.figure()
         plt.plot(x, y, label='sin')
         plt.plot(x, y2, '.', label='$\sin^{2}$')
         plt.plot(x, y3, linewidth=3)
         plt.annotate('example text', xy=(0.5, -0.75))
         plt.xlabel("X axis")
         plt.ylabel("Y axis")
         plt.title("Example plot")
         plt.legend()
         plt.show()
```



```
In [70]: fig, ax = plt.subplots(2, sharex=True)
         ax[0].plot(x, y)
         ax[1].plot(x, y2)
         ax[1].set_ylabel('y axis')
         plt.show()
```



```
In [71]: y, x = np.mgrid[0:20, 0:30]
         z = (x - 4)**2+ y**2
         plt.figure()
         plt.pcolormesh(x, y, z)
         plt.show()
```



4 SciPy: extra modules for scientific computation

https://www.scipy.org/

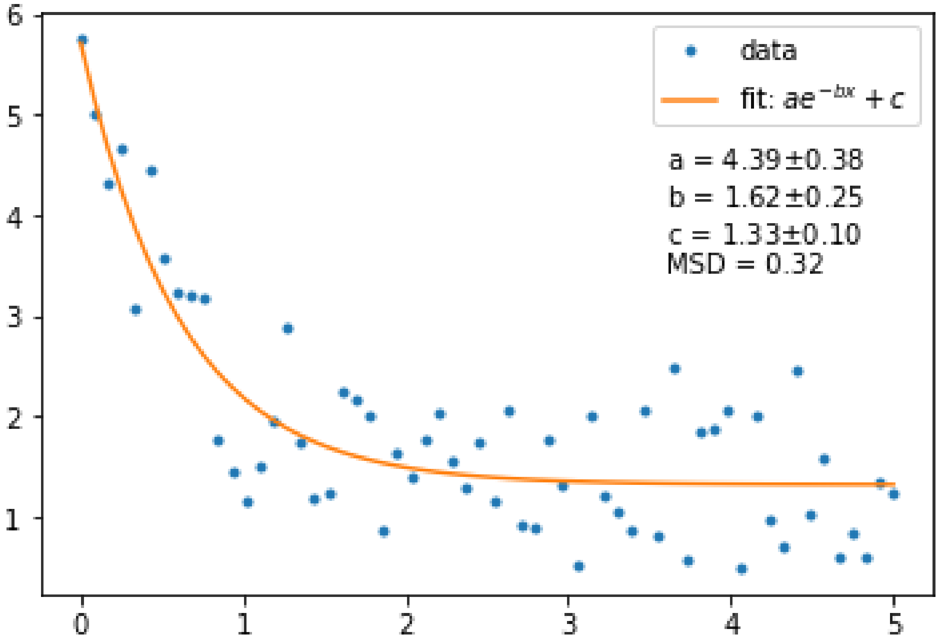
```
In [72]: from scipy.optimize import curve_fit
```

```
In [73]: def f(x, a, b, c):
             return a * np.exp(-b * x) + c
```

```
In [74]: n = 60
         x = np.linspace(0, 5, n)
         y = f(x, 5, 2, 0.5) + 2 * np.random.rand(n)
```

```
In [75]: popt, pcov = curve_fit(f, x, y)
         perr = np.sqrt(np.diag(pcov))
         y_fit = f(x, *popt)
         msd = np.sum((y - y_fit) ** 2) / n
```

```
In [76]: pnames = ['a', 'b', 'c']
         results = ''
         for name, value, error in zip(pnames, popt, perr):
             results += '{} = {:.2f}$\pm${:.2f}\n'.format(name, value, error)
         results += 'MSD = {:.2f}'.format(msd)

         plt.plot(x, y, '.', label='data')
         plt.plot(x, y_fit, label='fit: $ae^{-bx} + c$')
         plt.annotate(results, xy=(0.7, 0.55), xycoords='axes fraction')
         plt.legend()
         plt.show()
```



```
In [ ]:
```