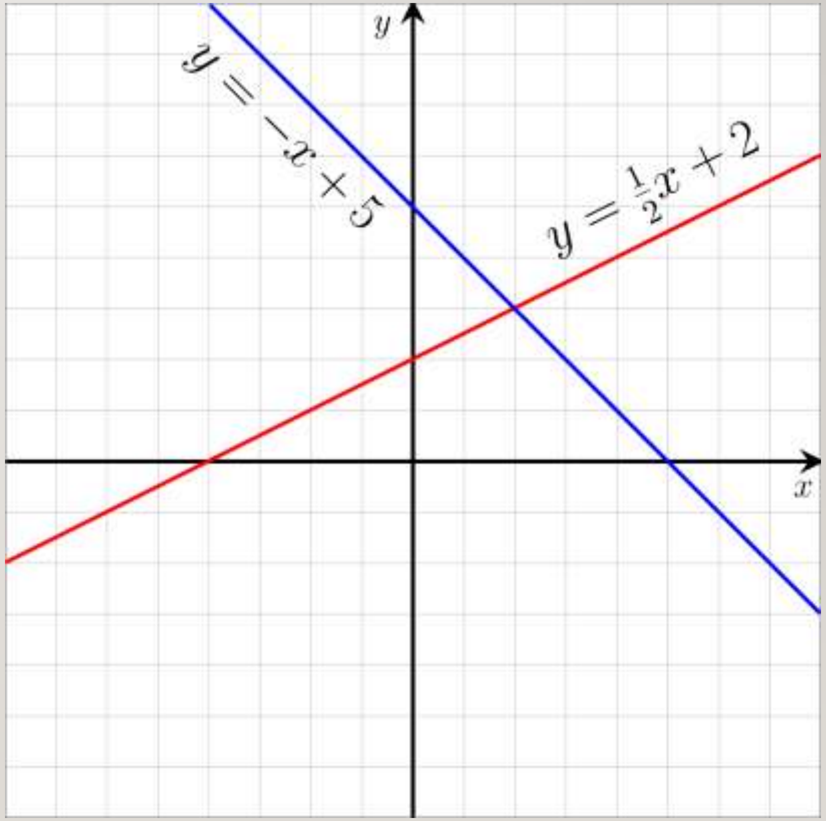
LINEAR ALGEBRA AND DESCRIPTIVE STATISTICS

LINEAR ALGEBRA

- Linear Algebra is a branch of mathematics which deals with linear equations:

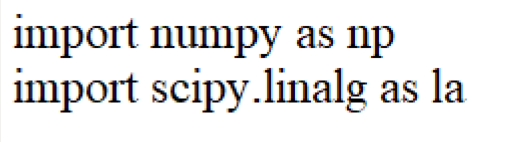
$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

Represented in the form of vectors and matrices.



LINEAR ALGEBRA IN PYTHON

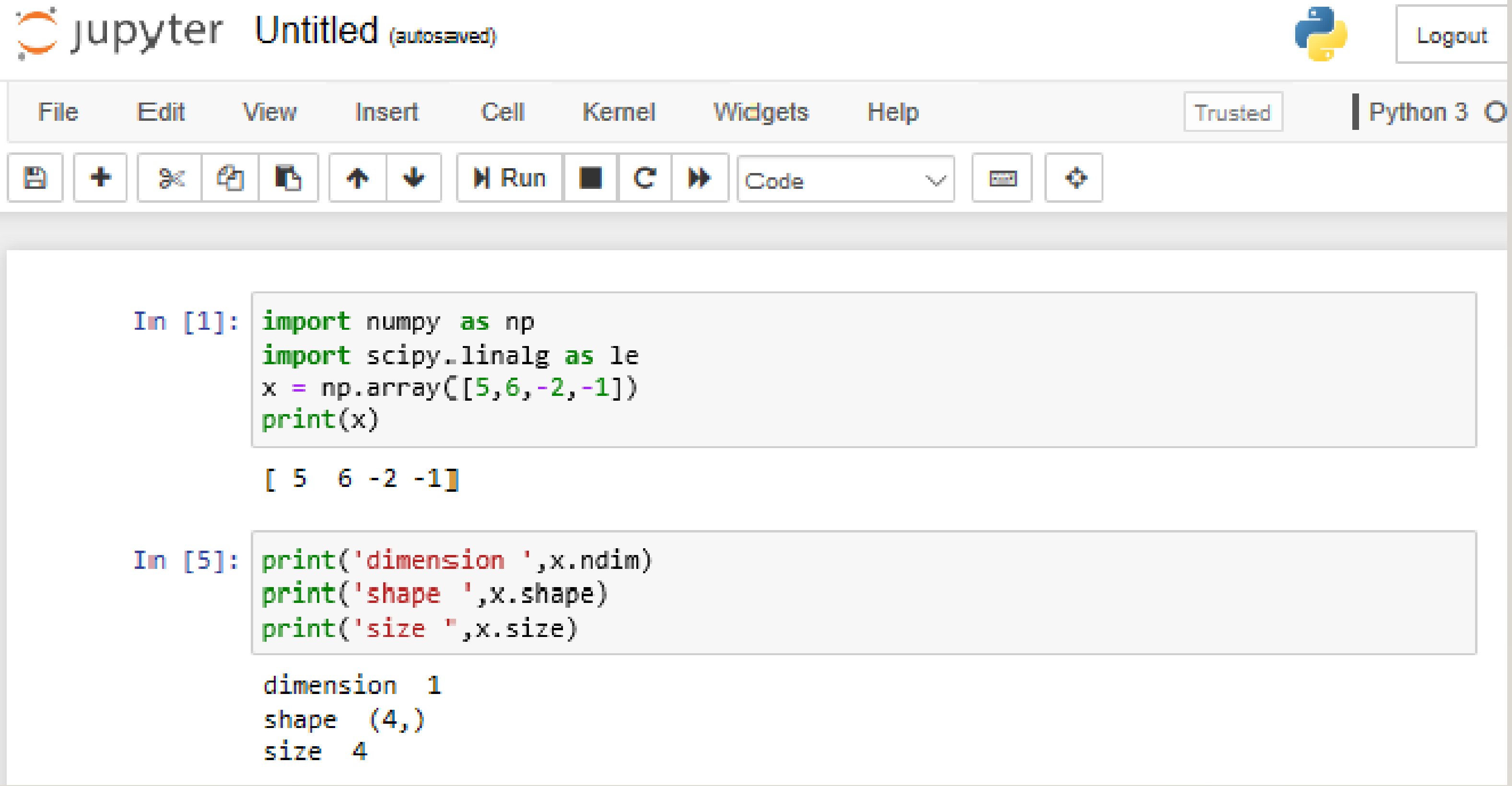
- The main package for linear algebra in python is 'SCIPY' which builds on 'NUMPY' therefore both of the libraries are imported.



```
import numpy as np
import scipy.linalg as la
```

- The 'numpy' library is used to create the arrays and perform the array operations of python while the 'scipy' library is use to perform linear algebraic operations

CREATING AND OPERATING ON 1-D ARRAY



jupyter Untitled (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

In [1]:

```
import numpy  as np
import scipy.linalg as le
x = np.array([5,6,-2,-1])
print(x)
```

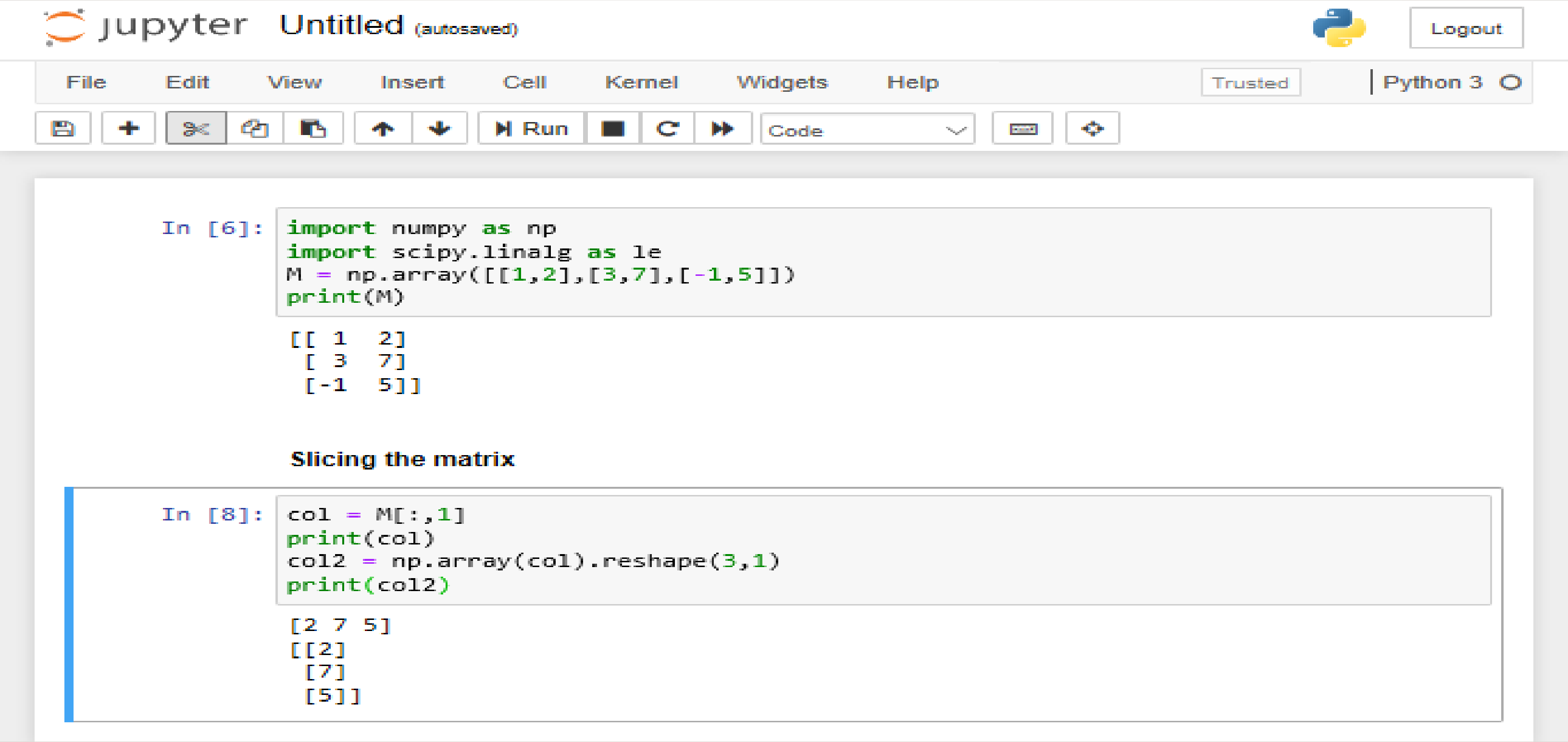
```
[ 5  6 -2 -1]
```

In [5]:

```
print('dimension ',x.ndim)
print('shape ',x.shape)
print('size ',x.size)
```

```
dimension  1
shape  (4,)
size  4
```

CREATING AND OPERATING ON 2-D ARRAY


jupyter Untitled (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

In [6]:
```
import numpy as np
import scipy.linalg as le
M = np.array([[1,2],[3,7],[-1,5]])
print(M)
```

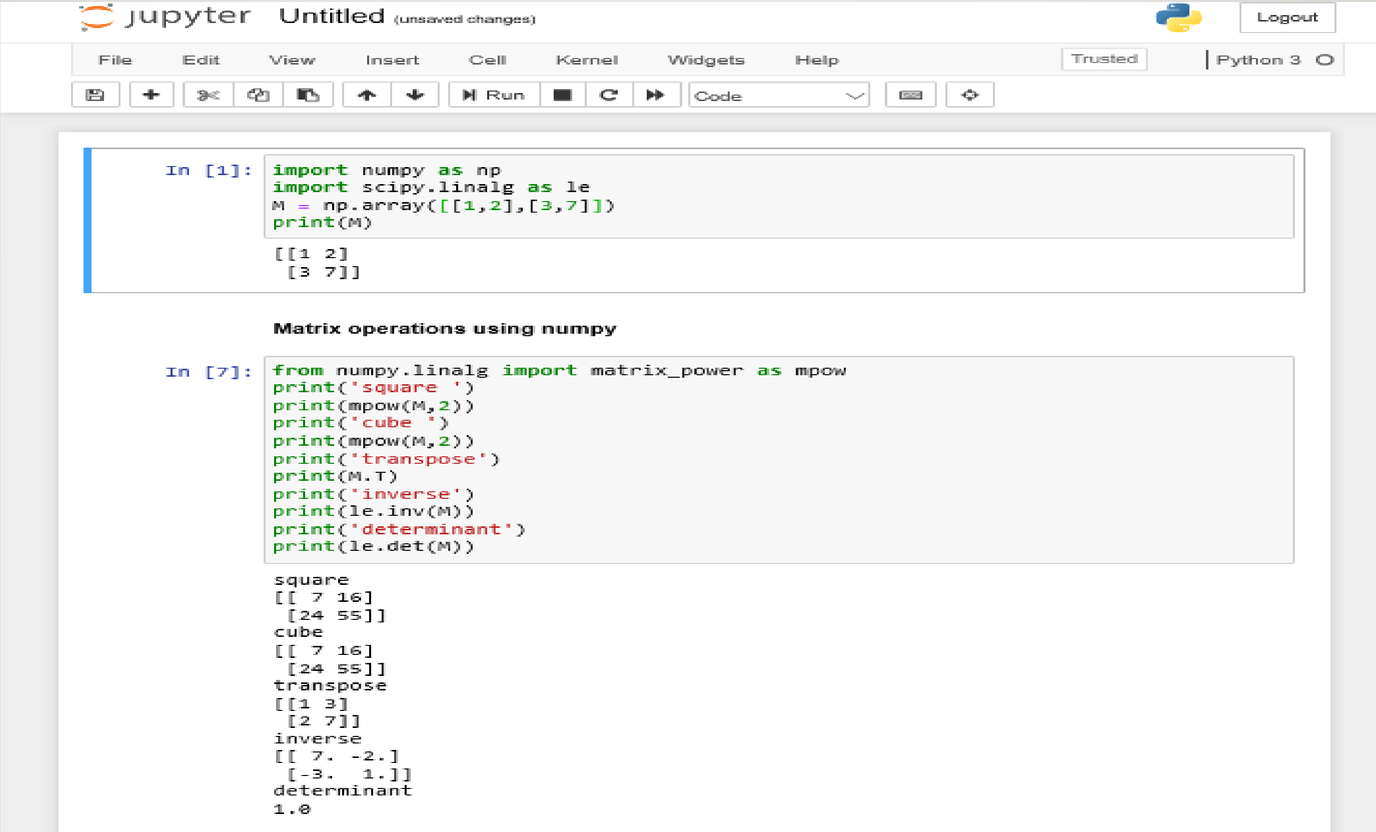
```
[[ 1  2]
 [ 3  7]
 [-1  5]]
```

Slicing the matrix

In [8]:
```
col = M[:,1]
print(col)
col2 = np.array(col).reshape(3,1)
print(col2)
```

```
[2 7 5]
[[2]
 [7]
 [5]]
```

ADVANCED MATRIX OPERATIONS(1/2)


jupyter Untitled (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

In [1]:
```
import numpy as np
import scipy.linalg as le
M = np.array([[1,2],[3,7]])
print(M)
```

```
[[1 2]
 [3 7]]
```

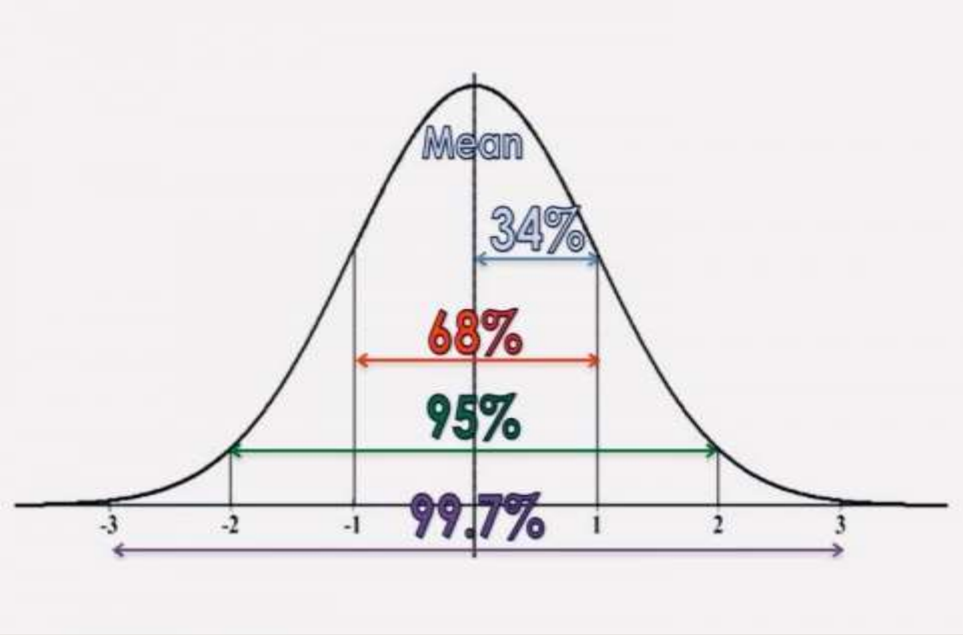
Matrix operations using numpy

In [7]:
```
from numpy.linalg import matrix_power as mpow
print('square ')
print(mpow(M,2))
print('cube ')
print(mpow(M,2))
print('transpose')
print(M.T)
print('inverse')
print(le.inv(M))
print('determinant')
print(le.det(M))
```

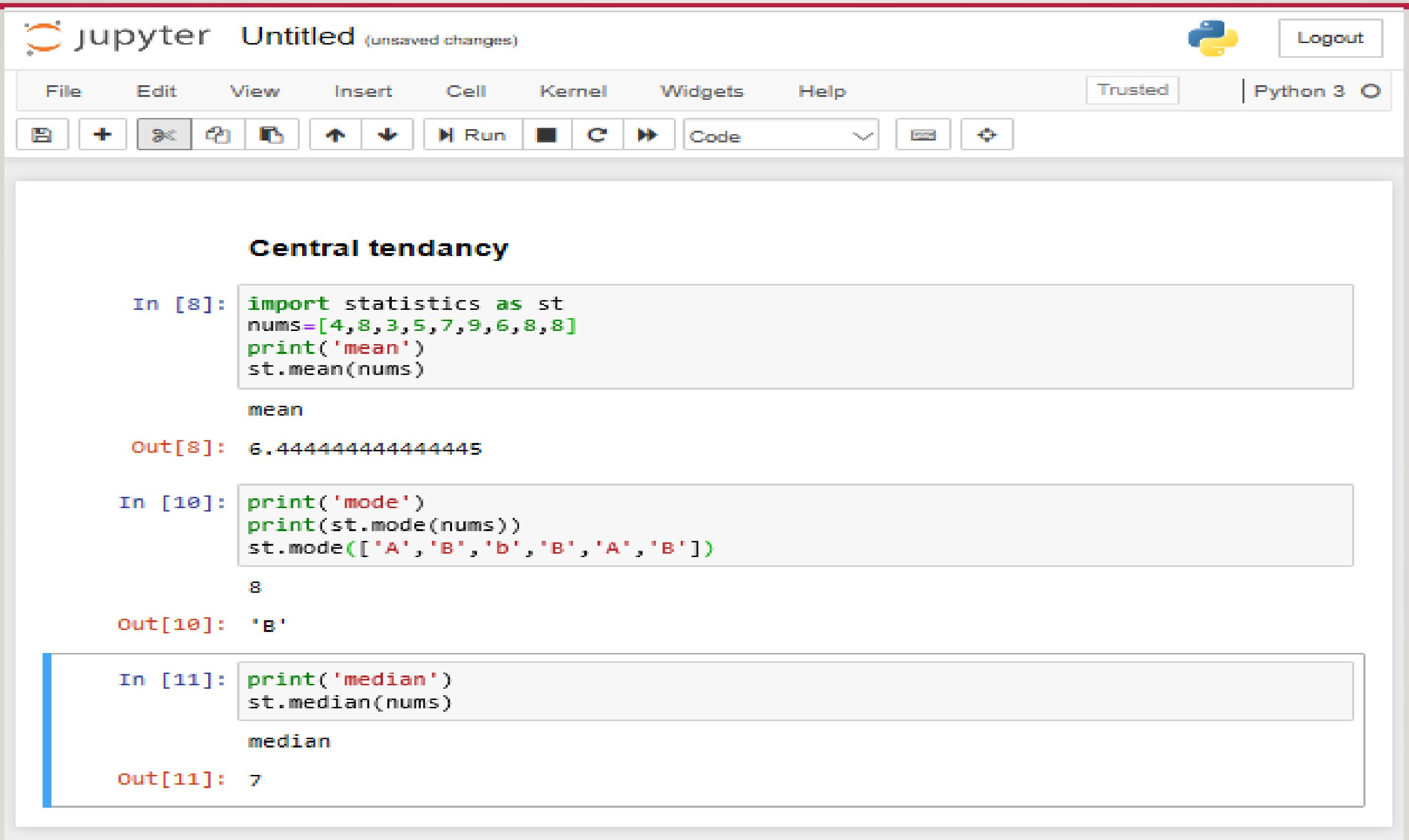
```
square
[[ 7 16]
 [24 55]]
cube
[[ 7 16]
 [24 55]]
transpose
[[1 3]
 [2 7]]
inverse
[[ 7. -2.]
 [-3.  1.]]
determinant
1.0
```

DESCRIPTIVE STATISTICS

- **Descriptive statistics** uses tools like mean and standard deviation on a sample to summarize data.
- **Central** tendency is described by median, mode, and the means.
- **Dispersion** is the degree to which data is distributed around this **central** tendency, and is represented by range, deviation, variance, standard deviation and standard error.



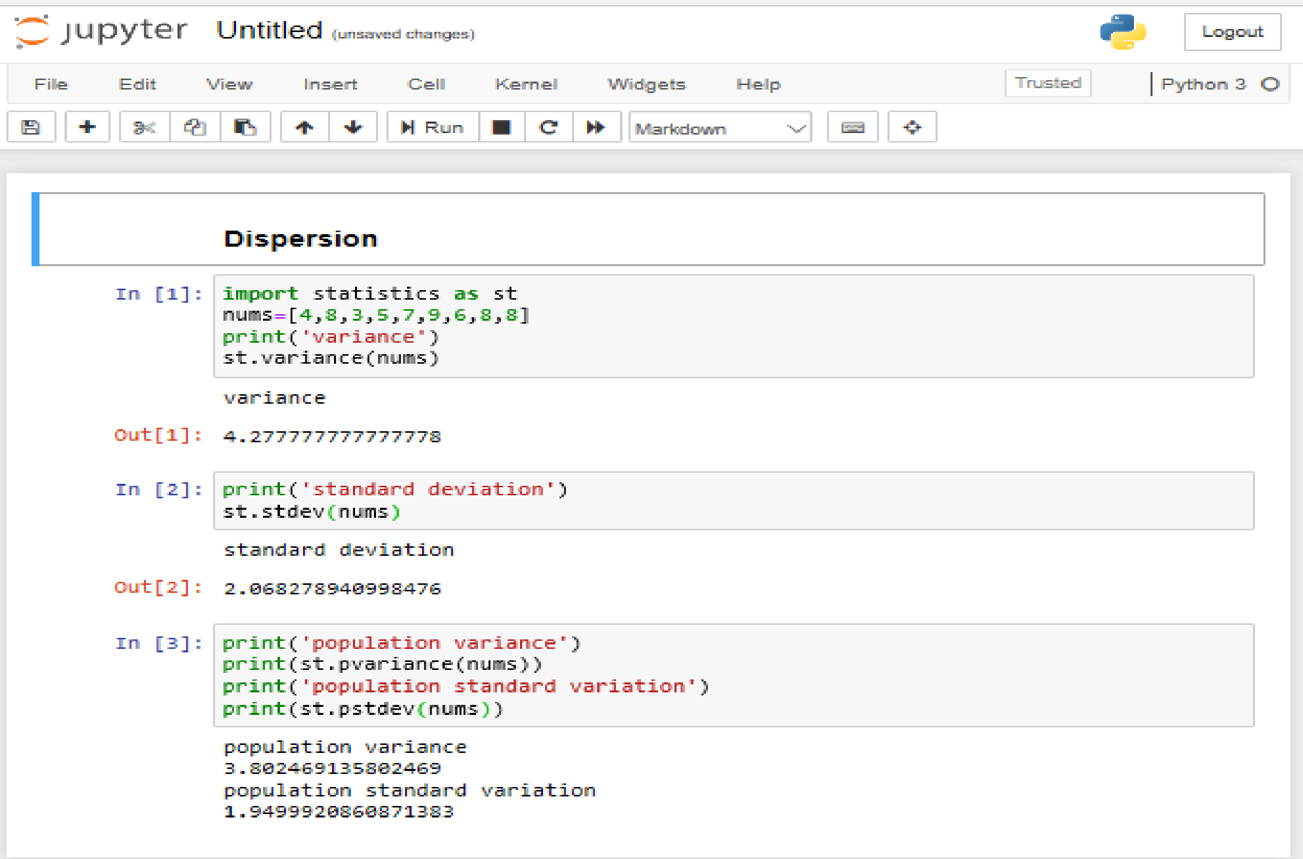
DESCRIPTIVE STATISTICS IN PYTHON(1/2)


jupyter Untitled (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

Central tendancy

```
In [8]:  import statistics as st
         nums=[4,8,3,5,7,9,6,8,8]
         print('mean')
         st.mean(nums)

         mean
Out[8]:  6.444444444444445

In [10]: print('mode')
         print(st.mode(nums))
         st.mode(['A','B','b','B','A','B'])

         8
Out[10]: 'B'

In [11]: print('median')
         st.median(nums)

         median
Out[11]: 7
```

DESCRIPTIVE STATISTICS IN PYTHON(2/2)



Dispersion

In [1]:
```
import statistics as st
nums=[4,8,3,5,7,9,6,8,8]
print('variance')
st.variance(nums)
```

```
variance
```

Out[1]:
```
4.277777777777778
```

In [2]:
```
print('standard deviation')
st.stdev(nums)
```

```
standard deviation
```

Out[2]:
```
2.068278940998476
```

In [3]:
```
print('population variance')
print(st.pvariance(nums))
print('population standard variation')
print(st.pstdev(nums))
```

```
population variance
3.802469135802469
population standard variation
1.9499208608871383
```

THANK YOU