In [2]:
```
#1

import numpy as np
np.random.seed(21) # This guarantees the code will generate the same
#set of random numbers whenever executed
random_integers = np.random.randint(1,high=500000, size=(20, 5))
random_integers
```

Out[2]:
```
array([[ 80842, 333008, 202553, 140037,  81969],
       [ 63857,  42105, 261540, 481981, 176739],
       [489984, 326386, 110795, 394863,  25024],
       [ 38317,  49982, 408830, 485118,  16119],
       [407675, 231729, 265455, 109413, 103399],
       [174677, 343356, 301717, 224120, 401101],
       [140473, 254634, 112262,  25063, 108262],
       [375059, 406983, 208947, 115641, 296685],
       [444899, 129585, 171318, 313094, 425041],
       [188411, 335140, 141681,  59641, 211420],
       [287650,   8973, 477425, 382803, 465168],
       [  3975,  32213, 160603, 275485, 388234],
       [246225,  56174, 244097,   9350, 496966],
       [225516, 273338,  73335, 283013, 212813],
       [ 38175, 282399, 318413, 337639, 379802],
       [198049, 101115, 419547, 260219, 325793],
       [148593, 425024, 348570, 117968, 107007],
       [ 52547, 180346, 178760, 305186, 262153],
       [ 11835, 449971, 494184, 472031, 353049],
       [476442,  35455, 191553, 384154,  29917]])
```

In [11]:
```
avg=np.mean(random_integers[:,1])
print(format(avg,".2f"))
avg2=np.mean(random_integers[:5,2:4])#mean fucntion is used to find the mean value
print(format(avg2,".2f"))
```

```
214895.80
286058.50
```

In [15]:
```
#b

#1
import numpy as np
first_matrix = np.array([[1, 2, 3], [4, 5, 6]])
print(first_matrix)

#2
second_matrix = np.array([1, 2, 3])
print(second_matrix)

#3
my_vector = np.array([1, 2, 3, 4, 5, 6])
selection = my_vector % 2 == 0
my_vector[selection]
```

```
[[1 2 3]
 [4 5 6]]
[1 2 3]
```

Out[15]:
```
array([2, 4, 6])
```

In [21]:
```
#c
import numpy as np
my_array = np.array([[1, 2, 3], [4, 5, 6]])
print(my_array)

#1
my_slice = my_array[:, 1:3]
print(my_slice)

#2
my_array=my_array*2
print(my_array)
my_slice = my_array[:, 1:3]
print(my_slice)
```

```
[[1 2 3]
 [4 5 6]]
[[2 3]
 [5 6]]
[[ 2  4  6]
 [ 8 10 12]]
[[ 4  6]
 [10 12]]
```

In [23]:
```
#d
import numpy as np
my_array = np.array([[1, 2, 3], [4, 5, 6]])
print(my_array)
my_slice = my_array[:, 1:3].copy() #while copy my_slice will create a new array in which chages made does not the effe
print(my_slice)
my_array[:, :] = my_array * 2
my_slice = my_array[:, 1:3].copy()
print(my_slice)
```

```
[[1 2 3]
 [4 5 6]]
[[2 3]
 [5 6]]
[[ 4  6]
 [10 12]]
```

In [24]:
```
#e
import numpy as np
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
sel=arr%2==0
arr[sel]
```

Out[24]:
```
array([0, 2, 4, 6, 8])
```

In [26]:
```
#f
import numpy as np
x = np.array([5,6,7,8,3,4])
y = np.array([5,3,4,5,2,4])
pos=np.where(x==y)
print(pos)
```

```
(array([0, 5], dtype=int64),)
```

In [27]:
```
#g

# standarisation to used in machine learning to plot the data it the data value is higher then it is difficult to plo
import numpy as np
k = np.array([5,3,4,5,2,4])
std=(k-np.min(k)/(np.max(k)-np.min(k)))
print(std)
```

```
[4.33333333 2.33333333 3.33333333 4.33333333 1.33333333 3.33333333]
```

In [29]:
```
#h

# percentile function is used to find the nth percentile of the array element
import numpy as np
p = np.array([15,10, 3,2,5,6,4])
print(np.percentile(p,50))
```

```
5.0
```

In [30]:
```
#i

import numpy as np
p = np.array([5,10, np.nan, 3, 2, 5, 6, np.nan])
pos=np.where(np.isnan(p))
pos
```

Out[30]:
```
(array([2, 7], dtype=int64),)
```

In [35]:
```
import numpy
#1

my_array = numpy.array([ [1, 2], [3, 4] ])
print(numpy.sum(my_array, axis = 0))
print(numpy.sum(my_array, axis = 1))
print(numpy.sum(my_array, axis = None))
print(numpy.sum(my_array))

#2

my_array = numpy.array([ [1, 2], [3, 4] ])
print (numpy.prod(my_array, axis = 0))
print (numpy.prod(my_array, axis = 1))
print (numpy.prod(my_array, axis = None))
print (numpy.prod(my_array))

#4

my_array = numpy.array([[2, 5],
 [3, 7],
 [1, 3],
 [4, 0]])
print (numpy.min(my_array, axis = 0)) #axis =0 it takes a individual row and column and perform the required operatio
print (numpy.min(my_array, axis = 1)) #axis=1 it take the particular row and column
print (numpy.min(my_array, axis = None))
print (numpy.min(my_array))

#5

my_array = numpy.array([[2, 5],
 [3, 7],
 [1, 3],
 [4, 0]])
print (numpy.max(my_array, axis = 0))
print (numpy.max(my_array, axis = 1))
print (numpy.max(my_array, axis = None))
print (numpy.max(my_array))
```

```
[4 6]
[3 7]
10
10
[3 8]
[ 2 12]
24
24
[1 0]
[2 3 1 0]
0
0
[4 7]
[5 7 3 4]
7
7
```

In [39]:
```
#6

import numpy
change_array = numpy.array([1,2,3,4,5,6])
change_array.shape = (3, 2)
print(change_array)

my_array = numpy.array([1,2,3,4,5,6])
print(numpy.reshape(my_array,(3,2)))
```

```
[[1 2]
 [3 4]
 [5 6]]
[[1 2]
 [3 4]
 [5 6]]
```

In [ ]: