Python Review

Table of content

Pycharm

Variables

```
print('Hello World’)

Hello = ‘Hello World’

x=3

y = 5

z = x + y

print(z)
```

Lists

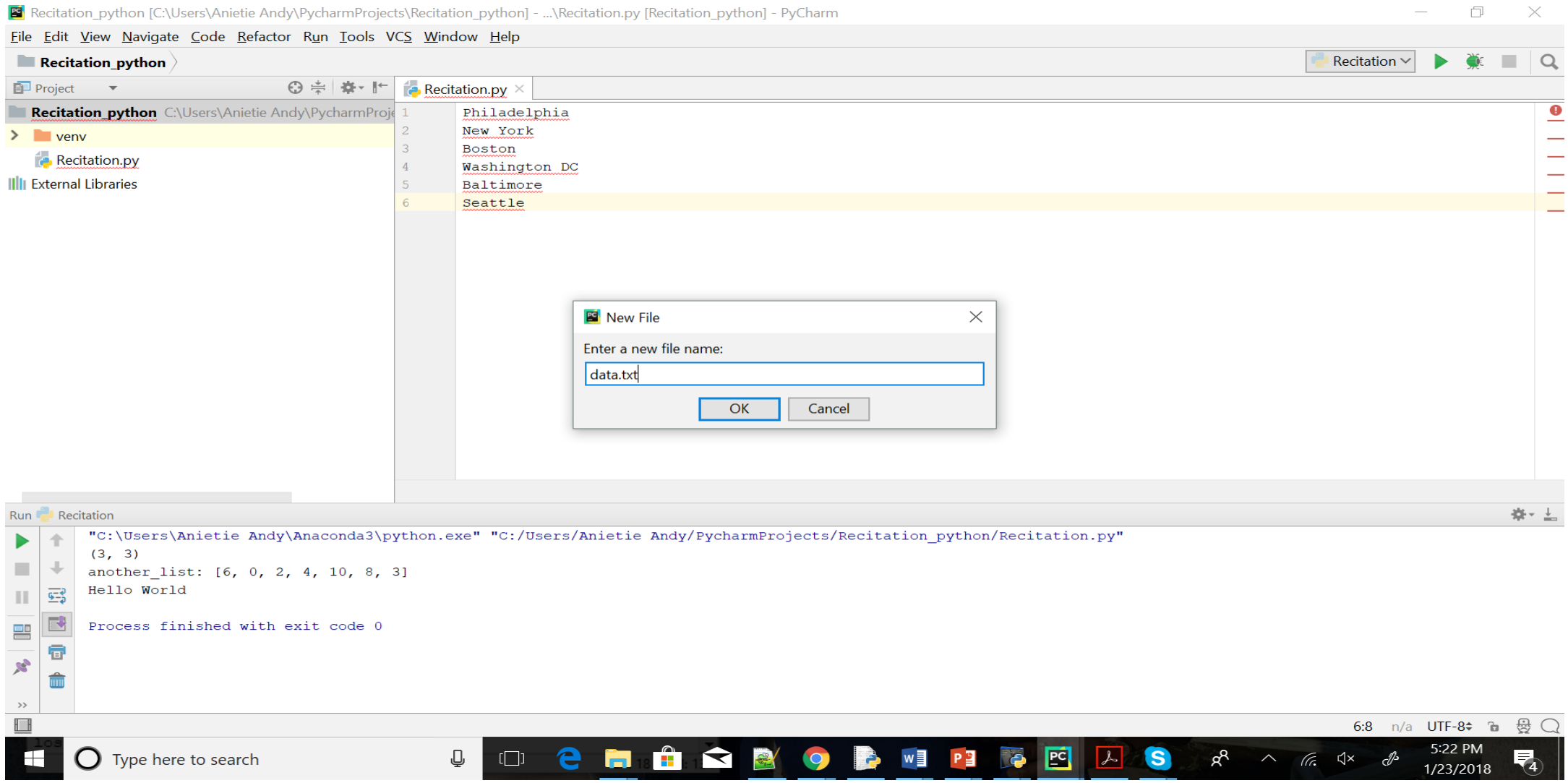
- #Empty string

```
empty = []
empty = list()
zero = [0] * 9
print('Zero:',zero)
empty.append(3)

print('Emtpy:',empty)
print('Length of empty:',len(empty))
```

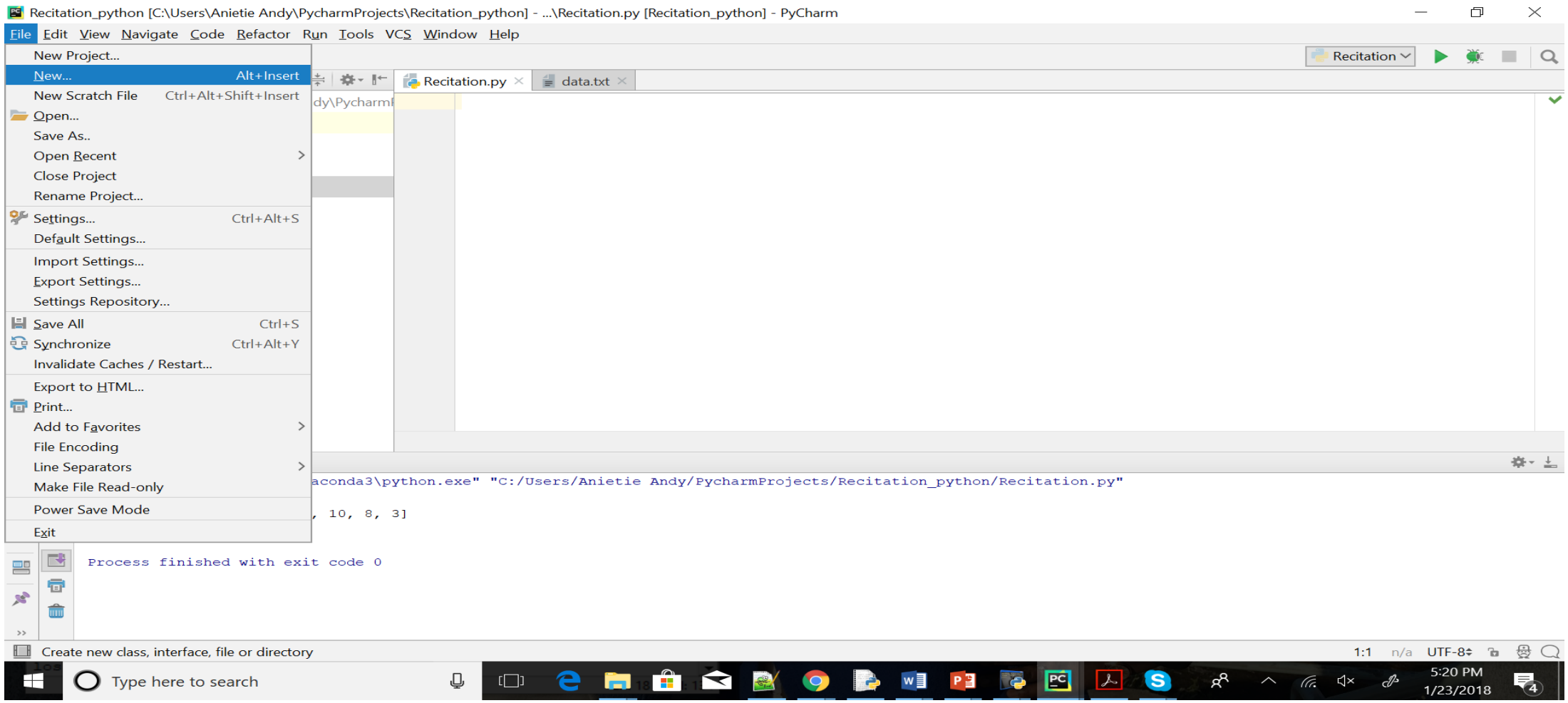
List Indexing

```
list_range = range(8)

another_list = [6,0,2,4,10,8,3]

print('list_range:',list_range)

print('another_list:', another_list)

print('Index 5 of list_range:', list_range[5])

print(another_list[3:6])
```

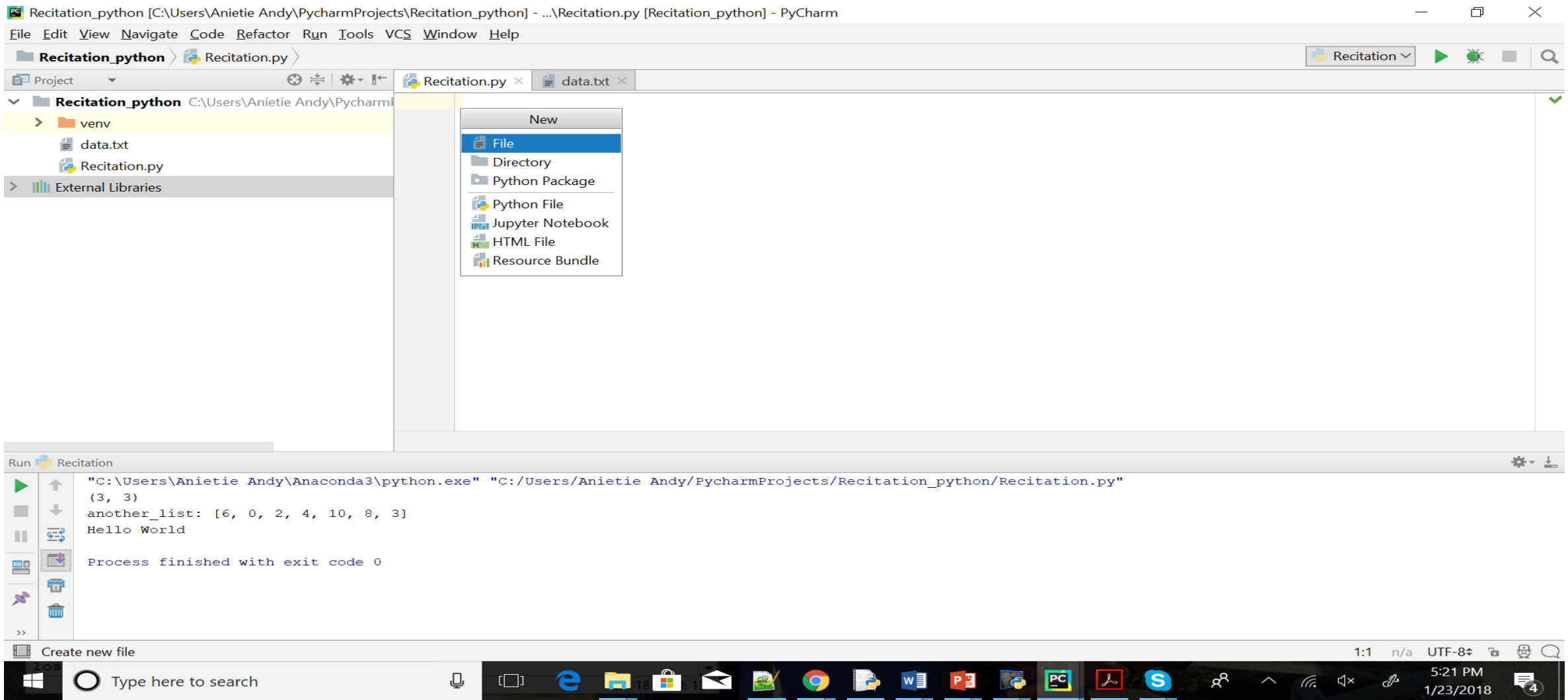
Reading from a file



Reading from a file



Reading from a file



Reading from a file

- Create text file , data.txt

- Philadelphia
- New York
- Boston
- Washington DC
- Baltimore
- Seattle

Reading from a file

```
#Read from a file

data=[]
with open('data.txt') as myfile:
    for line in myfile:
        data.append(line)

print('data:',data)
```

Reading from file

- ```
  #Remove end of line character
  ```
- ```
  data=[]
  with open('data.txt') as myfile:
      for line in myfile:
          data.append(line.rstrip())

  print('data',data)
  ```

Matrixes and Vectors: Numpy

- A scientific computing package for Python

- $\mathbf{M} = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$ $\mathbf{v} = \begin{vmatrix} 1 \\ 2 \\ 3 \end{vmatrix}$

```
import numpy

M = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
v = np.array([[1],[2],[3]])
```

Matrixes and Vectors: Numpy

```
print(M.shape)
(3, 3)

print(v.shape)
(3, 1)

print(v+v)
[[2]
 [4]
 [6]]
```

Other ways of creating Matrices and Vectors

```
a = np.zeros((2,2))   #Create an array of all
zeros
print(a)
```

```
[[ 0.  0.]
 [ 0.  0.]]
```

```
b = np.ones((1,2))    #Create an array of all ones
print(b)
```

```
[[ 1.  1.]]
```

- c = np.full((2,2), 7) #Create a constant array
  print(c)
  [[7. 7.]
   [7. 7.]]

- d = np.eye(2) #Create a 2x2 identity matrix
  print(d)
  [[1. 0.]
  [0. 1.]]

- e = np.random.random((2,2)) #Create an array with random values
- print(e)

  [[0.96285848 0.34266397]
- [0.89280528 0.58975698]]

Element-wise Multiplication

- $\mathbf{M} = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$ $\mathbf{v} = \begin{vmatrix} 1 \\ 2 \\ 3 \end{vmatrix}$

- print(np.multiply(M,v))
  [[1 2 3]
  [8 10 12]
  [21 24 27]]

- print(np.multiply(v,v))
  [[1]
  [4]
  [9]]

Transpose

- print(M)

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

- print(M.T)

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

- print(v)

  [[1]
  [2]
  [3]]

- print(v.T)

  [[1 2 3]]

Stochastic Gradient Descent

- from sklearn.linear_model import SGDClassifier

```
X = [[0., 0.], [1., 1.]]
y = [0, 1]
clf = SGDClassifier(loss="hinge", penalty="l2")
clf.fit(X, y)
```

- Predict new values

```
clf.predict([[2., 2.]]
```

End

Dictionaries

```
#Empty dictionary
empty_dict=dict()
another_empty_dict = {}

#Adding values to the dictionary
non_empty_dict = {1:'CIS', 2:'419/519'}

#printing values of the dictionary
print('empty_dict:',empty_dict)

print('non_empty_dict:', non_empty_dict[1] + non_empty_dict[2])
```

Dictionaries (continued)

```
print('non_empty_dict:', non_empty_dict.keys())

print('non_empty_dict:', non_empty_dict.values())

#Adding a value to an existing dictionary
non_empty_dict[3] = 'Recitation'
```

Dictionaries (continued)

```
#Print dictionary
print('non_empty_dict:', non_empty_dict)

#Deleting from a dictionary
del non_empty_dict[3]

print('non_empty_dict:',non_empty_dict)
```