Data Science(Week1)

Python Programming Language :Types and Sequences

>>>type('this is data science lab')

<class 'str'>

>>> type(1)

<class 'int'>

>>> type(1.4)

<class 'float'>

>>> l=[1,2,34,4]

>>> type(l)

<class 'list'>

>>> l.append(3.5)

>>> l

[1, 2, 34, 4, 3.5]

>>> for i in l:

print(i)

o\p:

1

2

34

4

3.5

>>> j=0

>>> while(j !=len(l)):

print(l[j])

j=j+1

o/p:

1

2

34

4

3.5

>>> [1,2] +[3,4]

[1, 2, 3, 4]

>>> [5] \* 3

[5, 5, 5]

>>> i in l

True

>>> x = 'this is string'

>>> print(x[0])

t

>>> print(x[0:1])

t

>>> print(x[0:2])

th

>>> print(x[-4:-1])

rin

>>> x[:3]

'thi'

>>> x[3:]

's is string'

>>>print(firstname+' '+lastname)

badulla shaik

>>> print(firstname\*3)

Badullabadullabadulla

>>> print('dulla' in firstname)

True

>>> d={"badulla":"badulla@gmail.com","abdul":"abdul@gmail.com"}

>>> d['badulla']

'badulla@gmail.com'

>>> d['shaik']='shaik@gmail.com'

>>> d

{'badulla': 'badulla@gmail.com', 'abdul': 'abdul@gmail.com', 'shaik': 'shaik@gmail.com'}

>>> for name in d:

print(name)

o/p:

badulla

abdul

shaik

>>> for mail in d.values():

print(mail)

o/p:

badulla@gmail.com

abdul@gmail.com

[shaik@gmail.com](mailto:shaik@gmail.com)

>>> for name,email in d.items():

print(name)

print(email)

o/p:

badulla

badulla@gmail.com

abdul

abdul@gmail.com

shaik

shaik@gmail.com

>>> l=('badulla','shaik','badulla@gmail.com')

(fname,lname,email)=l

>>> fname

'badulla'

>>> lname

'shaik'

>>> email

'badulla@gmail.com'

More on Strings:

>>>print('badulla'+2)

Traceback (most recent call last):

File "<pyshell#54>", line 1, in <module>

print('badulla'+2)

TypeError: must be str, not int

>>>print('badulla'+str(2))

badulla2

stmt='Hi {} This is ur mail id {}'.format(d.keys(),d['badulla'])

>>> stmt

"Hi dict\_keys(['badulla', 'abdul', 'shaik']) This is ur mail id [badulla@gmail.com](mailto:badulla@gmail.com)"

Reading and writing CSV files:

import csv

with open('mydata.csv') as mydata:

data=list(csv.DictReader(mydata))

print(type(csv.DictReader(mydata)))

print(type(data))

print(len(data))

print(data[0].keys())

print(sum(float(d['T']) for d in data))

print(set(d['name'] for d in data))

mynames=[]

for d in data:

mynames.append(d['name'])

print(set(mynames))

Date and time:

>>> import time

>>> import time as tm

>>> import datetime as dt

>>> tm.time()

1521617816.6670759

>>> ds=dt.datetime.fromtimestamp(tm.time())

>>> ds

datetime.datetime(2018, 3, 21, 13, 21, 36, 688643)

>>> ds.year

2018

>>> ds.month

3

>>> ds.day

21

>>> ds.hour

13

>>> ds.minute

21

>>> ds.second

36

If I want to know the date of 100 days ago.

>>> deltm=dt.timedelta(days=100)

>>> deltm

datetime.timedelta(100)

>>> todat=dt.date.today()

>>> todat

datetime.date(2018, 3, 21)

>>> todat-deltm

datetime.date(2017, 12, 11)

if you want more on date and time Follow below content

# Python Date and Time

Python provides **time** package to deal with Date and time. It helps to retrieve current date and time and manipulation using built-in methods.

## Retrieve Time

To retrieve current time, Python provides a predefined function localtime(), it receives a parameter time.time(). Here,

time is a module and time() is a function that returns the current system time in number of ticks since 12:00 am, January 1,1970. It is known as epoch.

Tick is simply a floating point number in seconds since epoch.

**Python Retrieving Time Example**

1. **import** time;
2. localtime = time.localtime(time.time())
3. **print**("Current Time is :", localtime)

**Output:**

1. >>>
2. Current Time **is** :time.struct\_time(tm\_year=2014, tm\_mon=6, tm\_mday=18, tm\_hour=12,
3. tm\_min=35, tm\_sec=44, tm\_wday=2, tm\_yday=169, tm\_isdst=0)
4. >>>

**Explanation:**

The time returned is a time structure which includes 9 attributes. These are tabled in the below table.

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| tm\_year | It returns the current year |
| tm\_mon | It returns the current month |
| tm\_mday | It returns the current month day |
| tm\_hour | It returns the current hour. |
| tm\_min | It returns the current minute |
| tm\_sec | It returns current seconds |
| tm\_wday | It returns the week day |
| tm\_yday | It returns the year day. |
| tm\_isdst | It returns -1,0 or 1. |

## Python Formatted Time

Python also supports formatted time. Proceed as follows:

1. Pass the time structure in a predefined function asctime(). It is a function defined in time module.
2. It returns a formatted time which includes Day ,month, date, time and year.
3. Print the formatted time.

**Python Formatted Time Example**

1. **import** time;
3. localtime = time.asctime( time.localtime(time.time()) )
4. **print**("Formatted time :", localtime)

**Output:**

1. >>>
2. Formatted time : Sun Jun 22 18:54:20 2014
3. >>>

## Python Time Module Methods

There are many built in functions defined in time module which are used to work with time.

|  |  |
| --- | --- |
| **Methods** | **Description** |
| time() | It returns floating point value in seconds since epoch i.e.,12:00am, January 1, 1970 |
| asctime(time) | It takes the tuple returned by localtime() as parameter. It returns a 24 character string. |
| sleep(time) | The execution will be stopped for the given interval of time. |
| strptime(String,format) | It returns an tuple with 9 time attributes. It receives an String of date and a format. |
| gtime()/gtime(sec) | It returns struct\_time which contains 9 time attributes. In case seconds are not specified it takes current second from epoch. |
| mktime() | It returns second in floating point since epoch. |
| strftime(format)/strftime(format,time) | It returns time in particular format. If time is not given, current time in seconds is fetched. |

### Python time() Method Example

This method is used to get current time using Python script. See, the following example.

1. **import** time
2. print(time.time())

**Output:**

1. >>>
2. 1403700740.39
3. >>>

### Python asctime(time) Method Example

This method is used to return 24 character Date Time string using Python script. See, the following example.

1. **import** time
2. t = time.localtime()
3. print(time.asctime(t))

**Output:**

1. >>>
2. Wed Jun 25 18:30:25 2014
3. >>>

### Python sleep(time) Method Example

This method is used to stop the execution of script for the given interval of time. See, the following example.

1. **import** time
3. localtime = time.asctime( time.localtime(time.time()) )
4. print(localtime)
5. time.sleep( 10 )
6. localtime = time.asctime( time.localtime(time.time()) )
7. print(localtime)

**Output:**

1. >>>
2. Wed Jun 25 18:15:30 2014
3. Wed Jun 25 18:15:40 2014
4. >>>

### Python strptime(String str,format f) Method Example

This method returns an tuple with 9 time's attributes. It receives an String of date and a format. See, the following example.

1. **import** time
3. timerequired = time.strptime("26 Jun 14", "%d %b %y")
4. print(timerequired)

**Output:**

1. >>>
2. time.struct\_time(tm\_year=2014, tm\_mon=6, tm\_mday=26, tm\_hour=0, tm\_min=0,
3. tm\_sec=0, tm\_wday=3, tm\_yday=177, tm\_isdst=-1)
4. >>>

**Explanation:**

The strptime() takes a String and format as argument. The format refers to String passed as an argument. "%a %b %d %H:%M:%S %Y" are the default directives. There are many other directives which can be used. In the given example we have used three directives: %d%b%y which specifies day of the month, month in abbreviated form and year without century respectively. Some of them are given as:

|  |  |
| --- | --- |
| %a | weekday name. |
| %b | month name |
| %c | date and time |
| %e | day of a month |
| %m | month in digit. |
| %n | new line character. |
| %S | second |
| %t | tab character |

etc...

### Python gtime() Method Example

It returns struct\_time which contains 9 time attributes. In case, seconds are not specified it takes current second from epoch. See, the following example.

1. **import** time
2. print(time.gmtime())

**Output:**

1. >>>
2. time.struct\_time(tm\_year=2014, tm\_mon=6, tm\_mday=28, tm\_hour=9, tm\_min=38, tm\_sec=0,
3. tm\_wday=5, tm\_yday=179, tm\_isdst=0)
4. >>>

### Python mktime() Method Example

It returns second in floating point since epoch. See, the following example.

1. **import** time
2. t = (2014, 2, 17, 17, 3, 38, 1, 48, 0)
3. second = time.mktime( t )
4. **print**(second)

**Output:**

1. >>>
2. 1392636818.0
3. >>>

### Python strftime() Method Example

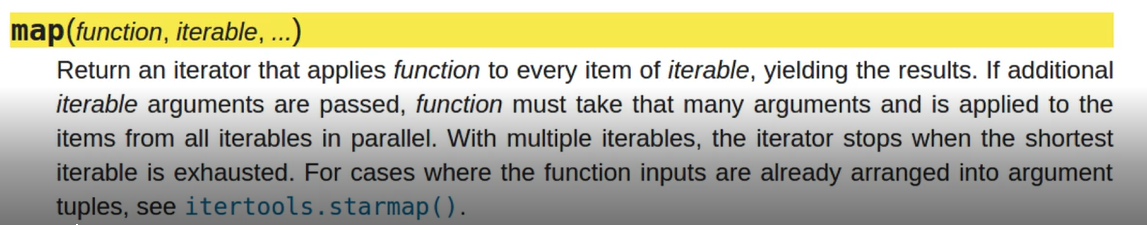
It returns time in particular format. If time is not given, current time in seconds is fetched.. See, the following example.

1. **import** time
2. t = (2014, 6, 26, 17, 3, 38, 1, 48, 0)
3. t = time.mktime(t)
4. print(time.strftime("%b %d %Y %H:%M:%S", time.gmtime(t)))

**Output:**

1. >>>
2. Jun 26 2014 11:33:38
3. >>>

Objects and map():



NumPy:

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed

**Numpy (Numeric Python):**

* Numpy module is used for Arrays concepts. Arrays is a data structure that contains a group of elements. Typically these elements are all of the same data type, such as an integer or sting.
* In python, Numpy module is working in below 3.5 versions without install module but If we want to Numpy module work in above python3.5 version then we have to install numpy module
* Open command prompt then enter below statement

|  |
| --- |
| **python.exe –m pip install numpy** |

**If u got any error in installation of numpy like this..**

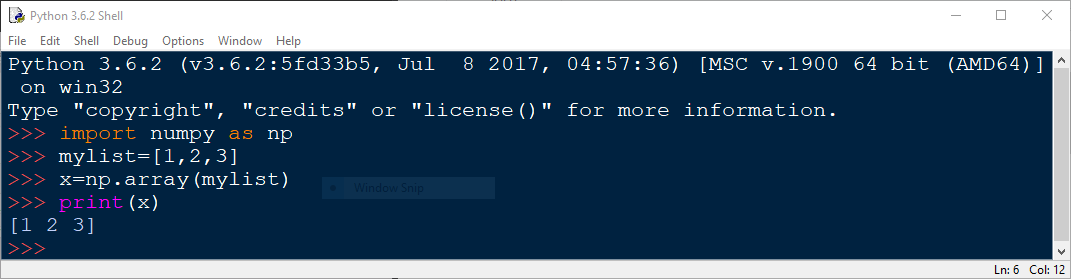
**You are using pip version 9.0.1, however version 9.0.3 is available.**

**You should consider upgrading via the 'python -m pip install --upgrade pip' command.**

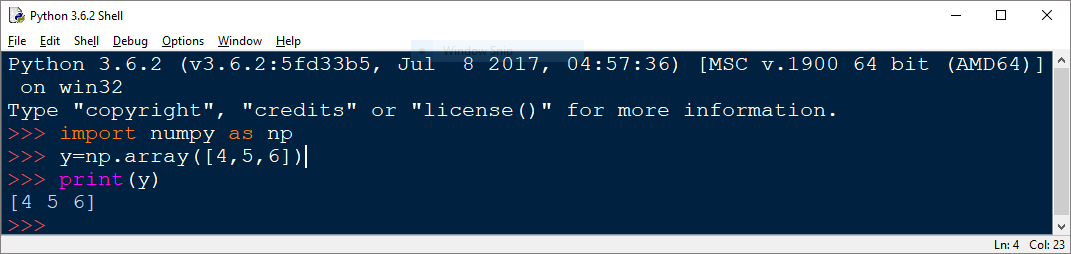
* Now open python shell and import numpy module

|  |
| --- |
| **import numpy as np** |

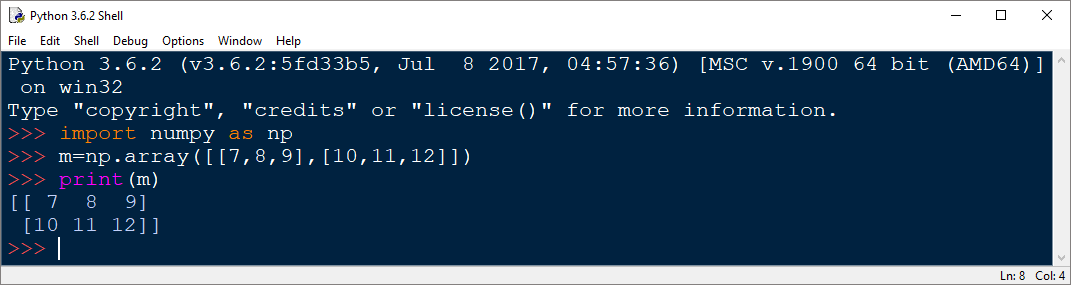
**Creating an array using numpy:**



**Another way to creating an array using numpy:**

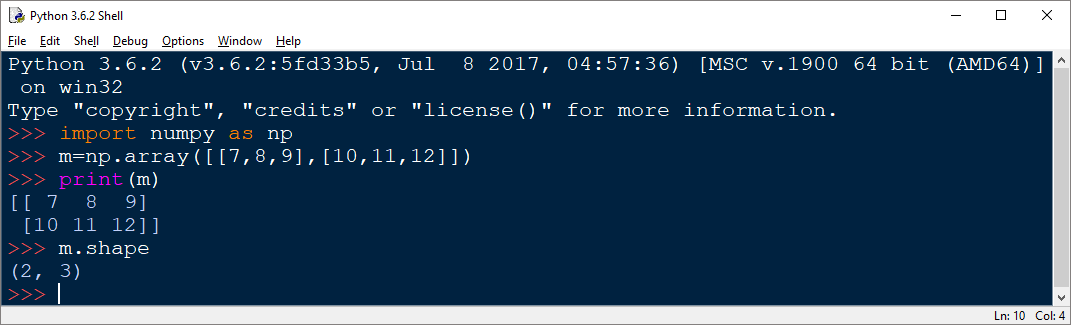


**Creating 2D array:** by using array() method we can generate matrixes.

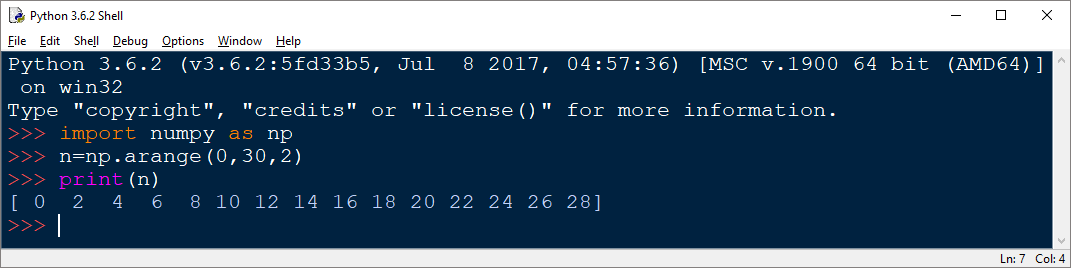


**Shape:** Tuple of array dimensions.

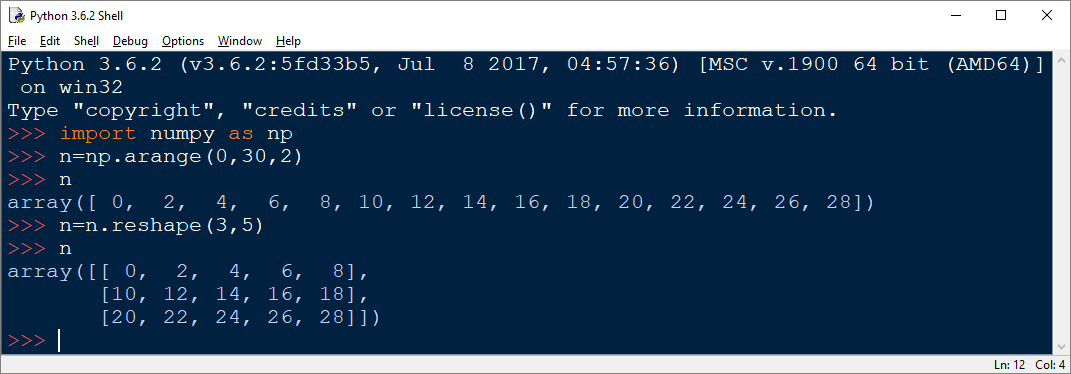
The shape property is usually used to get the current shape of an array.



**A-range:** it will work like as range().see below example.



**Reshape:** it returns the list of elements in the format of rows and columns.



>>> a = np.arange(16).reshape(4, 4)

>>> a

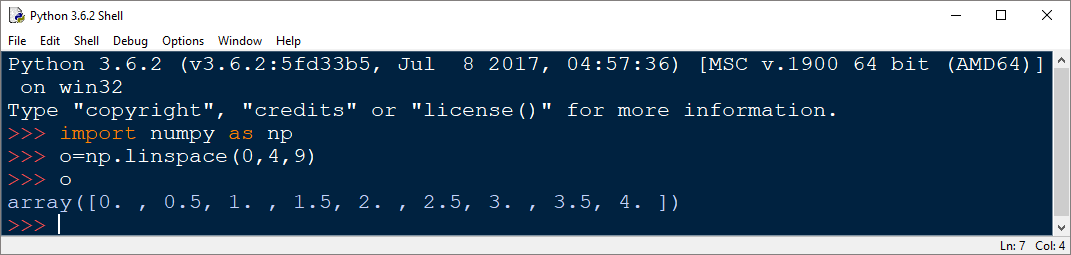
array([[ 0, 1, 2, 3],

[ 4, 5, 6, 7],

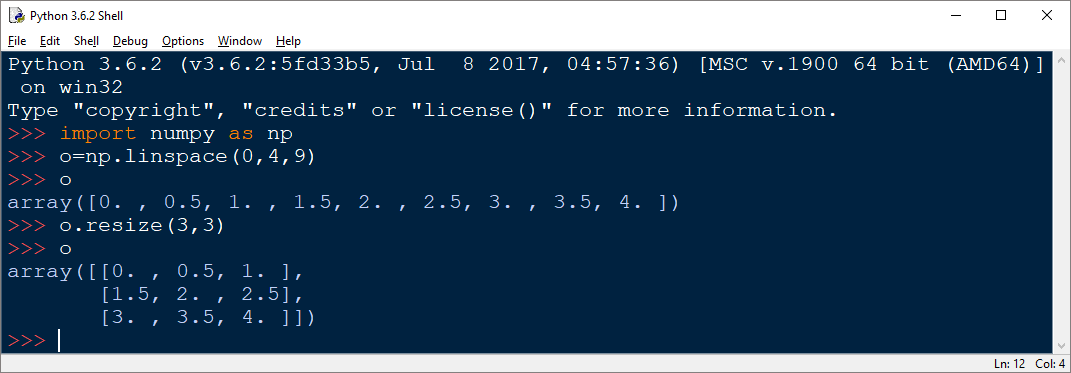
[ 8, 9, 10, 11],

[12, 13, 14, 15]])

**Linspace:** it returns a list at a specified range.

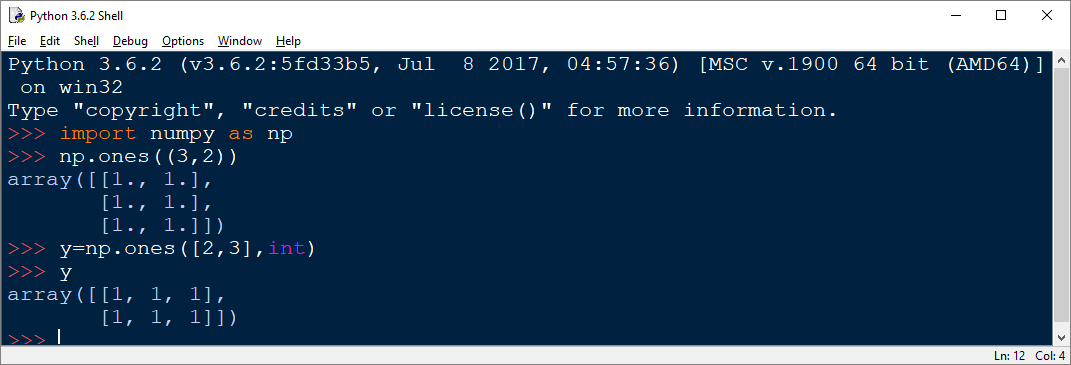


**Resize:** it working like as reshape()

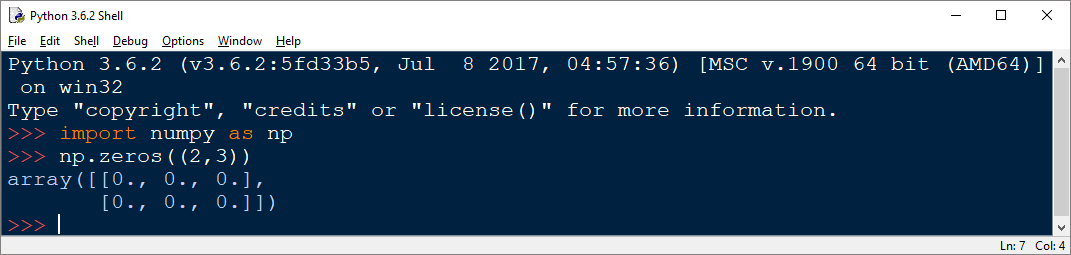


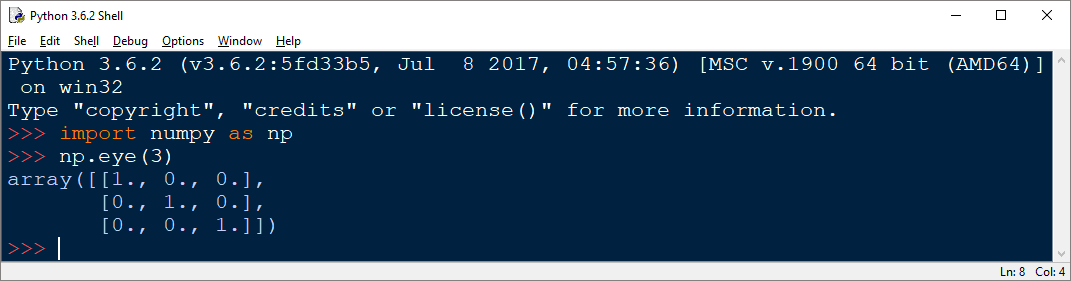
**Built in functions:**

**One’s:** it returns matrix with ones.

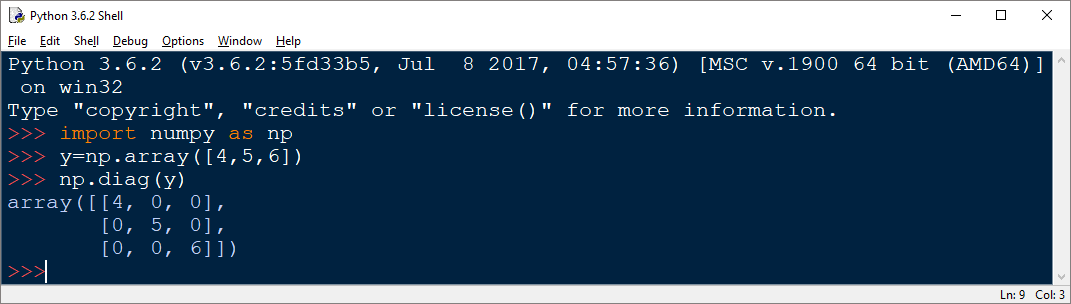


**Zero’s:** it returns matrix with zeros.

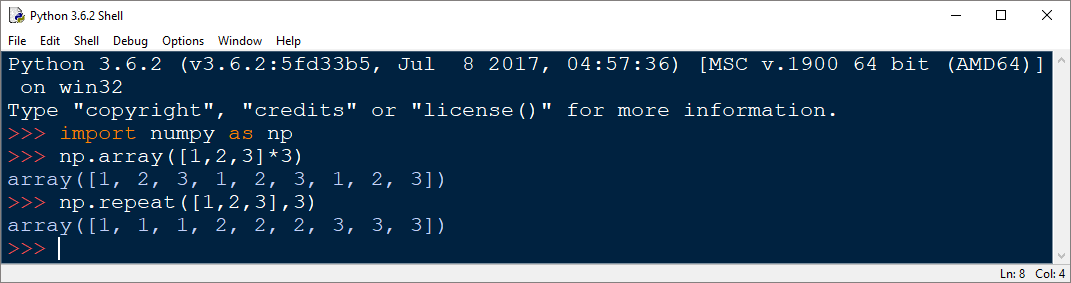


**Eye:** it returns the identity matrix.

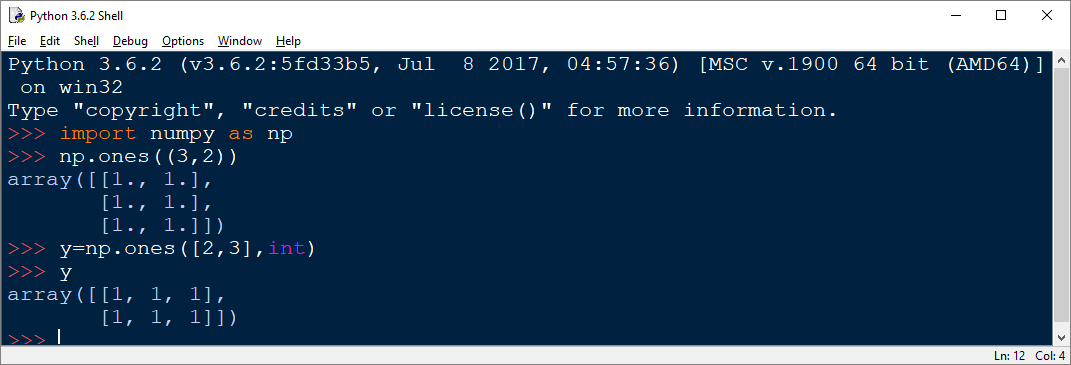
**Diagonals:** Return specified diagonals.



**Repeat:** by using repeat() method we can repeat each and every value at a specific digit in a list .



**V-Stack:** Stack arrays in sequence vertically (row wise).



**>>> n=np.ones([2,3],int)**

**>>> n**

**array([[1, 1, 1],**

**[1, 1, 1]])**

**>>> np.vstack([n,2\*n])**

**array([[1, 1, 1],**

**[1, 1, 1],**

**[2, 2, 2],**

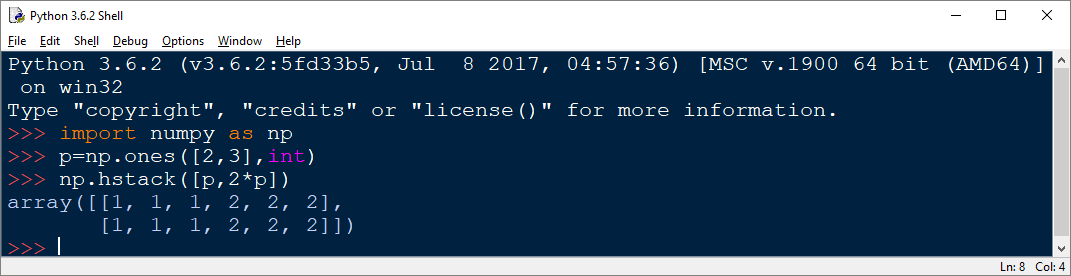
**[2, 2, 2]])**

**>>> np.vstack([2\*n])**

**array([[2, 2, 2],**

**[2, 2, 2]])**

**H-Stack:** Stack arrays in sequence horizontally (column wise).



**>>> np.hstack([2\*n])**

**array([[2, 2, 2],**

**[2, 2, 2]])**

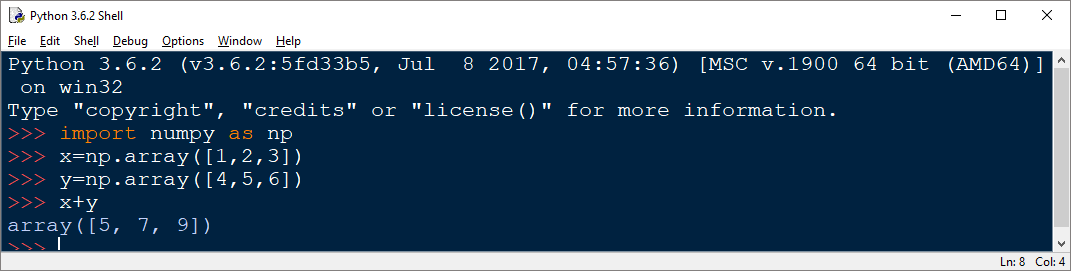
**>>> np.hstack([n,2\*n])**

**array([[1, 1, 1, 2, 2, 2],**

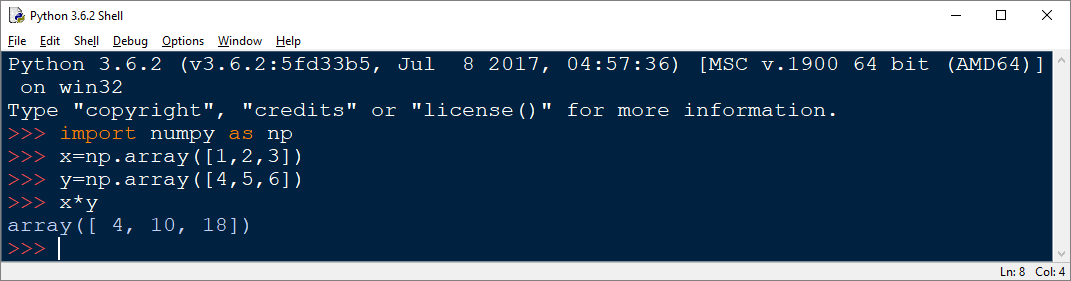
**[1, 1, 1, 2, 2, 2]])**

**Operations:**

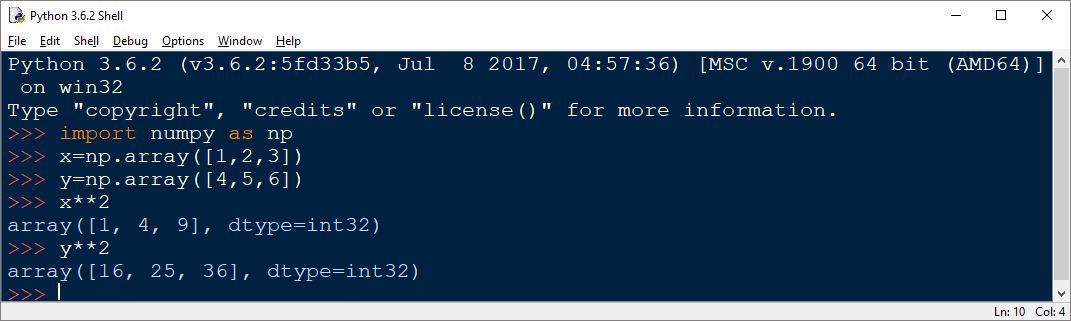
**Addition:** by using + operator we can perform addition of two list or two matrix.



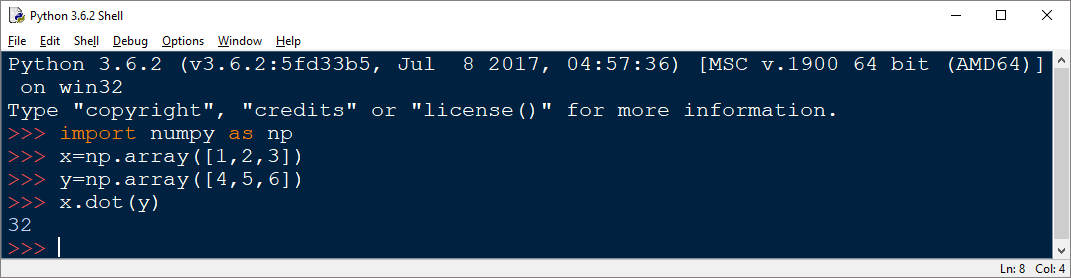
**Multiplication:**  by using \* operator we can perform multiplication of two list or two matrix.



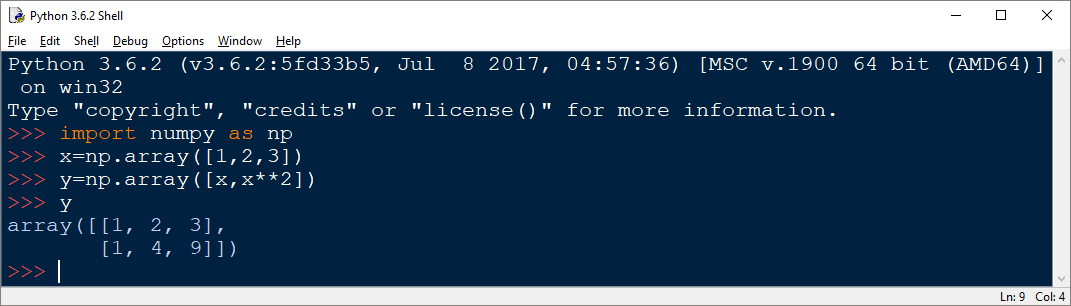
**Power:** by using \*\* operator we can perform power of two list or two matrix.



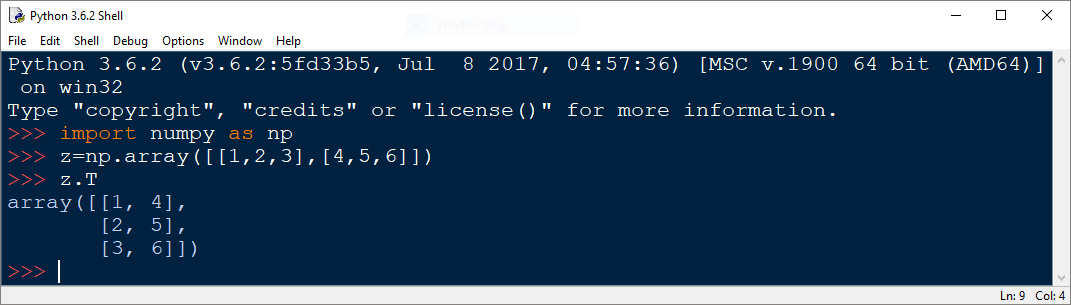
**Dot:** Return the dot product of two vectors.



Process: ( 1\*4)+(2\*5)+(3\*6)



**Transpose:** by using this we can transpose the matrix rows as columns or columns as rows



**Array datatype conversion:** Here we are discussing mainly two methods name as dtype() and astype().

by using dtype() we can find the list of values belongs to which type.

by using astype() we can perform the type conversions.

Examples:

| >>> x = np.array([1, 2, 2.5])

| >>> x

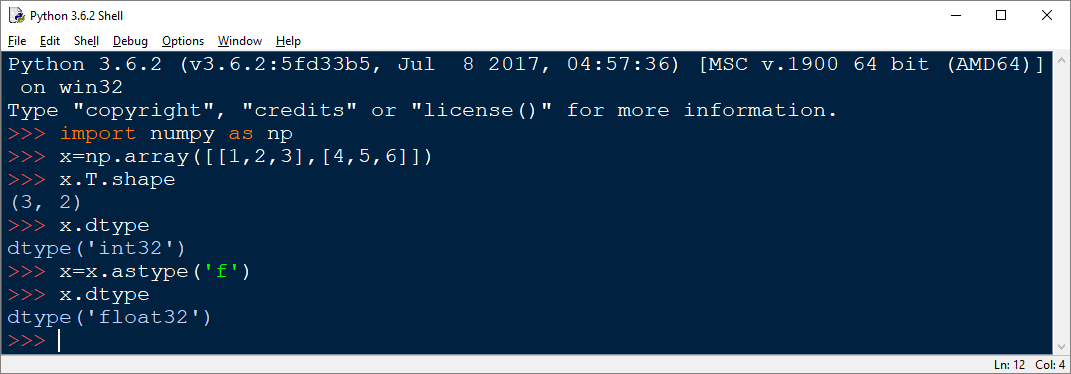
| array([ 1. , 2. , 2.5])

| >>>x.dtype()

| >>> x.astype(int)

| array([1, 2, 2])

>>>x.dtype()



**Math Functions:**

**Sum():** it returns sum of array.

**Min() :**it returns minimum value in array.

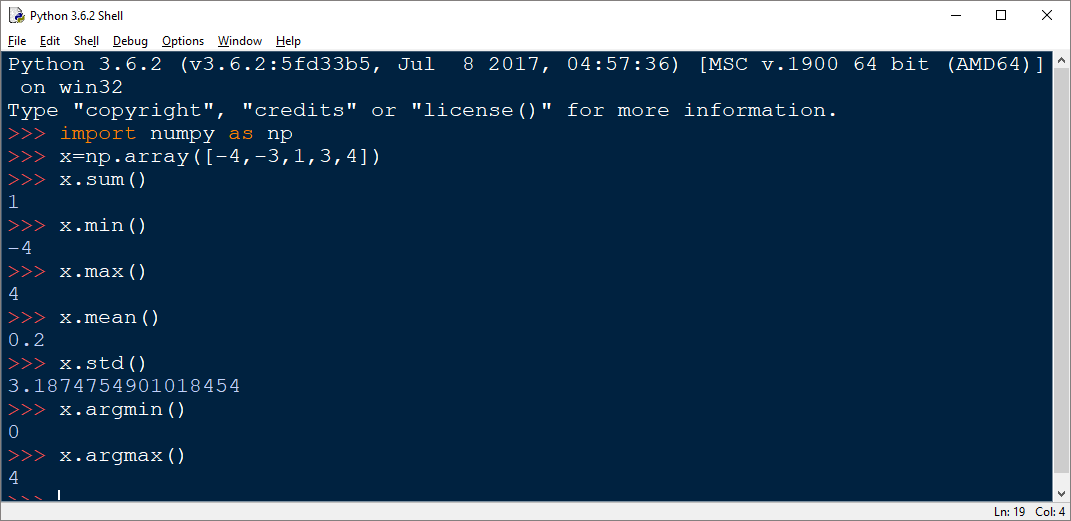
**Max():**it returns maximum value in array.

**Mean():**it returns mean value of array.

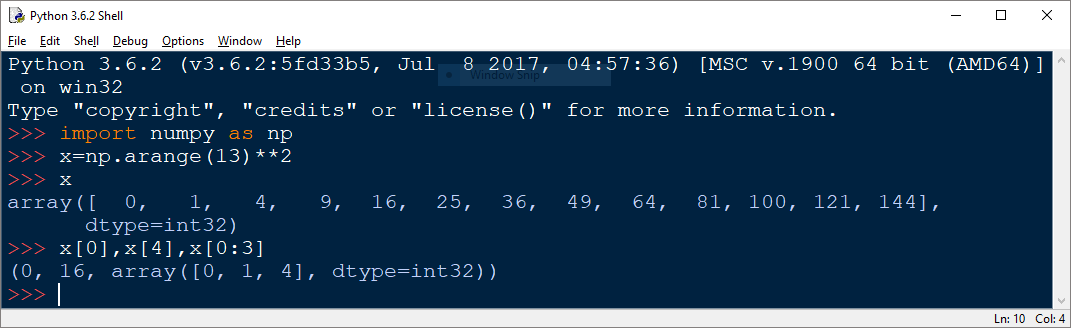
**Std():**it returns standard deviation value of array.

**Argmin():**it returns position of minimum value existing in array.

**Argmax():**it returns position of maximum value existing in array.



**Indexing and Slicing:**



**Iteraing over arrays:**

**>>> test=np.random.randint(0,10,(4,3))**

**>>> test**

**array([[2, 9, 1],**

**[2, 4, 8],**

**[5, 8, 8],**

**[9, 8, 2]])**

**>>> for row in test:**

**print(row)**

**Output:**

**[2 9 1]**

**[2 4 8]**

**[5 8 8]**

**[9 8 2]**

**>>> for row in range(len(test)):**

**print(test[row])**

**output:**

**[2 9 1]**

**[2 4 8]**

**[5 8 8]**

**[9 8 2]**

**>>> for row,i in enumerate(test):**

**print('row',row,'is',i)**

**Output:**

**row 0 is [2 9 1]**

**row 1 is [2 4 8]**

**row 2 is [5 8 8]**

**row 3 is [9 8 2]**

**>>> test=test\*\*2**

**>>> test**

**array([[ 4, 81, 1],**

**[ 4, 16, 64],**

**[25, 64, 64],**

**[81, 64, 4]], dtype=int32)**

**>>> test1=test\*\*2**

**>>> test1**

**array([[ 16, 6561, 1],**

**[ 16, 256, 4096],**

**[ 625, 4096, 4096],**

**[6561, 4096, 16]], dtype=int32)**

**>>> for i,j in zip(test,test1):**

**print(i,'+',j,'=',i+j)**

**Output:**

**[ 4 81 1] + [ 16 6561 1] = [ 20 6642 2]**

**[ 4 16 64] + [ 16 256 4096] = [ 20 272 4160]**

**[25 64 64] + [ 625 4096 4096] = [ 650 4160 4160]**

**[81 64 4] + [6561 4096 16] = [6642 4160 20]**

**If u want to know more methods on numpy use method ‘dir(numpy)’.**