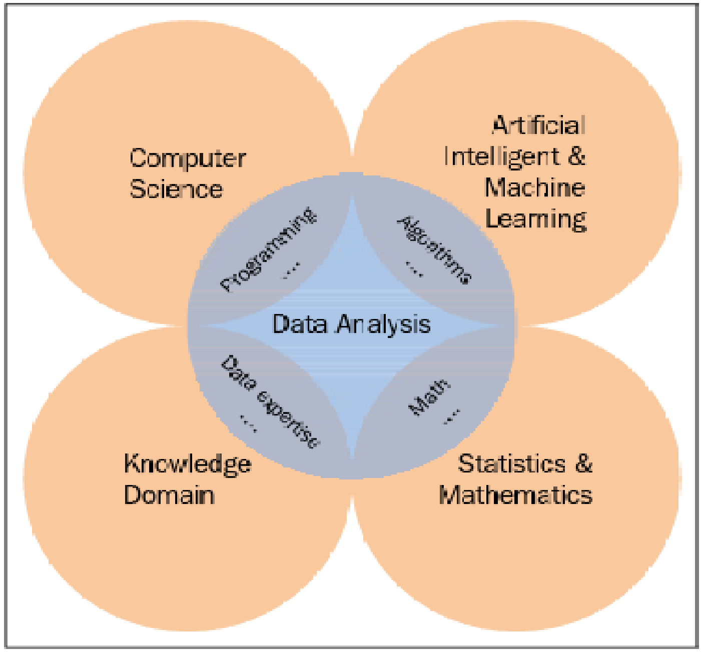
Data analytics in Python Programming

Week 10:Data Analytics

Learning objectives

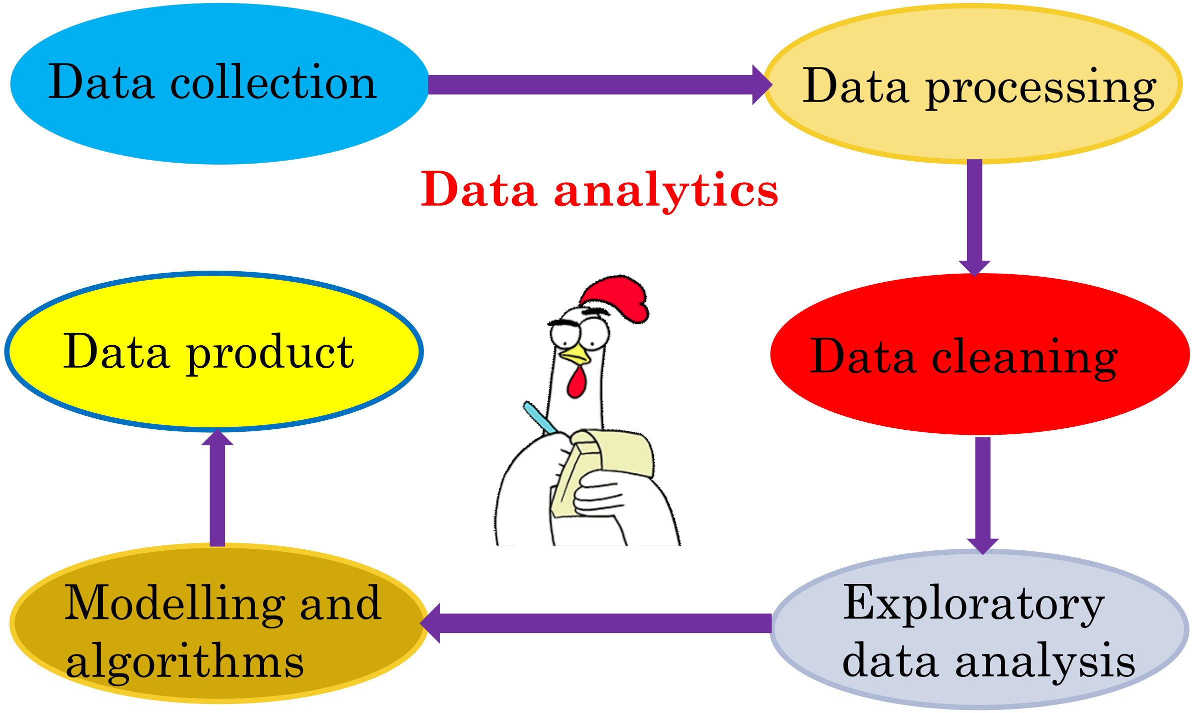
- ❑ Basics of data analytics
- ❑ Data analytics from the programmer's perspectives
- ❑ To know programming styles
- ❑ Data analytics and Python programming

At the conclusion of this lecture, students will be able to understand the fundamentals of data analytics and writing simple data analytics based programs in Python



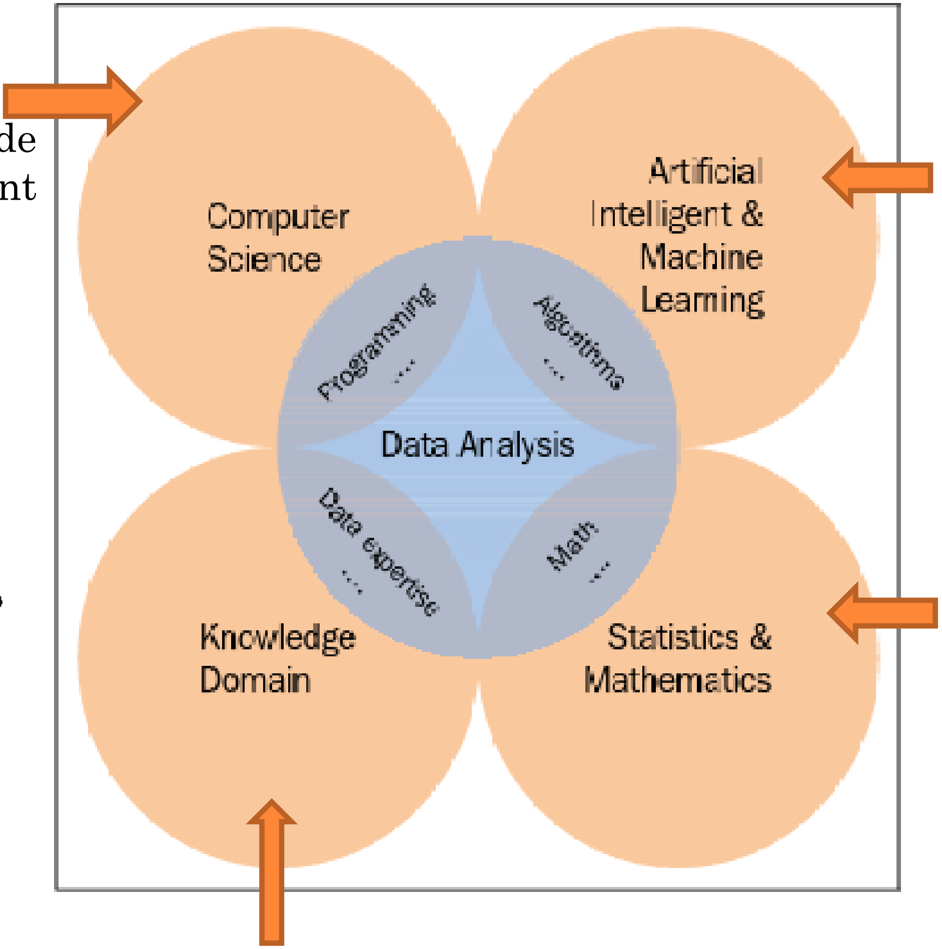
Ref: **Python : data analytics and visualization : understand, evaluate, visualize data**
Vo. T. H, Phuong, author.; Vo. T. H, Phuong, author.

- **What is data analytics?**
- It refers techniques that applied to analyse and to build data products that receive data input and generate output according to the problem requirements.
- The function of data analytics includes data collection, data processing, data cleaning, exploratory data analysis, modelling and algorithms, and data product.



Domain knowledge required for data analysis

Computer science knowledge required to develop data analysis tools, in order to provide abstractions for efficient data processing.

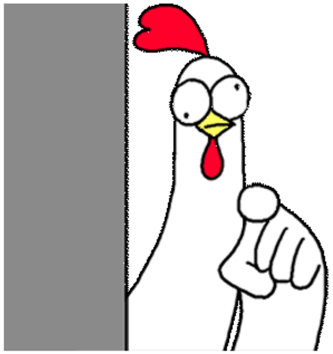


Knowing **AI and machine learning algorithms** helps to proceed predictive analytics (Predictive models) in order to provide possible future outcomes

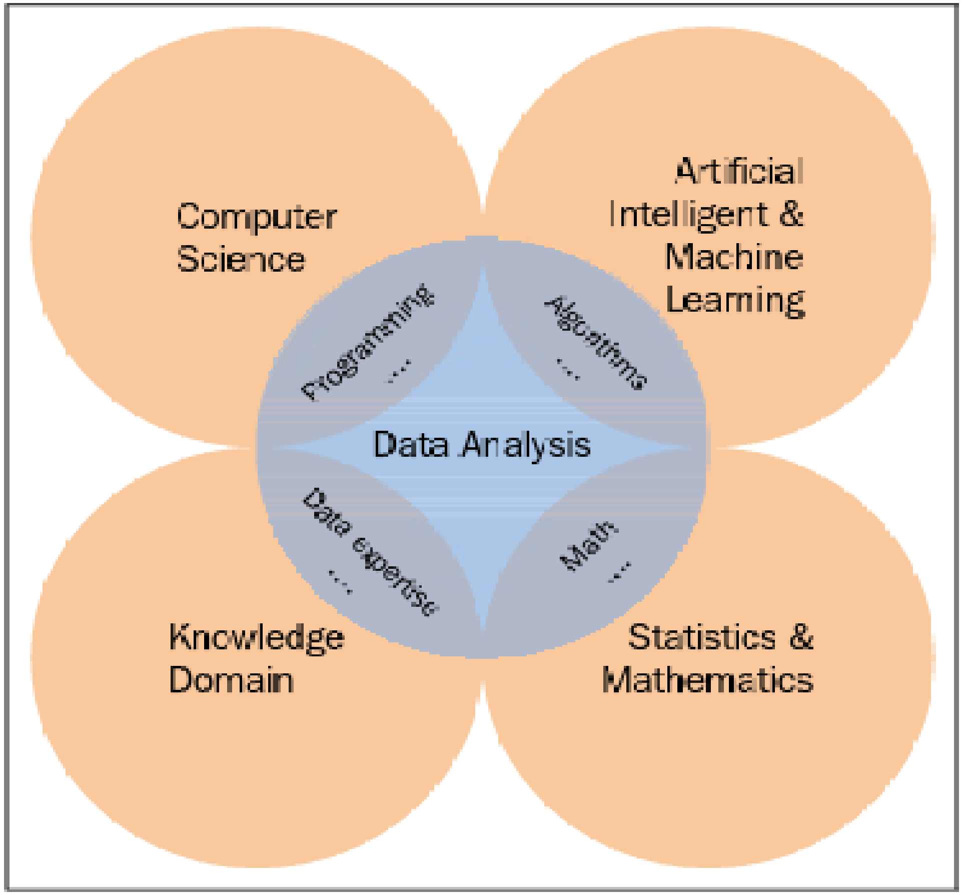
Knowledge in **mathematics and statistics** is required to convert raw data into meaningful information

Domain knowledge is required to understand the nature of data
Example: learning analytics, business analytics, health data analytics…..

- **Data analytics-based jobs and responsibilities**



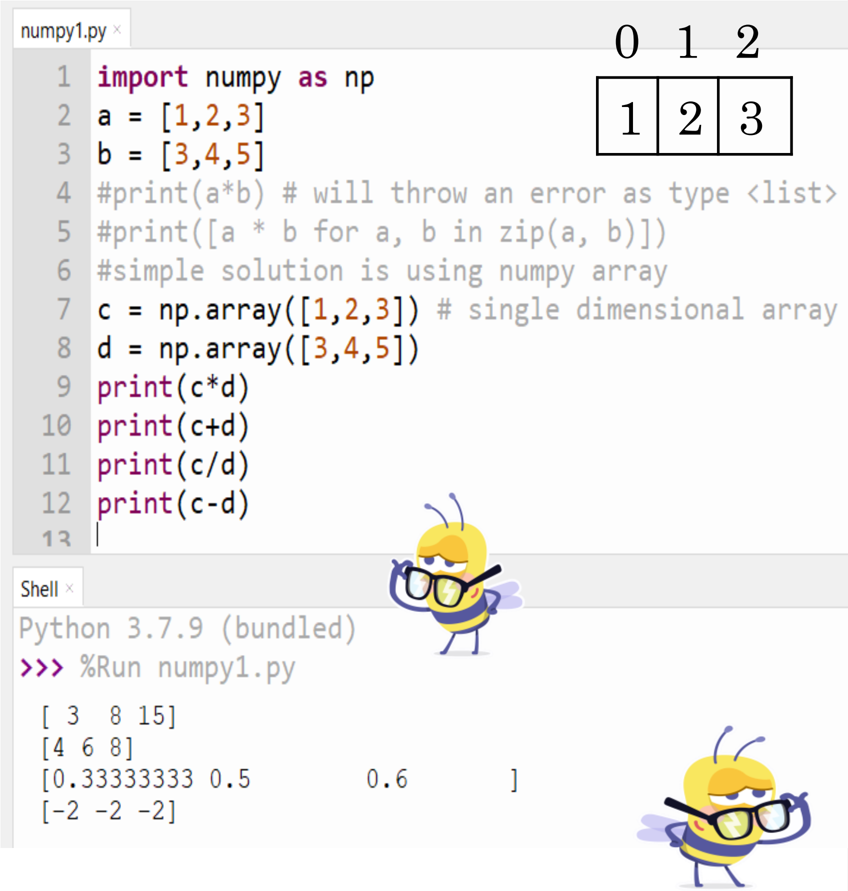
Where do you fit here as a programmer/data analyst/data scientist/ data mining engineer/ data architect/ machine learning engineer?



- **Data analytics and Python**
- In general Python and R programming languages are frequently used for data analytics.

- **Python libraries for data analysis**
- NumPy (Numerical Python)
- Pandas (Python data analysis)
- Matplotlib (Plotting library)
- And more…

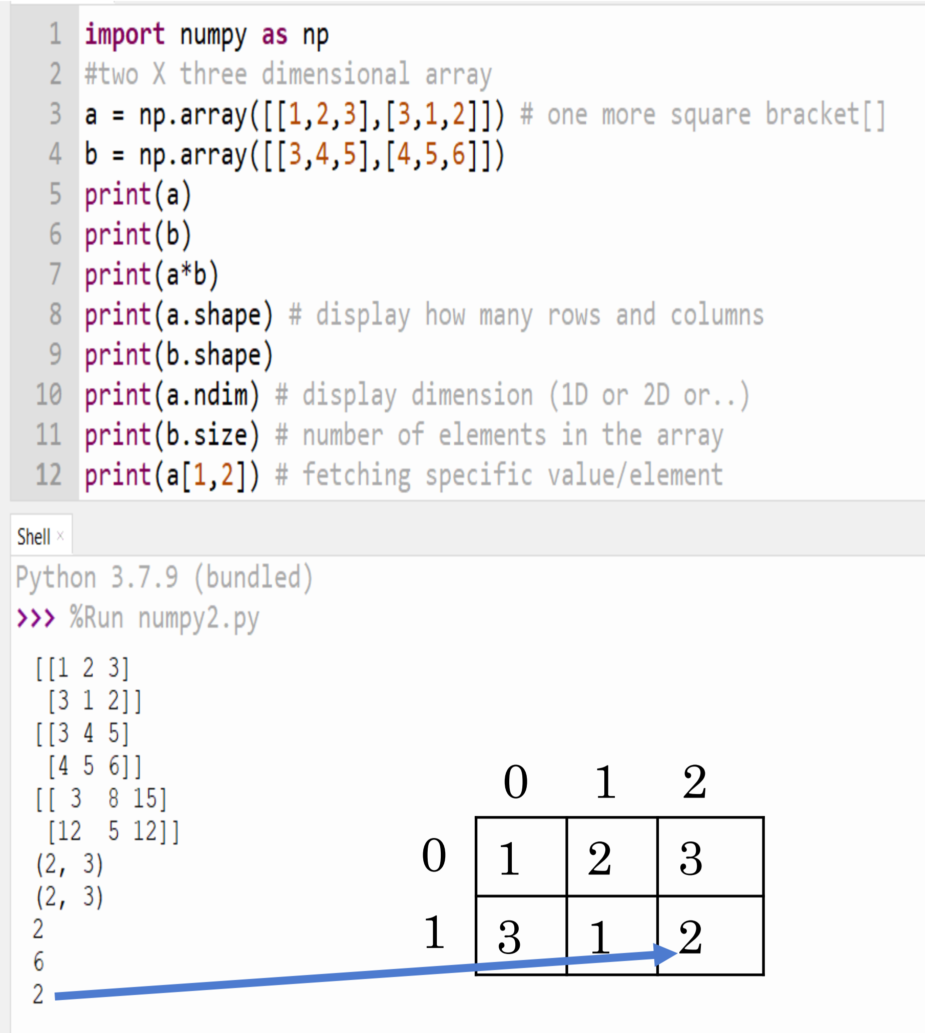
- **Numerical Python (NumPy)**
- It contains powerful functions that handle 1-n-dimensional array objects
- In addition, it contain functions that can handle linear algebra operations, Fourier transformations (decomposition of image), and random number capabilities.



```python
import numpy as np
a = [1,2,3]
b = [3,4,5]
#print(a*b) # will throw an error as type <list>
#print([a * b for a, b in zip(a, b)])
#simple solution is using numpy array
c = np.array([1,2,3]) # single dimensional array
d = np.array([3,4,5])
print(c*d)
print(c+d)
print(c/d)
print(c-d)

```

0	1	2
1	2	3

```
Python 3.7.9 (bundled)
>>> %Run numpy1.py

 [ 3  8 15]
 [4 6 8]
 [0.33333333 0.5        0.6       ]
 [-2 -2 -2]
```

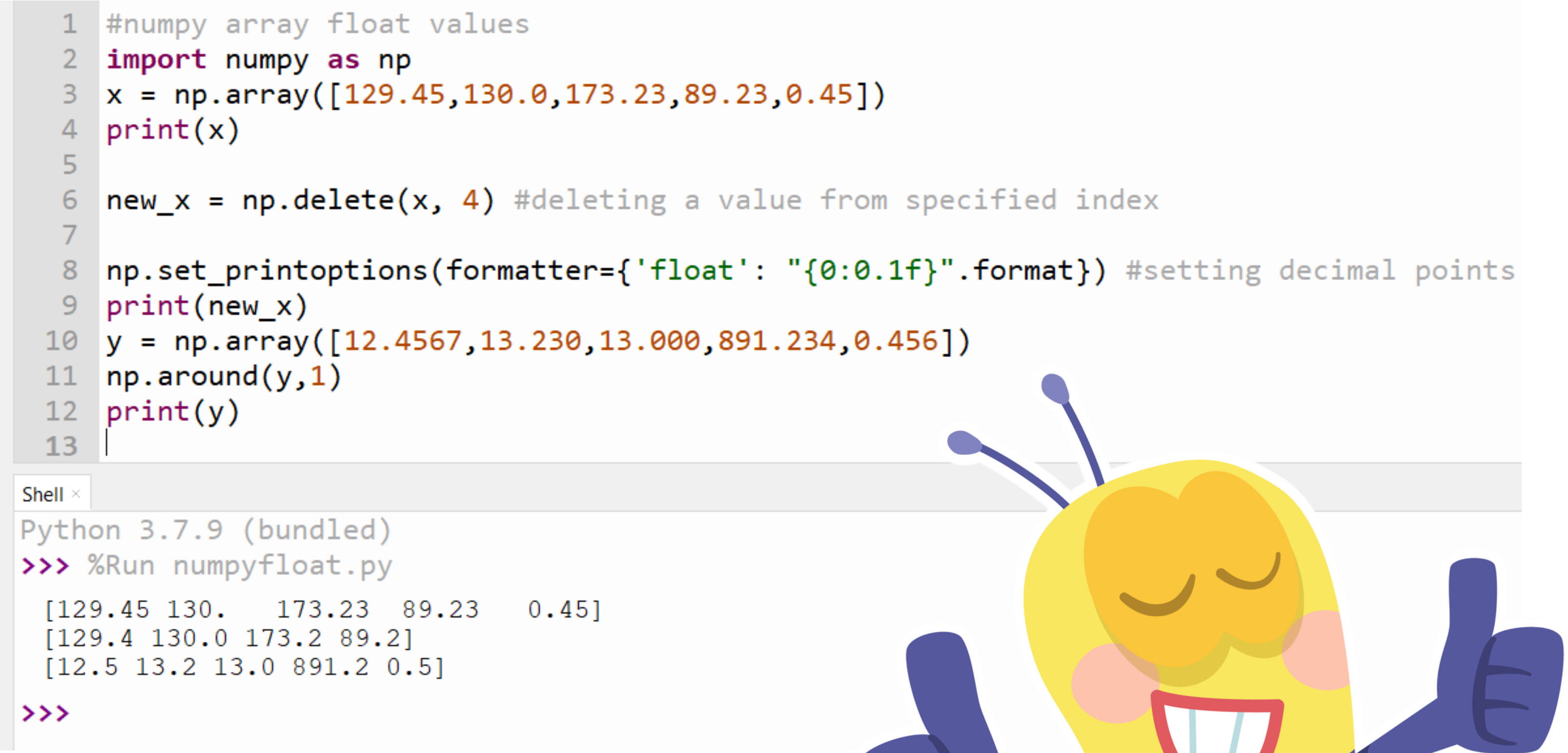


```python
import numpy as np
#two X three dimensional array
a = np.array([[1,2,3],[3,1,2]]) # one more square bracket[]
b = np.array([[3,4,5],[4,5,6]])
print(a)
print(b)
print(a*b)
print(a.shape) # display how many rows and columns
print(b.shape)
print(a.ndim) # display dimension (1D or 2D or..)
print(b.size) # number of elements in the array
print(a[1,2]) # fetching specific value/element
```

```
Python 3.7.9 (bundled)
>>> %Run numpy2.py

 [[1 2 3]
  [3 1 2]]
 [[3 4 5]
  [4 5 6]]
 [[ 3  8 15]
  [12  5 12]]
 (2, 3)
 (2, 3)
 2
 6
 2
```

	0	1	2
0	1	2	3
1	3	1	2

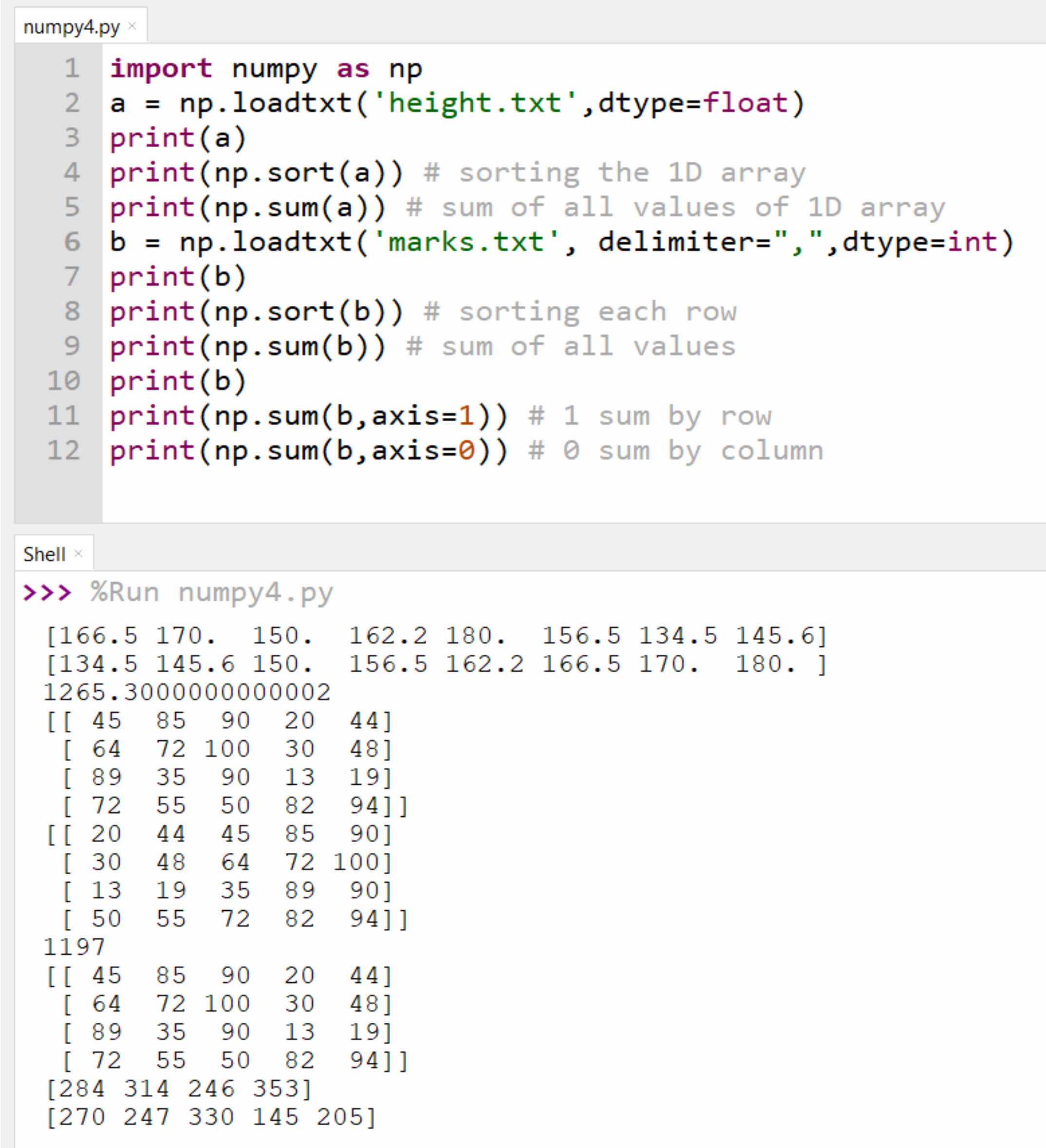
NymPy array floating point values



```python
#numpy array float values
import numpy as np
x = np.array([129.45,130.0,173.23,89.23,0.45])
print(x)

new_x = np.delete(x, 4) #deleting a value from specified index

np.set_printoptions(formatter={'float': "{0:0.1f}".format}) #setting decimal points
print(new_x)
y = np.array([12.4567,13.230,13.000,891.234,0.456])
np.around(y,1)
print(y)

```

```
Python 3.7.9 (bundled)
>>> %Run numpyfloat.py

 [129.45 130.    173.23   89.23    0.45]
 [129.4 130.0 173.2 89.2]
 [12.5 13.2 13.0 891.2 0.5]

>>>
```

- **Reading data from file and converting as NumPy array**

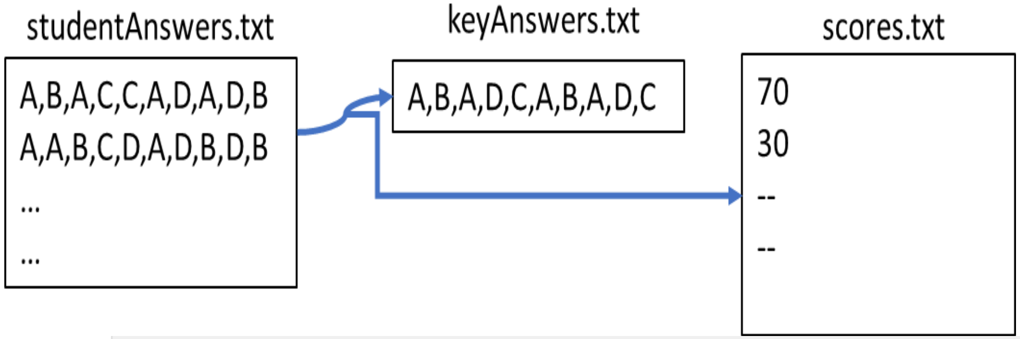


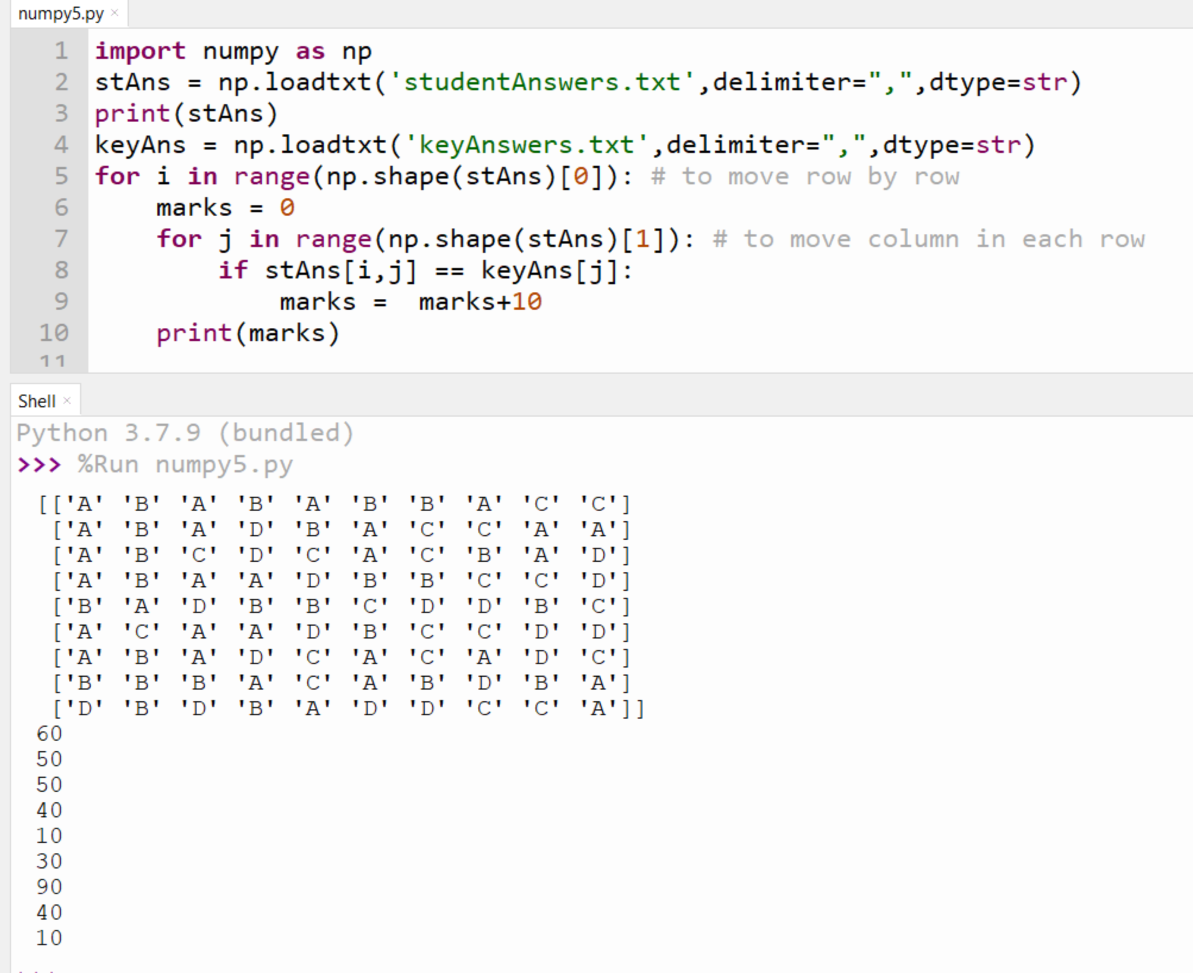
```python
import numpy as np
a = np.loadtxt('height.txt',dtype=float)
print(a)
print(np.sort(a)) # sorting the 1D array
print(np.sum(a)) # sum of all values of 1D array
b = np.loadtxt('marks.txt', delimiter=",",dtype=int)
print(b)
print(np.sort(b)) # sorting each row
print(np.sum(b)) # sum of all values
print(b)
print(np.sum(b,axis=1)) # 1 sum by row
print(np.sum(b,axis=0)) # 0 sum by column
```

```
>>> %Run numpy4.py
 [166.5 170.  150.  162.2 180.  156.5 134.5 145.6]
 [134.5 145.6 150.  156.5 162.2 166.5 170.  180. ]
 1265.3000000000002
 [[ 45  85  90  20  44]
  [ 64  72 100  30  48]
  [ 89  35  90  13  19]
  [ 72  55  50  82  94]]
 [[ 20  44  45  85  90]
  [ 30  48  64  72 100]
  [ 13  19  35  89  90]
  [ 50  55  72  82  94]]
 1197
 [[ 45  85  90  20  44]
  [ 64  72 100  30  48]
  [ 89  35  90  13  19]
  [ 72  55  50  82  94]]
 [284 314 246 353]
 [270 247 330 145 205]
```

Reading data and store it as NumPy array

The students' answers for multiple choice exam are stored in a file called ***"studentAnswers.txt"***. The key answer for each question is stored in another file called ***"keyAnswers.txt"***. <u>Do not modify the contents of these text files</u>. Write code that evaluates student answers by using that key answer text given in another file. Then write the final score computed for each student in a new file called ***"scores.txt"***. Then print the final score by invoking the procedure called ***"printScores()"*** at main program*.* Check the example given below. Each row represents a student. In total 10 questions were given for the exam.

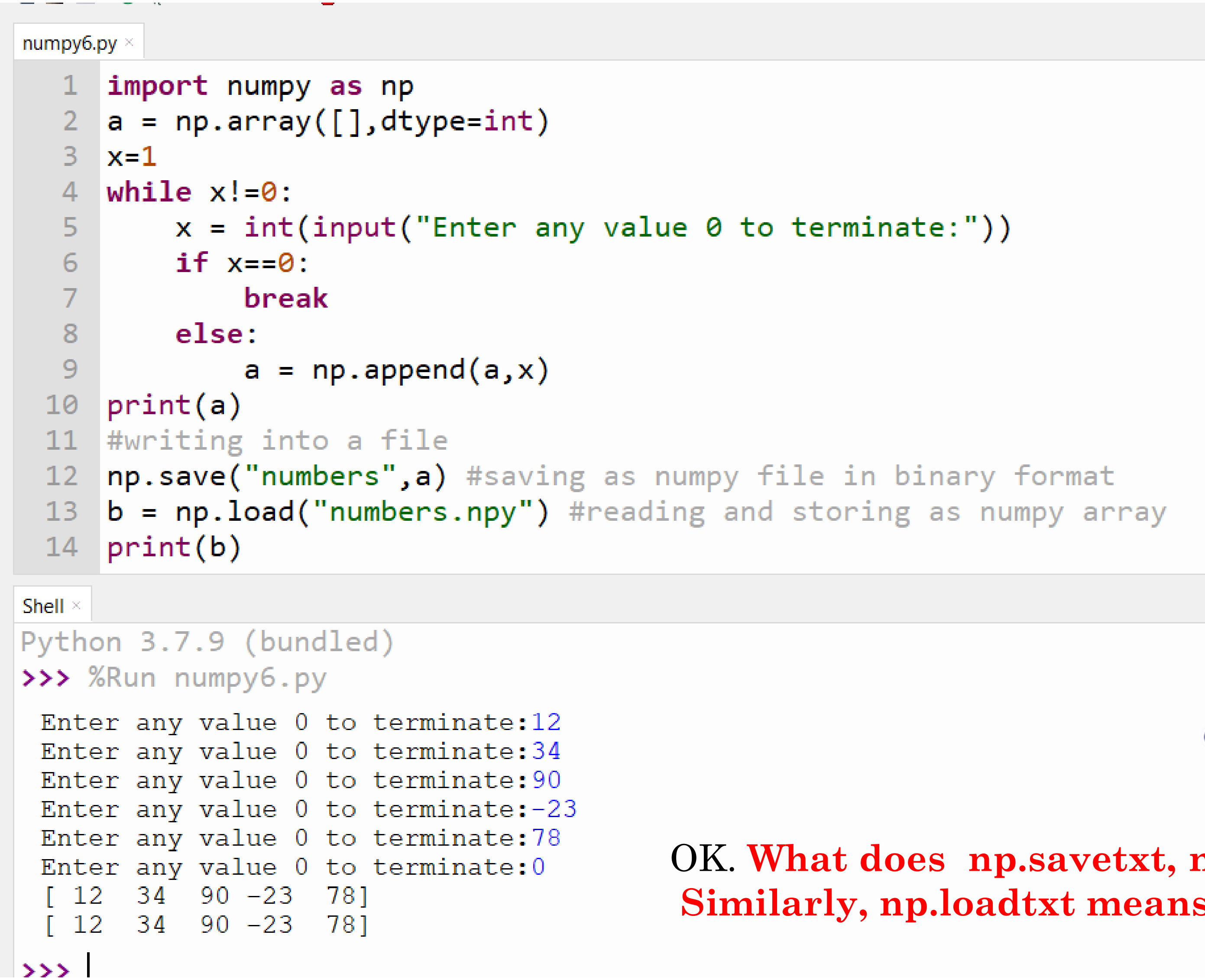




```python
import numpy as np
stAns = np.loadtxt('studentAnswers.txt',delimiter=",",dtype=str)
print(stAns)
keyAns = np.loadtxt('keyAnswers.txt',delimiter=",",dtype=str)
for i in range(np.shape(stAns)[0]): # to move row by row
    marks = 0
    for j in range(np.shape(stAns)[1]): # to move column in each row
        if stAns[i,j] == keyAns[j]:
            marks =  marks+10
    print(marks)
```

```
Python 3.7.9 (bundled)
>>> %Run numpy5.py

 [['A' 'B' 'A' 'B' 'A' 'B' 'B' 'A' 'C' 'C']
  ['A' 'B' 'A' 'D' 'B' 'A' 'C' 'C' 'A' 'A']
  ['A' 'B' 'C' 'D' 'C' 'A' 'C' 'B' 'A' 'D']
  ['A' 'B' 'A' 'A' 'D' 'B' 'B' 'C' 'C' 'D']
  ['B' 'A' 'D' 'B' 'B' 'C' 'D' 'D' 'B' 'C']
  ['A' 'C' 'A' 'A' 'D' 'B' 'C' 'C' 'D' 'D']
  ['A' 'B' 'A' 'D' 'C' 'A' 'C' 'A' 'D' 'C']
  ['B' 'B' 'B' 'A' 'C' 'A' 'B' 'D' 'B' 'A']
  ['D' 'B' 'D' 'B' 'A' 'D' 'D' 'C' 'C' 'A']]
 60
 50
 50
 40
 10
 30
 90
 40
 10
```

- **Saving NumPy array as NumPy file(.npy)**



```
import numpy as np
a = np.array([],dtype=int)
x=1
while x!=0:
    x = int(input("Enter any value 0 to terminate:"))
    if x==0:
        break
    else:
        a = np.append(a,x)
print(a)
#writing into a file
np.save("numbers",a) #saving as numpy file in binary format
b = np.load("numbers.npy") #reading and storing as numpy array
print(b)
```

```
Python 3.7.9 (bundled)
>>> %Run numpy6.py

 Enter any value 0 to terminate:12
 Enter any value 0 to terminate:34
 Enter any value 0 to terminate:90
 Enter any value 0 to terminate:-23
 Enter any value 0 to terminate:78
 Enter any value 0 to terminate:0
 [ 12  34  90 -23  78]
 [ 12  34  90 -23  78]

>>> 
```

OK. **What does np.savetxt, np.savez does?**
Similarly, np.loadtxt means ?

Let's think as data analyst