Python Basics

長庚大學電機系

詹曉龍

Control flow

For loop in python

```python
sum = 0

for i in range(5):
    print('i=', i)
    sum = i + sum

print('sum=', sum)
```

```python
sum = 0

for i in range(0, 5, 1):
    print('i=', i)
    sum = i + sum

print('sum=', sum)
```

Encapsulation

Define a **function** then reuse it

```python
def summation(start, end):
    sum = 0
    for i in range(start, end+1, 1):
        sum = i + sum
    return sum

sum_1 = summation(1, 4)
print('sum_1=', sum_1)
sum_2 = summation(2, 7)
print('sum_2=', sum_2)
```

Conditional statement

Check **condition** and change behavior

```python
num_1 = 1
num_2 = 3
if num_1 > num_2:
    print('num_1 is greater than num_2')
else:
    print('num_1 is not greater than num_2')
```

Python modules and packages

Numpy (matrix computing)

SciPy (scientific computing)

Matplotlib (picture plotting)

Pandas (data structures)

general purposes

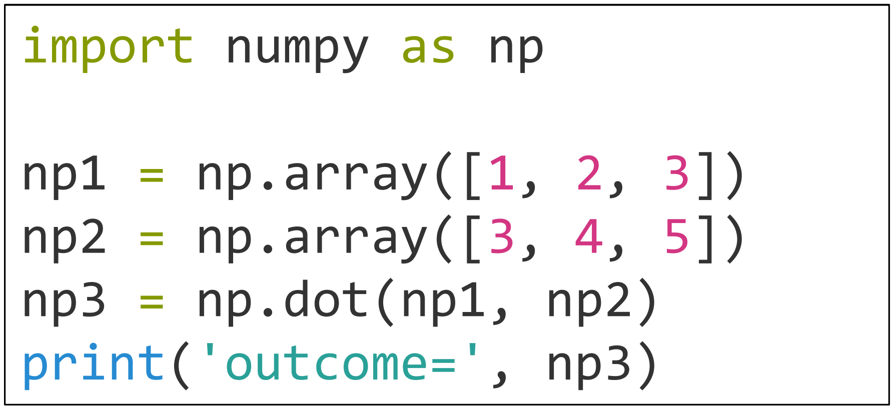
Scikit-learn (machine learning)

TensorFlow (deep learning)

PyTorch (deep learning)

specific purpose

Import module

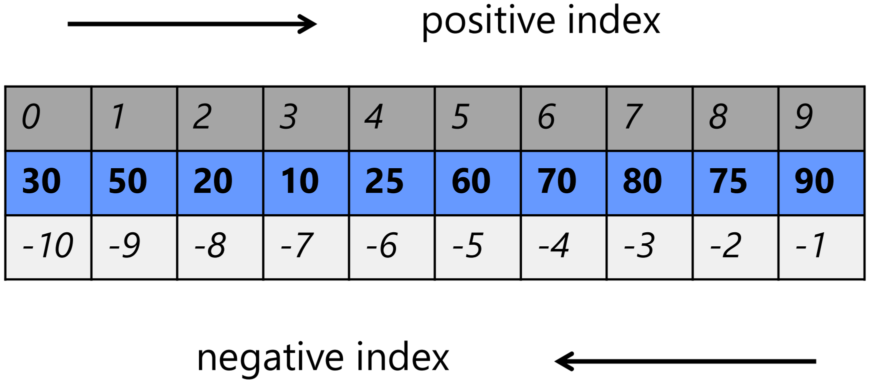


```
import numpy as np

np1 = np.array([1, 2, 3])
np2 = np.array([3, 4, 5])
np3 = np.dot(np1, np2)
print('outcome=', np3)
```

Indexing and slice

```
vector = np.arange(10)
print(vector)
#indexing
print(vector[0])
print(vector[2])
print(vector[-3])
print(vector[:])
#indexing with stride
print(vector[::2])
#slice
print(vector[3:6])
print(vector[:6])
print(vector[6:])
#slice with stride
print(vector[:6:2])
print(vector[6::2])
```



positive index

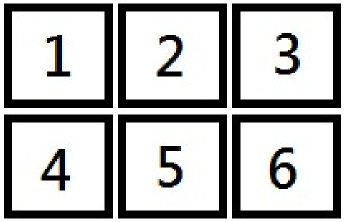
0	1	2	3	4	5	6	7	8	9
30	**50**	**20**	**10**	**25**	**60**	**70**	**80**	**75**	**90**
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

negative index

Create matrix and tensor

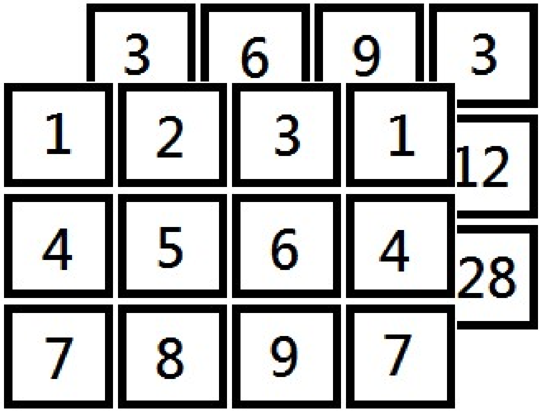
```
matrix_1 = np.array([[1, 2, 3], [4, 5, 6]])
print(matrix_1)
tensor_1 = np.array([[[1, 2, 3, 1], [4, 5, 6, 4], [7, 8, 9,
7]],[[3, 6, 9, 3], [12, 15, 18, 12],[28, 32, 36, 28]]])
print(tensor_1)
```

```
print(matrix_1.shape)
print(tensor_1.shape)
```



1 2 3
4 5 6

matrix



3 6 9 3
12
28

1 2 3 1
4 5 6 4
7 8 9 7

tensor

Reshape and resize

Change the vector to a matrix (tensor) by dimension transformation and vice versa

```
vector = np.arange(10)
matrix_2 = vector.reshape(2, 5)
print(vector)
print(matrix_2)
vector.resize(2, 5)
print(vector)
```

```
vector_2 = matrix_2.reshape(matrix_2.shape[0]*matrix_2.shape[1])
print(vector_2)
vector_3 = matrix_2.flatten()
print(vector_3)
```

Reshape and resize

Change the vector to a matrix (tensor) by dimension transformation and vice versa

```
vector = np.arange(10)
matrix_2 = vector.reshape(2, 5)
print(vector)
print(matrix_2)
vector.resize(2, 5)
print(vector)
```

```
vector_2 = matrix_2.reshape(matrix_2.shape[0]*matrix_2.shape[1])
print(vector_2)
vector_3 = matrix_2.flatten()
print(vector_3)
```

Simulation of bungee jumper model

velocity *v* (m/s)
time *t* (s)
mass *m* (kg)
drag coefficient c_d, (kg/m)
gravitational acceleration g = 9.81 m/s^2

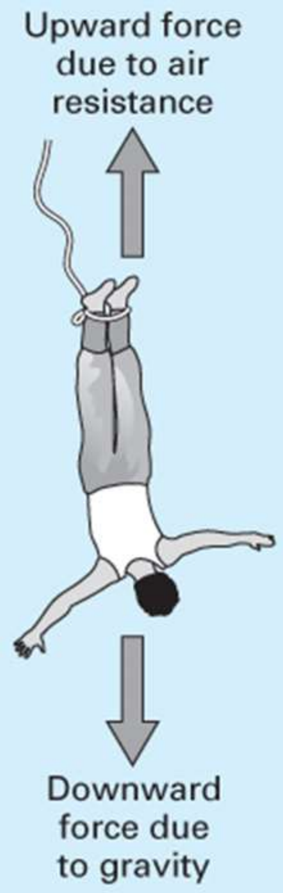
```
import matplotlib.pyplot as plt
import numpy as np

# Create a column vector t that contains values from 0
to 20 in steps of 0.5
t = np.arange(0, 20, 0.5)

#Assign values to the parameters
g = 9.81; m = 68.1; cd = 0.25;

# Evaluate the formula v = f (t)
v = np.sqrt(g*m/cd)*np.tanh(np.sqrt(g*cd/m)*t)
```

$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}}t\right)$$

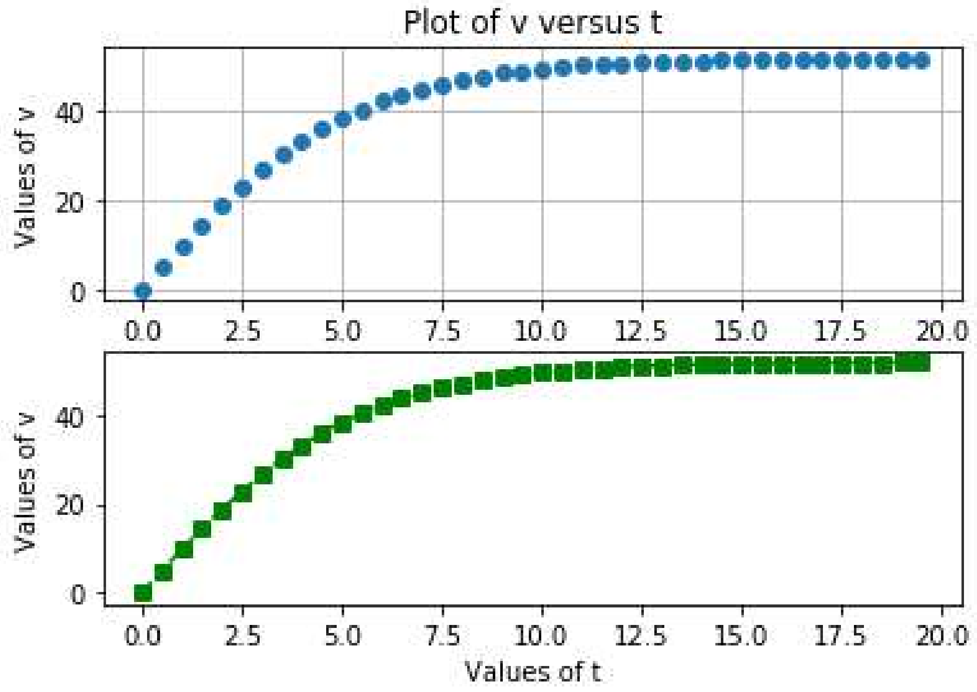

Upward force due to air resistance
Downward force due to gravity

Simulation of bungee jumper model (cont.)

```
# plot with open circles
plt.subplot(211)
plt.plot(t, v, 'o')
plt.title('Plot of v versus t')
plt.xlabel('Values of t')
plt.ylabel('Values of v')
plt.grid(1)

# Plot with square green markers
 connected by green dashed lines
plt.subplot(212)
plt.plot(t, v, 's--g')
plt.xlabel('Values of t')
plt.ylabel('Values of v')

plt.show()
```


Plot of v versus t
Values of v
Values of v
Values of t