In [1]:
```
my_arr = list(range(1000000))
```

In [6]:
```
%time for _ in range(10): my_arr2 = [x * 2 for x in my_arr]
```

```
Wall time: 1.07 s
```

In [7]:
```
import numpy as np
num_arr = np.arange(1000000)
```

In [8]:
```
%time for _ in range(10): num_arr2 = num_arr * 2
```

```
Wall time: 36 ms
```

In [9]:
```
zeroes_arr = np.zeros(10)
```

In [10]:
```
zeroes_arr
```

Out[10]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [13]:
```
two_d_arr = np.zeros((4,3))
```

In [14]:
```
two_d_arr
```

Out[14]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [15]:
```
two_d_arr.reshape((2,2,3))
```

Out[15]:
```
array([[[0., 0., 0.],
        [0., 0., 0.]],

       [[0., 0., 0.],
        [0., 0., 0.]]])
```

In [16]:
```
ones_arr = np.ones((3,4))
```

In [18]:
```
ones_arr = ones_arr * 5
```

In [19]:
```
ones_arr
```

Out[19]:
```
array([[5., 5., 5., 5.],
       [5., 5., 5., 5.],
       [5., 5., 5., 5.]])
```

In [22]:
```
arr = np.array([1,2,3,4,5,6,7,8])
```

In [24]:
```
arr_res = arr.reshape((2,4))
```

In [25]:
```
arr_res
```

Out[25]:
```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

In [26]:
```
arr_res[1,3]
```

Out[26]:
```
8
```

In [27]:
```
arr_res[0,2]
```

Out[27]:
```
3
```

In [29]:
```
arr_res[0:2,0:2]
```

Out[29]:
```
array([[1, 2],
       [5, 6]])
```

In [32]:
```
arr_2d = np.arange(0,101,5)
```

In [33]:
```
arr_2d
```

Out[33]:
```
array([  0,   5,  10,  15,  20,  25,  30,  35,  40,  45,  50,  55,  60,
        65,  70,  75,  80,  85,  90,  95, 100])
```

In [34]:
```
emp_arr = np.empty((3,3))
```

In [35]:
```
emp_arr
```

Out[35]:
```
array([[0.00000000e+000, 0.00000000e+000, 0.00000000e+000],
       [0.00000000e+000, 0.00000000e+000, 2.68771711e-321],
       [1.23075756e-312, 1.97345609e-312, 8.01089061e-307]])
```

In [37]:
```
emp_arr.shape
```

Out[37]:
```
(3, 3)
```

In [38]:
```
emp_arr.dtype
```

Out[38]:
```
dtype('float64')
```

In [39]:
```
emp_arr.ndim
```

Out[39]:
```
2
```

```
11111
```

```
10001
10001
10001
11111
```

In [47]:
```
arr = np.ones((10,10))
```

In [48]:
```
arr[1:-1,1:-1] = 0
```

In [49]:
```
arr
```

Out[49]:
```
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]])
```

```
111111
000000
111111
000000
111111
```

In [53]:
```
arr=np.ones((5,5))
arr[1:-1:2]=0
```

In [54]:
```
arr
```

Out[54]:
```
array([[1., 1., 1., 1., 1.],
       [0., 0., 0., 0., 0.],
       [1., 1., 1., 1., 1.],
       [0., 0., 0., 0., 0.],
       [1., 1., 1., 1., 1.]])
```

```
10101010
01010101
10101010
01010101
10101010
01010101
10101010
01010101
```

In [ ]:
```
array[rows,columns]
```

In [83]:
```
arr = np.zeros((8,8))
arr
```

Out[83]:
```
array([[0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0.]])
```

In [84]:
```
arr[::2,::2]=1 #for even
arr[1::2,1::2] = 1 #for odd
```

In [85]:
```
arr
```

Out[85]:
```
array([[1., 0., 1., 0., 1., 0., 1., 0.],
       [0., 1., 0., 1., 0., 1., 0., 1.],
       [1., 0., 1., 0., 1., 0., 1., 0.],
       [0., 1., 0., 1., 0., 1., 0., 1.],
       [1., 0., 1., 0., 1., 0., 1., 0.],
       [0., 1., 0., 1., 0., 1., 0., 1.],
       [1., 0., 1., 0., 1., 0., 1., 0.],
       [0., 1., 0., 1., 0., 1., 0., 1.]])
```

```
10001
01010
00100
01010
10001
```

In [64]:
```
arr.dtype
```

Out[64]:
```
dtype('float64')
```

In [68]:
```
arr.astype("int8")
```

Out[68]:
```
array([[1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1]], dtype=int8)
```

In [70]:
```
arr1 = np.array([1,2,3,4])
arr2 = np.array([5,6,7,8])
add_arr = arr1 + arr2
```

In [71]:
```
add_arr
```

Out[71]:
```
array([ 6,  8, 10, 12])
```

In [74]:
```
dir(np.amin)
```

Out[74]:
```
['__annotations__',
 '__call__',
 '__class__',
 '__closure__',
 '__code__',
 '__defaults__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__get__',
 '__getattribute__',
 '__globals__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__kwdefaults__',
 '__le__',
 '__lt__',
 '__module__',
 '__name__',
 '__ne__',
 '__new__',
 '__qualname__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__wrapped__',
 '_implementation']
```

In [75]:
```
arr1
```

Out[75]:
```
array([1, 2, 3, 4])
```

In [77]:
```
arr1>3
```

Out[77]:
```
array([False, False, False,  True])
```

In [78]:
```
arr1[arr1>3]
```

Out[78]:
```
array([4])
```

In [79]:
```
arr1.astype("float64")
```

Out[79]:
```
array([1., 2., 3., 4.])
```

In [87]:
```
arr1[[0,2]]
```

Out[87]:
```
array([1, 3])
```

In [88]:
```
np.sin(arr1)
```

Out[88]:
```
array([ 0.84147098,  0.90929743,  0.14112001, -0.7568025 ])
```

In [89]:
```
np.max(arr1)
```

Out[89]:
```
4
```

In [90]:
```
np.log(arr1)
```

Out[90]:
```
array([0.        , 0.69314718, 1.09861229, 1.38629436])
```

In [91]:
```
np.argmax(arr1)
```

Out[91]:
```
3
```

In [92]:
```
arr1
```

Out[92]:
```
array([1, 2, 3, 4])
```

In [93]:
```
marks = np.array([30, 80 , 60 ,90 , 10])
```

In [94]:
```
result = np.where(marks>33 , "Pass" , "Fail")
```

In [95]:
```
result
```

Out[95]:
```
array(['Fail', 'Pass', 'Pass', 'Pass', 'Fail'], dtype='<U4')
```

In [101]:
```
array1 = np.arange(1,11).reshape((2,5))
array1
```

Out[101]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10]])
```

In [102]:
```
array1.T
```

Out[102]:
```
array([[ 1,  6],
       [ 2,  7],
       [ 3,  8],
       [ 4,  9],
       [ 5, 10]])
```

In [103]:
```
array1.transpose()
```

Out[103]:
```
array([[ 1,  6],
       [ 2,  7],
       [ 3,  8],
       [ 4,  9],
       [ 5, 10]])
```

In [104]:
```
array1.mean()
```

Out[104]:
```
5.5
```

In [107]:
```
array1.mean(axis = 0)
```

Out[107]:
```
array([3.5, 4.5, 5.5, 6.5, 7.5])
```

In [106]:
```
array1
```

Out[106]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10]])
```

In [108]:
```
array1.mean(axis=1)
```

Out[108]:
```
array([3., 8.])
```

In [110]:
```
arr1.cumsum()
```

Out[110]:
```
array([ 1,  3,  6, 10], dtype=int32)
```

In [111]:
```
arr1
```

Out[111]:
```
array([1, 2, 3, 4])
```

In [112]:
```
arr1.cumprod()
```

Out[112]:
```
array([ 1,  2,  6, 24], dtype=int32)
```

In [113]:
```
marks
```

Out[113]:
```
array([30, 80, 60, 90, 10])
```

In [114]:
```
marks>33
```

Out[114]:
```
array([False,  True,  True,  True, False])
```

In [115]:
```
(marks>33).sum()
```

Out[115]:
```
3
```

In [116]:
```
(marks>33).any()
```

Out[116]:
```
True
```

In [117]:
```
(marks>33).all()
```

Out[117]:
```
False
```

In [134]:
```
arr1 = np.array([5,7,3,1])
np.sort(arr1)
```

Out[134]:
```
array([1, 3, 5, 7])
```

In [130]:
```
arr1
```

Out[130]:
```
array([1, 2, 3, 4])
```

In [135]:
```
arr2 = np.array([5,7,5,3,7,1])
np.unique(arr2)
```

Out[135]:
```
array([1, 3, 5, 7])
```

In [136]:
```
d2_arr = np.array([[1,5],[3,7]])
```

In [137]:
```
d2_arr
```

Out[137]:
```
array([[1, 5],
       [3, 7]])
```

In [138]:
```
d2_arr.trace()
```

Out[138]:
```
8
```

In [147]:
```
from numpy import linalg
linalg.eig(d2_arr)
```

Out[147]:
```
(array([-0.89897949,  8.89897949]), array([[-0.93484692, -0.53484692],
        [ 0.35505103, -0.84494897]]))
```

In [151]:
```
np.random.normal((2,2))
```

Out[151]:
```
array([1.36780622, 2.92820814])
```

In [152]:
```
arr1 = np.array([[1,2],[3,4]])
arr2 = np.array([[5,6],[7,8]])
```

In [154]:
```
np.concatenate([arr1,arr2] , axis=0)
```

Out[154]:
```
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
```

In [155]:
```
arr1
```

Out[155]:
```
array([[1, 2],
       [3, 4]])
```

In [156]:
```
arr2
```

Out[156]:
```
array([[5, 6],
       [7, 8]])
```

In [157]:
```
np.concatenate([arr1,arr2] , axis=1)
```

Out[157]:
```
array([[1, 2, 5, 6],
       [3, 4, 7, 8]])
```

In [158]:
```
np.hstack([arr1 , arr2])
```

Out[158]:
```
array([[1, 2, 5, 6],
       [3, 4, 7, 8]])
```

In [159]:
```
np.vstack([arr1 , arr2])
```

Out[159]:
```
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
```

In [160]:
```
np.r_[arr1,arr2]
```

Out[160]:
```
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
```

In [161]:
```
np.c_[arr1,arr2]
```

Out[161]:
```
array([[1, 2, 5, 6],
       [3, 4, 7, 8]])
```

In [ ]:

In [1]:
```
import numpy as np
```

In [4]:
```
#Vectorization
```

In [2]:
```
arr = np.zeros((2,10))
```

In [6]:
```
arr+10 #10 is scalar
```

Out[6]:
```
array([[10., 10., 10., 10., 10., 10., 10., 10., 10., 10.],
       [10., 10., 10., 10., 10., 10., 10., 10., 10., 10.]])
```

In [10]:
```
arr
```

Out[10]:
```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

In [5]:
```
arr2 = np.ones((2,10))
```

In [11]:
```
arr2
```

Out[11]:
```
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]])
```

In [7]:
```
arr+arr2 #Both operands are arrays
```

Out[7]:
```
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]])
```

In [8]:
```
l1 = [1,2,4,5]
l2 = [3,4,6,7]
```

In [13]:
```
#Adding two list
sum = []
for i,_ in enumerate(l1): #0,1 1,2 2,4 3,5
    sum.append(l1[i] + l2[i])
print(sum)
```

```
[4, 6, 10, 12]
```

In [14]:
```
#where
```

In [15]:
```
marks = np.array([75,39,50, 80,90])
```

In [18]:
```
x = np.where(marks>=50,"Pass", "Fail")
```

In [19]:
```
x
```

Out[19]:
```
array(['Pass', 'Fail', 'Pass', 'Pass', 'Pass'], dtype='<U4')
```

In [20]:
```
salary = np.array([10,70, 80, 50, 30])
```

In [22]:
```
incremented_salary = np.where(salary<50, salary*1.1, salary*1.05)
```

In [24]:
```
incremented_salary.dtype
```

Out[24]:
```
dtype('float64')
```

In [26]:
```
arr = np.arange(1,101).reshape((10,10))
```

In [27]:
```
arr
```

Out[27]:
```
array([[  1,   2,   3,   4,   5,   6,   7,   8,   9,  10],
       [ 11,  12,  13,  14,  15,  16,  17,  18,  19,  20],
       [ 21,  22,  23,  24,  25,  26,  27,  28,  29,  30],
       [ 31,  32,  33,  34,  35,  36,  37,  38,  39,  40],
       [ 41,  42,  43,  44,  45,  46,  47,  48,  49,  50],
       [ 51,  52,  53,  54,  55,  56,  57,  58,  59,  60],
       [ 61,  62,  63,  64,  65,  66,  67,  68,  69,  70],
       [ 71,  72,  73,  74,  75,  76,  77,  78,  79,  80],
       [ 81,  82,  83,  84,  85,  86,  87,  88,  89,  90],
       [ 91,  92,  93,  94,  95,  96,  97,  98,  99, 100]])
```

In [28]:
```
arr.T
```

Out[28]:
```
array([[  1,  11,  21,  31,  41,  51,  61,  71,  81,  91],
       [  2,  12,  22,  32,  42,  52,  62,  72,  82,  92],
       [  3,  13,  23,  33,  43,  53,  63,  73,  83,  93],
       [  4,  14,  24,  34,  44,  54,  64,  74,  84,  94],
       [  5,  15,  25,  35,  45,  55,  65,  75,  85,  95],
       [  6,  16,  26,  36,  46,  56,  66,  76,  86,  96],
       [  7,  17,  27,  37,  47,  57,  67,  77,  87,  97],
       [  8,  18,  28,  38,  48,  58,  68,  78,  88,  98],
       [  9,  19,  29,  39,  49,  59,  69,  79,  89,  99],
       [ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100]])
```

In [29]:
```
arr
```

Out[29]:
```
array([[  1,   2,   3,   4,   5,   6,   7,   8,   9,  10],
       [ 11,  12,  13,  14,  15,  16,  17,  18,  19,  20],
       [ 21,  22,  23,  24,  25,  26,  27,  28,  29,  30],
       [ 31,  32,  33,  34,  35,  36,  37,  38,  39,  40],
       [ 41,  42,  43,  44,  45,  46,  47,  48,  49,  50],
       [ 51,  52,  53,  54,  55,  56,  57,  58,  59,  60],
       [ 61,  62,  63,  64,  65,  66,  67,  68,  69,  70],
       [ 71,  72,  73,  74,  75,  76,  77,  78,  79,  80],
       [ 81,  82,  83,  84,  85,  86,  87,  88,  89,  90],
       [ 91,  92,  93,  94,  95,  96,  97,  98,  99, 100]])
```

In [30]:
```
arr.transpose()
```

Out[30]:
```
array([[  1,  11,  21,  31,  41,  51,  61,  71,  81,  91],
       [  2,  12,  22,  32,  42,  52,  62,  72,  82,  92],
       [  3,  13,  23,  33,  43,  53,  63,  73,  83,  93],
       [  4,  14,  24,  34,  44,  54,  64,  74,  84,  94],
       [  5,  15,  25,  35,  45,  55,  65,  75,  85,  95],
       [  6,  16,  26,  36,  46,  56,  66,  76,  86,  96],
       [  7,  17,  27,  37,  47,  57,  67,  77,  87,  97],
       [  8,  18,  28,  38,  48,  58,  68,  78,  88,  98],
       [  9,  19,  29,  39,  49,  59,  69,  79,  89,  99],
       [ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100]])
```

In [31]:
```
arr = np.array([1,3,4,5,2])
```

In [32]:
```
arr.mean()
```

Out[32]:
```
3.0
```

In [33]:
```
arr.cumsum()
```

Out[33]:
```
array([ 1,  4,  8, 13, 15], dtype=int32)
```

In [34]:
```
arr.cumprod()
```

Out[34]:
```
array([  1,   3,  12,  60, 120], dtype=int32)
```

In [ ]:
```
#Procedure Vs. Function
#Procedure doesn't return any value
#Function returns a value
```

In [35]:
```
#Methods for boolean arrays
```

In [36]:
```
gender = np.array(['Male', 'Female', 'Male', 'Male', 'Male'])
```

In [41]:
```
arr = gender == 'Male'
arr
```

Out[41]:
```
array([ True, False,  True,  True,  True])
```

In [40]:
```
arr.sum()
```

Out[40]:
```
4
```

In [42]:
```
arr = gender == 'Female'
arr.sum()
```

Out[42]:
```
1
```

In [43]:
```
arr
```

Out[43]:
```
array([False,  True, False, False, False])
```

In [44]:
```
arr.any()
```

Out[44]:
```
True
```

In [45]:
```
arr.all()
```

Out[45]:
```
False
```

In [46]:
```
arr = np.array([2,4,10,20,16])
```

In [50]:
```
x = (arr%2 == 0)
```

In [48]:
```
x
```

Out[48]:
```
array([ True,  True,  True,  True,  True])
```

In [49]:
```
x.all()
```

Out[49]:
```
True
```

In [ ]:
```
#Universal quantifier
#Existential quantifier
```

In [51]:
```
arr
```

Out[51]:
```
array([ 2,  4, 10, 20, 16])
```

In [52]:
```
arr.sort()
```

In [53]:
```
arr
```

Out[53]:
```
array([ 2,  4, 10, 16, 20])
```

In [55]:
```
arr.sort() #Inplace
```

In [56]:
```
arr[::-1]
```

Out[56]:
```
array([20, 16, 10,  4,  2])
```

In [57]:
```
arr = np.array([1,2,4,1,6,2])
```

In [59]:
```
u = np.unique(arr)
```

In [61]:
```
#Save array
```

In [60]:
```
np.save('data',u)
```

In [62]:
```
#Load array
```

In [64]:
```
content = np.load('data.npy')
```

In [65]:
```
content
```

Out[65]:
```
array([1, 2, 4, 6])
```

In [66]:
```
#Saving multiple arrays
```

In [67]:
```
students = np.array(["Ali","Asad","Ahmed"])
marks = np.array([30,20,60])
```

In [68]:
```
np.savez('arrays',std=students,m=marks)
```

In [74]:
```
results = np.load('arrays.npz')
```

In [75]:
```
results['std']
```

Out[75]:
```
array(['Ali', 'Asad', 'Ahmed'], dtype='<U5')
```

In [77]:
```
results['m']
```

Out[77]:
```
array([30, 20, 60])
```

In [78]:
```
arr=np.arange(16).reshape((4,4))
```

In [79]:
```
arr
```

Out[79]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [81]:
```
arr.diagonal()
```

Out[81]:
```
array([ 0,  5, 10, 15])
```

In [82]:
```
arr.trace()
```

Out[82]:
```
30
```

In [83]:
```
arr1=np.arange(10,26).reshape((4,4))
```

In [84]:
```
arr.dot(arr1)
```

Out[84]:
```
array([[ 116,  122,  128,  134],
       [ 372,  394,  416,  438],
       [ 628,  666,  704,  742],
       [ 884,  938,  992, 1046]])
```

In [88]:
```
from numpy import linalg
```

In [89]:
```
linalg.det(arr)
```

Out[89]:
```
0.0
```

In [91]:
```
linalg.eig(arr)
```

Out[91]:
```
(array([ 3.24642492e+01, -2.46424920e+00,  2.14966418e-15, -1.17200157e-16]),
 array([[-0.11417645,  0.7327781 , -0.40377562,  0.05533605],
        [-0.3300046 ,  0.28974835,  0.81421492,  0.33237807],
        [-0.54583275, -0.15328139, -0.41710299, -0.83076428],
        [-0.76166089, -0.59631113,  0.00666369,  0.44305017]]))
```

In [94]:
```
linalg.inv(np.array([[1,5],[9,8]]))
```

Out[94]:
```
array([[-0.21621622,  0.13513514],
       [ 0.24324324, -0.02702703]])
```

In [95]:
```
linalg.qr(arr)
```

Out[95]:
```
(array([[ 0.        , -0.83666003,  0.54061686,  0.08793982],
        [-0.26726124, -0.47809144, -0.78636895,  0.28569892],
        [-0.53452248, -0.11952286, -0.04911268, -0.8352173 ],
        [-0.80178373,  0.23904572,  0.29486477,  0.46157856]]),
 array([[-1.49666295e+01, -1.65701970e+01, -1.81737645e+01,
         -1.97773319e+01],
        [ 0.00000000e+00, -1.19522861e+00, -2.39045722e+00,
         -3.58568583e+00],
        [ 0.00000000e+00,  0.00000000e+00, -3.82019983e-15,
         -4.31063089e-15],
        [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
          7.74364831e-17]]))
```

In [96]:
```
linalg.svd(arr)
```

Out[96]:
```
(array([[-0.09184212, -0.83160389,  0.53515573,  0.11665482],
        [-0.31812733, -0.44586433, -0.80049034,  0.24334177],
        [-0.54441254, -0.06012478, -0.00448651, -0.836648  ],
        [-0.77069775,  0.32561478,  0.26982112,  0.47665141]]),
 array([3.51399637e+01, 2.27661021e+00, 1.69970911e-15, 8.07447880e-17]),
 array([[-0.42334086, -0.47243254, -0.52152422, -0.57061589],
        [ 0.72165263,  0.27714165, -0.16736932, -0.6118803 ],
        [-0.22125808,  0.66846675, -0.67315927,  0.2259506 ],
        [ 0.50104377, -0.50314233, -0.49684666,  0.49894522]]))
```

In [97]:
```
np.random.normal(3,4,(3,3))
```

Out[97]:
```
array([[ 0.43606583,  4.27203603, -1.85127604],
       [-1.46327196,  1.86373419,  9.48334897],
       [ 3.11570102,  1.04362901,  6.9876883 ]])
```

In [99]:
```
np.random.gamma(1,2,(4,4))
```

Out[99]:
```
array([[3.06371388, 1.86591286, 2.58023061, 0.89127798],
       [2.84303265, 5.33219431, 4.33796096, 1.42668399],
       [4.55751492, 0.98769555, 0.61390022, 1.73705547],
       [0.43125193, 5.56393488, 0.24515717, 0.50041773]])
```

In [106]:
```
np.random.seed(5)
```

In [107]:
```
np.random.rand()
```

Out[107]:
```
0.22199317108973948
```

In [108]:
```
arr
```

Out[108]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [110]:
```
arr.reshape((2,2,4))
```

Out[110]:
```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7]],

       [[ 8,  9, 10, 11],
        [12, 13, 14, 15]]])
```

In [119]:
```
arr = np.arange(16)
```

In [120]:
```
arr
```

Out[120]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [121]:
```
arr.reshape((4,4),order='F') #column major order
```

Out[121]:
```
array([[ 0,  4,  8, 12],
       [ 1,  5,  9, 13],
       [ 2,  6, 10, 14],
       [ 3,  7, 11, 15]])
```

In [122]:
```
arr.reshape((4,4),order='C') #row major order
```

Out[122]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [123]:
```
arr = np.array([[1,8,7],[3,4,5],[9,0,1]])
```

In [124]:
```
arr
```

Out[124]:
```
array([[1, 8, 7],
       [3, 4, 5],
       [9, 0, 1]])
```

In [125]:
```
arr.flatten()
```

Out[125]:
```
array([1, 8, 7, 3, 4, 5, 9, 0, 1])
```

In [126]:
```
arr.ravel()
```

Out[126]:
```
array([1, 8, 7, 3, 4, 5, 9, 0, 1])
```

In [130]:
```
arr.reshape(9)
```

Out[130]:
```
array([1, 8, 7, 3, 4, 5, 9, 0, 1])
```

In [131]:
```
#Assignment: Difference between ravel and flatten
```

In [132]:
```
#Concatenate
```

In [133]:
```
a = np.array([[1,2],[3,4]])
b = np.array([[7,8],[9,10]])
```

In [136]:
```
print('a=',a)
print('b=',b)
```

```
a= [[1 2]
 [3 4]]
b= [[ 7  8]
 [ 9 10]]
```

In [137]:
```
np.concatenate((a,b),axis=0)
```

Out[137]:
```
array([[ 1,  2],
       [ 3,  4],
       [ 7,  8],
       [ 9, 10]])
```

In [138]:
```
np.concatenate((a,b),axis=1)
```

Out[138]:
```
array([[ 1,  2,  7,  8],
       [ 3,  4,  9, 10]])
```

In [139]:
```
a = np.array([[1,7,9],[2,3,4],[5,6,7]])
b = np.array([[9,4,9],[2,3,2],[0,6,7]])
```

In [141]:
```
np.concatenate((a,b))
```

Out[141]:
```
array([[1, 7, 9],
       [2, 3, 4],
       [5, 6, 7],
       [9, 4, 9],
       [2, 3, 2],
       [0, 6, 7]])
```

In [143]:
```
np.hstack((a,b)) #concatenate axis = 1
```

Out[143]:
```
array([[1, 7, 9, 9, 4, 9],
       [2, 3, 4, 2, 3, 2],
       [5, 6, 7, 0, 6, 7]])
```

In [144]:
```
np.vstack((a,b)) #concatenate axis = 0
```

Out[144]:
```
array([[1, 7, 9],
       [2, 3, 4],
       [5, 6, 7],
       [9, 4, 9],
       [2, 3, 2],
       [0, 6, 7]])
```

In [147]:
```
np.r_[a,b] #concatenate axis = 0
```

Out[147]:
```
array([[1, 7, 9],
       [2, 3, 4],
       [5, 6, 7],
       [9, 4, 9],
       [2, 3, 2],
       [0, 6, 7]])
```

In [148]:
```
np.c_[a,b] #concatenate axis = 1
```

Out[148]:
```
array([[1, 7, 9, 9, 4, 9],
       [2, 3, 4, 2, 3, 2],
       [5, 6, 7, 0, 6, 7]])
```

In [149]:
```
x = np.array([1,9,9,7,1,2,3])
```

In [153]:
```
np.split(x,[3, 5])
```

Out[153]:
```
[array([1, 9, 9]), array([7, 1]), array([2, 3])]
```

In [164]:
```
l = np.arange(2,7,2)
```

In [165]:
```
l
```

Out[165]:
```
array([2, 4, 6])
```

In [166]:
```
x
```

Out[166]:
```
array([1, 9, 9, 7, 1, 2, 3])
```

In [167]:
```
np.split(x,l)
```

Out[167]:
```
[array([1, 9]), array([9, 7]), array([1, 2]), array([3])]
```

In [168]:
```
x = np.array([[1,3],[2,4]])
```

In [170]:
```
x
```

Out[170]:
```
array([[1, 3],
       [2, 4]])
```

In [169]:
```
np.split(x,[1])
```

Out[169]:
```
[array([[1, 3]]), array([[2, 4]])]
```

In [ ]:
```

```

In [1]:
```
import pandas as pd
```

```
D:\Anaconda3\lib\site-packages\pandas\compat\_optional.py:138: UserWarning: P
andas requires version '2.7.0' or newer of 'numexpr' (version '2.6.9' current
ly installed).
  warnings.warn(msg, UserWarning)
```

In [143]:
```
#Two types of datastructures in pandas
# Series - A series of data (1 dimensional)
# DataFrames - Tabular data (2 dimensional)
```

In [144]:
```
marks = pd.Series([80,70,60]) #Creating series through a list
```

In [5]:
```
marks
```

Out[5]:
```
0    80
1    70
2    60
dtype: int64
```

In [145]:
```
marks[1] #Access element through indexes
```

Out[145]:
```
70
```

In [146]:
```
#specifying custom indexes
```

In [8]:
```
marks = pd.Series([80,70,60], index=['ali','sara','ahmed'])
```

In [9]:
```
marks
```

Out[9]:
```
ali      80
sara     70
ahmed    60
dtype: int64
```

In [10]:
```
marks['ali']
```

Out[10]:
```
80
```

In [11]:
```
#Specifying your own datatype
age = pd.Series([80,70,60], index=['ali','sara','ahmed'],dtype='float')
```

In [12]:
```
age
```

Out[12]:
```
ali      80.0
sara     70.0
ahmed    60.0
dtype: float64
```

In [13]:
```
age[age>70] #filtering data in a series
```

Out[13]:
```
ali    80.0
dtype: float64
```

In [14]:
```
age[['sara','ali']] #fancy indexing
```

Out[14]:
```
sara    70.0
ali     80.0
dtype: float64
```

In [16]:
```
age['sara'] = 18 #changing a value in a series
```

In [17]:
```
#Creating series from dictionary
```

In [20]:
```
age = pd.Series({'ali':80,'sara':70,'ahmed':None}) #Key will be assumed as ind
```

In [21]:
```
age
```

Out[21]:
```
ali      80.0
sara     70.0
ahmed     NaN
dtype: float64
```

In [22]:
```
pd.isnull(age) #Checking for null values. There is a not null function as well
```

Out[22]:
```
ali      False
sara     False
ahmed     True
dtype: bool
```

In [147]:
```
pd.notnull(age)
```

Out[147]:
```
ali       True
sara      True
ahmed    False
dtype: bool
```

In [24]:
```
age.mean()
```

Out[24]:
```
75.0
```

In [31]:
```
age1=pd.Series([100,50],index=['ali','ahmed'])
```

In [35]:
```
display(age1)
display(age)
```

```
ali      100
ahmed     50
dtype: int64
```

```
ali      80.0
sara     70.0
ahmed     NaN
dtype: float64
```

In [34]:
```
age+age1 #Adding two series data
```

Out[34]:
```
ahmed      NaN
ali      180.0
sara       NaN
dtype: float64
```

In [42]:
```
#Dataframes: Tabular data
#pd.DataFrame - Create a dataframe
```

In [40]:
```
stds = pd.DataFrame({
    'names':['Sarah','Ahmed','Ali'],
    'age':[50,60,70],
    'cgpa':[3.4,2.9,3.6],
    'address':['Karachi','Lahore','Islamabad']
})
#Keys will be columns
```

In [41]:
```
stds
```

Out[41]:

	names	age	cgpa	address
0	Sarah	50	3.4	Karachi
1	Ahmed	60	2.9	Lahore
2	Ali	70	3.6	Islamabad

In [43]:
```
stds.columns #Displaying the columns of dataframe
```

Out[43]:
```
Index(['names', 'age', 'cgpa', 'address'], dtype='object')
```

In [44]:
```
stds['names']
```

Out[44]:
```
0    Sarah
1    Ahmed
2      Ali
Name: names, dtype: object
```

In [45]:
```
stds[['names','age','cgpa']] #fancy indexing
```

Out[45]:

	names	age	cgpa
0	Sarah	50	3.4
1	Ahmed	60	2.9
2	Ali	70	3.6

In [47]:
```
stds[stds['age']>50] #Students with age > 50, filtering
```

Out[47]:

	names	age	cgpa	address
1	Ahmed	60	2.9	Lahore
2	Ali	70	3.6	Islamabad

In [50]:
```
stds.age
```

Out[50]:
```
0    50
1    60
2    70
Name: age, dtype: int64
```

In [52]:
```
import numpy as np
stds['semester']=np.arange(1,4) #using numpy function to specify values
```

In [54]:
```
stds[['semester','names']]
```

Out[54]:

	semester	names
0	1	Sarah
1	2	Ahmed
2	3	Ali

In [57]:
```
doctors = pd.DataFrame({
    'names':['Asad','Rahim'],
    'qualification':['PhD','MS']
},index=['d1','d2'])
```

In [58]:
```
doctors
```

Out[58]:

	names	qualification
d1	Asad	PhD
d2	Rahim	MS

In [59]:
```
doctors['address']='Karachi' #Adding a new column
```

In [60]:
```
doctors
```

Out[60]:

	names	qualification	address
d1	Asad	PhD	Karachi
d2	Rahim	MS	Karachi

In [62]:
```
doctors.loc['d1'] #Accessing row-wise data using index
#loc and iloc
```

Out[62]:
```
names               Asad
qualification        PhD
address          Karachi
Name: d1, dtype: object
```

In [63]:
```
doctors.iloc[0] #Accessing row-wise data using index location
```

Out[63]:
```
names               Asad
qualification        PhD
address          Karachi
Name: d1, dtype: object
```

In [64]:
```
doctors.iloc[1]
```

Out[64]:
```
names              Rahim
qualification         MS
address          Karachi
Name: d2, dtype: object
```

In [65]:
```
#iloc - position, row number
#loc - to access through index
```

In [69]:
```
doctors
```

Out[69]:

	names	qualification	address
d1	Asad	PhD	Karachi
d2	Rahim	MS	Karachi

In [70]:
```
del doctors['address'] #Removing a column
```

In [71]:
```
doctors
```

Out[71]:

	names	qualification
d1	Asad	PhD
d2	Rahim	MS

In [72]:
```
del doctors['names']
```

In [73]:
```
doctors
```

Out[73]:

	qualification
d1	PhD
d2	MS

In [74]:
```
doctors['names']='Test'
```

In [75]:
```
doctors
```

Out[75]:

	qualification	names
d1	PhD	Test
d2	MS	Test

In [77]:
```
stds
```

Out[77]:

	names	age	cgpa	address	semester
0	Sarah	50	3.4	Karachi	1
1	Ahmed	60	2.9	Lahore	2
2	Ali	70	3.6	Islamabad	3

In [78]:
```
stds.T
```

Out[78]:

	0	1	2
names	Sarah	Ahmed	Ali
age	50	60	70
cgpa	3.4	2.9	3.6
address	Karachi	Lahore	Islamabad
semester	1	2	3

In [81]:
```
stds.iloc[::2] #slicing in a dataframe
```

Out[81]:

	names	age	cgpa	address	semester
0	Sarah	50	3.4	Karachi	1
2	Ali	70	3.6	Islamabad	3

In [82]:
```
#Negative indexing
# -1: last value, -2: 2nd last value
```

In [83]:
```
stds.iloc[-1]
```

Out[83]:
```
names             Ali
age                70
cgpa              3.6
address     Islamabad
semester            3
Name: 2, dtype: object
```

In [84]:
```
stds.values #Getting numy array from frame
```

Out[84]:
```
array([['Sarah', 50, 3.4, 'Karachi', 1],
       ['Ahmed', 60, 2.9, 'Lahore', 2],
       ['Ali', 70, 3.6, 'Islamabad', 3]], dtype=object)
```

In [86]:
```
stds.drop([0],axis=0) #removing specific rows from dataframe
```

Out[86]:

	names	age	cgpa	address	semester
1	Ahmed	60	2.9	Lahore	2
2	Ali	70	3.6	Islamabad	3

In [87]:
```
#Creating dataframe with different data structures
```

In [88]:
```
arr = np.array([['Sarah', 50, 3.4, 'Karachi', 1],
       ['Ahmed', 60, 2.9, 'Lahore', 2],
       ['Ali', 70, 3.6, 'Islamabad', 3],
        ['Shoaib',37,2.9,'Pindi', 4]
               ])
```

In [92]:
```
df = pd.DataFrame(arr,columns=['Name','Age','CGPA', 'City','Semester'])
```

In [95]:
```
df
```

Out[95]:

	Name	Age	CGPA	City	Semester
0	Sarah	50	3.4	Karachi	1
1	Ahmed	60	2.9	Lahore	2
2	Ali	70	3.6	Islamabad	3
3	Shoaib	37	2.9	Pindi	4

In [101]:
```
lst = [['Sarah', 50, 3.4, 'Karachi', 1],
       ['Ahmed', 60, 2.9, 'Lahore', 2],
       ['Ali', 70, 3.6, 'Islamabad', 3],
        ['Shoaib',37,2.9,'Pindi', 4]
               ]

frame = pd.DataFrame(lst,columns=['A','B','C','D','E'])
```

In [99]:
```
#Try creating dataframe from different types of datastructures Slide-16
```

In [100]:
```
#Drop rows/columns
```

In [102]:
```
frame
```

Out[102]:

	A	B	C	D	E
0	Sarah	50	3.4	Karachi	1
1	Ahmed	60	2.9	Lahore	2
2	Ali	70	3.6	Islamabad	3
3	Shoaib	37	2.9	Pindi	4

In [104]:
```
frame.drop(['A','B'],axis=1) #drop column-wise
```

Out[104]:

	C	D	E
0	3.4	Karachi	1
1	2.9	Lahore	2
2	3.6	Islamabad	3
3	2.9	Pindi	4

In [106]:
```
frame.drop([0,1],axis=0) #row-wise
```

Out[106]:

	A	B	C	D	E
2	Ali	70	3.6	Islamabad	3
3	Shoaib	37	2.9	Pindi	4

In [107]:
```
#Slicing in dataframe
```

In [111]:
```
frame.iloc[::2] #start,end,step
```

Out[111]:

	A	B	C	D	E
0	Sarah	50	3.4	Karachi	1
2	Ali	70	3.6	Islamabad	3

In [116]:
```
frame.loc[:,'A':'C']
```

Out[116]:

	A	B	C
0	Sarah	50	3.4
1	Ahmed	60	2.9
2	Ali	70	3.6
3	Shoaib	37	2.9

In [117]:
```
#Apply function
```

In [119]:
```
frame.columns = ['Name','Age','CGPA','Address','Semester']
```

In [123]:
```
def old_young(age):
    if(age > 40):
        return 'Old'
    else:
        return 'Young'
```

In [127]:
```
frame['Status']=frame['Age'].apply(old_young)
```

In [132]:
```
frame
```

Out[132]:

	Name	Age	CGPA	Address	Semester	Status
0	Sarah	50	3.4	Karachi	1	Old
1	Ahmed	60	2.9	Lahore	2	Old
2	Ali	70	3.6	Islamabad	3	Old
3	Shoaib	37	2.9	Pindi	4	Young

In [133]:
```
#Lambda functions
old_young = lambda age: "Old" if age>40 else "Young"
```

In [134]:
```
frame['Status']=frame['Age'].apply(old_young)
```

In [135]:
```
frame
```

Out[135]:

	Name	Age	CGPA	Address	Semester	Status
0	Sarah	50	3.4	Karachi	1	Old
1	Ahmed	60	2.9	Lahore	2	Old
2	Ali	70	3.6	Islamabad	3	Old
3	Shoaib	37	2.9	Pindi	4	Young

In [136]:
```
employee = pd.DataFrame(
{
    'salary':[1000,2000,3000],
    'name':['Mark','Jordan','Yuaan']
}
)
```

In [137]:
```
employee
```

Out[137]:

	salary	name
0	1000	Mark
1	2000	Jordan
2	3000	Yuaan

In [139]:
```
employee['Increment']=\
        employee['salary'].apply(lambda salary: 50 if salary<=1000 else 25)
```

In [140]:
```
employee
```

Out[140]:

	salary	name	Increment
0	1000	Mark	50
1	2000	Jordan	25
2	3000	Yuaan	25

In [141]:
```
np.abs(employee['Increment'])
```

Out[141]:
```
0    50
1    25
2    25
Name: Increment, dtype: int64
```

In [142]:
```
#Apply numpy functions over dataframe
```

In [ ]:
```

```

In [1]:
```
import pandas as pd
```

```
D:\Anaconda3\lib\site-packages\pandas\compat\_optional.py:138: UserWarning: P
andas requires version '2.7.0' or newer of 'numexpr' (version '2.6.9' current
ly installed).
  warnings.warn(msg, UserWarning)
```

In [31]:
```
#Let's create a dataframe of cricket teams
stats = pd.DataFrame(
    {
        'teams': ['Pakistan','India','Srilanka','S. Africa'],
        'played' :[5,6,4,5],
        'points' : [10,8,8,6]
    },
    index=['t1','t2','t3','t4']
)
```

In [10]:
```
stats
```

Out[10]:

	teams	played	points
t1	Pakistan	5	10
t2	India	6	8
t3	Srilanka	4	8
t4	S. Africa	5	6

In [11]:
```
stats.head(2) #Display first two rows
```

Out[11]:

	teams	played	points
t1	Pakistan	5	10
t2	India	6	8

In [12]:
```
stats.tail(2) #Display last two rows
```

Out[12]:

	teams	played	points
t3	Srilanka	4	8
t4	S. Africa	5	6

In [13]:
```
stats['teams']
```

Out[13]:
```
t1     Pakistan
t2        India
t3     Srilanka
t4    S. Africa
Name: teams, dtype: object
```

In [14]:
```
stats.iloc[1]
```

Out[14]:
```
teams     India
played        6
points        8
Name: t2, dtype: object
```

In [15]:
```
stats.loc['t1']
```

Out[15]:
```
teams     Pakistan
played           5
points          10
Name: t1, dtype: object
```

In [17]:
```
#Getting numpy array
stats.values
```

Out[17]:
```
array([['Pakistan', 5, 10],
       ['India', 6, 8],
       ['Srilanka', 4, 8],
       ['S. Africa', 5, 6]], dtype=object)
```

In [21]:
```
import numpy as np
stats['No.'] = np.arange(1,5) # [1,2,3,4] ; New column added with specific val
```

In [22]:
```
stats
```

Out[22]:

	teams	played	points	No.
t1	Pakistan	5	10	1
t2	India	6	8	2
t3	Srilanka	4	8	3
t4	S. Africa	5	6	4

In [23]:
```
stats.iloc[0:3,2] #Specify row and then column
```

Out[23]:
```
t1    10
t2     8
t3     8
Name: points, dtype: int64
```

In [25]:
```
stats.iloc[0:3] #Only specify the rows
```

Out[25]:

	teams	played	points	No.
t1	Pakistan	5	10	1
t2	India	6	8	2
t3	Srilanka	4	8	3

In [38]:
```
new_df = stats.drop('teams',axis=1) #This is not in-place/ in-memory operation
```

In [39]:
```
display(stats)
display(new_df)
```

	teams	played	points	Teams	win
t1	Pakistan	5	10	Pakistn	5.0
t2	India	6	8	India	4.0
t3	Srilanka	4	8	Srilanka	4.0
t4	S. Africa	5	6	S. Africa	3.0

	played	points	Teams	win
t1	5	10	Pakistn	5.0
t2	6	8	India	4.0
t3	4	8	Srilanka	4.0
t4	5	6	S. Africa	3.0

In [40]:
```
stats.drop('teams',axis=1, inplace=True)
```

In [41]:
```
stats
```

Out[41]:

	played	points	Teams	win
t1	5	10	Pakistn	5.0
t2	6	8	India	4.0
t3	4	8	Srilanka	4.0
t4	5	6	S. Africa	3.0

In [35]:
```
#Apply
```

In [36]:
```
stats['Teams']=['Pakistan','India','Srilanka','S. Africa']
stats['win']=stats['points']/2
```

In [42]:
```
stats
```

Out[42]:

	played	points	Teams	win
t1	5	10	Pakistn	5.0
t2	6	8	India	4.0
t3	4	8	Srilanka	4.0
t4	5	6	S. Africa	3.0

In [46]:
```
stats['win']=stats['win'].astype('int') #Change the datatype to integer
```

In [47]:
```
stats
```

Out[47]:

	played	points	Teams	win
t1	5	10	Pakistn	5
t2	6	8	India	4
t3	4	8	Srilanka	4
t4	5	6	S. Africa	3

In [55]:
```
def calculate_loss_tie(row):
    return row['played'] - row['win']
```

In [58]:
```
stats['loss_tie']=stats.apply(calculate_loss_tie,axis=1)
#The function calculate_loss will be called four time with following rows:
#1  5   10  Pakistn 5 - row
#t2 6   8   India   4 - row
#t3 4   8   Srilanka    4
#t4 5   6   S. Africa   3
```

In [66]:
```
#Above work is equal to stats['loss_tie'] = stats['played'] - stats['win']
stats
```

Out[66]:

	played	points	Teams	win	loss_tie
t1	5	10	Pakistn	5	0
t2	6	8	India	4	2
t3	4	8	Srilanka	4	0
t4	5	6	S. Africa	3	2

In [60]:
```
#Get the team with the maximum points
```

In [64]:
```
index = stats['points'].idxmax() #Get the index of the team with maximum points
```

In [65]:
```
stats.loc[index]
```

Out[65]:
```
played           5
points          10
Teams       Pakistn
win              5
loss_tie         0
Name: t1, dtype: object
```

In [67]:
```
np.sqrt(stats['played'])
```

Out[67]:
```
t1    2.236068
t2    2.449490
t3    2.000000
t4    2.236068
Name: played, dtype: float64
```

In [68]:
```
stats.describe()
```

Out[68]:

	played	points	win	loss_tie
count	4.000000	4.000000	4.000000	4.000000
mean	5.000000	8.000000	4.000000	1.000000
std	0.816497	1.632993	0.816497	1.154701
min	4.000000	6.000000	3.000000	0.000000
25%	4.750000	7.500000	3.750000	0.000000
50%	5.000000	8.000000	4.000000	1.000000
75%	5.250000	8.500000	4.250000	2.000000
max	6.000000	10.000000	5.000000	2.000000

In [71]:
```
display(stats)
stats.sum()
```

	played	points	Teams	win	loss_tie
t1	5	10	Pakistn	5	0
t2	6	8	India	4	2
t3	4	8	Srilanka	4	0
t4	5	6	S. Africa	3	2

Out[71]:
```
played                                20
points                                32
Teams       PakistnIndiaSrilankaS. Africa
win                                   16
loss_tie                               4
dtype: object
```

In [72]:
```
#Assignment: Practice various descriptive statistics method
```

In [73]:
```
stats.to_csv('data.csv')
#stats.to_csv('data.csv',index=False
```

In [82]:
```
my_data = pd.read_csv('data.csv',index_col=0)
```

In [83]:
```
my_data
```

Out[83]:

	played	points	Teams	win	loss_tie
t1	5	10	Pakistn	5	0
t2	6	8	India	4	2
t3	4	8	Srilanka	4	0
t4	5	6	S. Africa	3	2

In [85]:
```
my_data.loc['t4']
```

Out[85]:
```
played              5
points              6
Teams        S. Africa
win                 3
loss_tie            2
Name: t4, dtype: object
```

In [87]:
```
my_data = pd.read_excel('data.xlsx',header=None)
display(my_data)
```

	0	1	2	3	4
0	5	10	Pakistn	5	0
1	6	8	India	4	2
2	4	8	Srilanka	4	0
3	5	6	S. Africa	3	2

In [88]:
```
my_data.columns=['Played','Points','Team','Wins','Loss/Tie']
```

In [89]:
```
my_data
```

Out[89]:

	Played	Points	Team	Wins	Loss/Tie
0	5	10	Pakistn	5	0
1	6	8	India	4	2
2	4	8	Srilanka	4	0
3	5	6	S. Africa	3	2

In [90]:
```
#Assignment: Read/ write data in various formats
```

In [91]:
```
df = pd.read_csv('students.csv')
```

In [93]:
```
df.describe()
```

Out[93]:

	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	Best of Assignments	Sess
count	48.000000	45.000000	45.000000	48.000000	45.000000	36.000000	48.000000	48.00
mean	22.937500	7.288889	4.866667	7.020833	7.755556	9.444444	8.937500	38.89
std	5.236558	1.561209	1.455397	1.973113	1.170772	1.629100	2.127892	7.17
min	9.000000	4.000000	1.000000	0.000000	5.000000	2.000000	0.000000	18.00
25%	20.000000	6.000000	4.000000	6.000000	7.000000	9.000000	8.750000	36.75
50%	25.000000	8.000000	5.000000	7.000000	8.000000	10.000000	10.000000	41.00
75%	27.000000	9.000000	6.000000	9.000000	8.000000	10.000000	10.000000	44.00
max	30.000000	9.000000	7.000000	9.000000	10.000000	11.000000	11.000000	46.00

In [98]:
```
display(df)
df.isna().sum() #Number of missing values
```

	Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	B Assign
0	022-14-19987	BS(CS)	Abdul Basit	28	8.0	3.0	8	7.0	9.0	
1	022-14-110233	BS(CS)	Adeel Ahmed	17	NaN	5.0	5	8.0	10.0	
2	022-14-110585	BS(CS)	Afrah Zareen	18	5.0	2.0	5	8.0	10.0	
3	022-14-19718	BS(CS)	Ahmed Ali Raza	14	7.0	2.0	7	NaN	2.0	
4	022-14-110648	BS(CS)	Ahsan Ali Vohra	27	7.0	6.0	7	7.0	9.0	
5	022-14-110232	BS(CS)	Ameer Hamza	25	9.0	6.0	9	8.0	10.0	
6	022-14-	BS(CS)	Anas Ali	28	5.0	6.0	6	8.0	10.0	

In [101]:
```
(~df.isna()).sum()
```

Out[101]:
```
Student Code            48
Degree                  48
Student Name            48
Mid                     48
Quiz 1                  45
Quiz 2                  45
Best of Quizzes         48
Assignment 1            45
Assignment 2            36
Best of Assignments     48
Total Sessional (50)    48
Final (50)              47
Total (100)             48
Grade                   48
dtype: int64
```

In [102]:
```
df.columns
```

Out[102]:
```
Index(['Student Code', 'Degree', 'Student Name', 'Mid', 'Quiz 1', 'Quiz 2',
       'Best of Quizzes', 'Assignment 1', 'Assignment 2',
       'Best of Assignments', 'Total Sessional (50)', 'Final (50)',
       'Total (100)', 'Grade'],
      dtype='object')
```

In [103]:
```
df[df['Grade']=='A']
```

Out[103]:

	Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	Best Assignmer
7	022-14-110388	BS(CS)	Aneebullah Niazi	26	9.0	6.0	9	8.0	10.0	
9	022-14-110599	BS(CS)	Arsalan	28	8.0	6.0	8	8.0	NaN	
10	022-14-110214	BS(CS)	Fatima Haider Warsi	30	8.0	7.0	8	8.0	NaN	
19	022-14-110222	BS(CS)	Mohammad Hunain	27	9.0	6.0	9	8.0	10.0	
25	022-14-110105	BS(CS)	Muhammad Faraz	26	9.0	6.0	9	8.0	10.0	
34	022-14-110413	BS(CS)	Nazeer Bin Zafar	25	9.0	6.0	9	8.0	10.0	
40	022-14-110584	BS(CS)	Sharif Taqi	27	9.0	6.0	9	8.0	10.0	
45	022-14-110400	BS(CS)	Syeda Sabika Raza	27	9.0	6.0	9	9.0	NaN	

In [104]:
```
df.isna().sum()
```

Out[104]:
```
Student Code             0
Degree                   0
Student Name             0
Mid                      0
Quiz 1                   3
Quiz 2                   3
Best of Quizzes          0
Assignment 1             3
Assignment 2            12
Best of Assignments      0
Total Sessional (50)     0
Final (50)               1
Total (100)              0
Grade                    0
dtype: int64
```

In [105]:
```
df[df['Quiz 1'].isna()]
```

Out[105]:

	Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	Best Assignmer
1	022-14-110233	BS(CS)	Adeel Ahmed	17	NaN	5.0	5	8.0	10.0	
15	022-14-19916	BS(CS)	Haseeb Sajid	18	NaN	NaN	0	NaN	NaN	
32	022-14-110035	BS(CS)	Muhammad Wajahat Khan	9	NaN	2.0	2	8.0	10.0	

In [106]:
```
df['Quiz 1'].fillna(0,inplace=True)
```

In [108]:
```
df[df['Quiz 2'].isna()]
```

Out[108]:

	Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	Best Assignmer
15	022-14-19916	BS(CS)	Haseeb Sajid	18	0.0	NaN	0	NaN	NaN	
23	022-14-19983	BS(CS)	Muhammad Ali Iqbal	26	5.0	NaN	5	8.0	10.0	
26	022-14-110370	BS(CS)	Muhammad Ghazali Faridi	27	6.0	NaN	6	6.0	8.0	

In [109]:
```
df.fillna(0)
```

Out[109]:

	Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	B Assign
0	022-14-19987	BS(CS)	Abdul Basit	28	8.0	3.0	8	7.0	9.0	
1	022-14-110233	BS(CS)	Adeel Ahmed	17	0.0	5.0	5	8.0	10.0	
2	022-14-110585	BS(CS)	Afrah Zareen	18	5.0	2.0	5	8.0	10.0	
3	022-14-19718	BS(CS)	Ahmed Ali Raza	14	7.0	2.0	7	0.0	2.0	
4	022-14-110648	BS(CS)	Ahsan Ali Vohra	27	7.0	6.0	7	7.0	9.0	
5	022-14-110232	BS(CS)	Ameer Hamza	25	9.0	6.0	9	8.0	10.0	
6	022-14-	BS(CS)	Anas Ali	28	5.0	6.0	6	8.0	10.0	

In [111]:
```
df.dropna() #Drop the entire row with nan values
```

Out[111]:

	Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	Best Assignmer
0	022-14-19987	BS(CS)	Abdul Basit	28	8.0	3.0	8	7.0	9.0	
1	022-14-110233	BS(CS)	Adeel Ahmed	17	0.0	5.0	5	8.0	10.0	
2	022-14-110585	BS(CS)	Afrah Zareen	18	5.0	2.0	5	8.0	10.0	
4	022-14-110648	BS(CS)	Ahsan Ali Vohra	27	7.0	6.0	7	7.0	9.0	
5	022-14-110232	BS(CS)	Ameer Hamza	25	9.0	6.0	9	8.0	10.0	
6	022-14-110588	BS(CS)	Anas Ali Khan	28	5.0	6.0	6	8.0	10.0	
7	022-14-110388	BS(CS)	Aneebullah Niazi	26	9.0	6.0	9	8.0	10.0	
8	022-14-110601	BS(CS)	Areesha Sohail	19	9.0	4.0	9	7.0	9.0	
12	022-15-110994	BS(CS)	Hafiza Tooba Akbani	23	7.0	5.0	7	8.0	10.0	
13	022-14-110600	BS(CS)	Hamza Abdul Jabbar	24	8.0	4.0	8	8.0	10.0	
14	022-14-110389	BS(CS)	Hareem Afshan	21	7.0	4.0	7	6.0	8.0	
16	022-14-110596	BS(CS)	Hassam Ahmed	23	5.0	5.0	5	7.0	9.0	
17	022-14-110396	BS(CS)	Khalid Anwer	20	8.0	5.0	8	9.0	11.0	
18	022-14-110223	BS(CS)	Madiha Jabeen	16	5.0	2.0	5	8.0	10.0	
19	022-14-110222	BS(CS)	Mohammad Hunain	27	9.0	6.0	9	8.0	10.0	
20	022-14-110412	BS(CS)	Muhammad Aamir	24	7.0	6.0	7	7.0	9.0	
21	022-14-110593	BS(CS)	Muhammad Abdul Rehman Siddiqui	20	9.0	6.0	9	9.0	11.0	
24	022-14-110215	BS(CS)	Muhammad Bilal	28	5.0	4.0	5	9.0	11.0	
25	022-14-110105	BS(CS)	Muhammad Faraz	26	9.0	6.0	9	8.0	10.0	
27	022-14-110452	BS(CS)	Muhammad Osama Khan	27	8.0	6.0	8	8.0	10.0	

	Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	Best Assignmer
28	022-14-110387	BS(CS)	Muhammad Saqib Intizar	22	6.0	6.0	6	9.0	11.0	
32	022-14-110035	BS(CS)	Muhammad Wajahat Khan	9	0.0	2.0	2	8.0	10.0	
33	022-14-19919	BS(CS)	Muhammad Younus Baig	20	4.0	2.0	4	8.0	10.0	
34	022-14-110413	BS(CS)	Nazeer Bin Zafar	25	9.0	6.0	9	8.0	10.0	
38	022-14-110229	BS(CS)	Sadaquat Rafique	9	6.0	6.0	6	5.0	7.0	
39	022-14-110107	BS(CS)	Sania Iqbal	28	4.0	4.0	4	9.0	11.0	
40	022-14-110584	BS(CS)	Sharif Taqi	27	9.0	6.0	9	8.0	10.0	
41	022-14-110225	BS(CS)	Shariqa Ahmad	20	7.0	6.0	7	8.0	10.0	
42	022-14-110587	BS(CS)	Sumbul Rehman	28	5.0	6.0	6	8.0	10.0	
43	022-14-110451	BS(CS)	Syed Faizan Uddin	28	9.0	4.0	9	6.0	8.0	
46	022-14-19911	BS(CS)	Usman Khan	25	8.0	5.0	8	8.0	10.0	
47	022-14-110219	BS(CS)	Waqar Ahmed	11	9.0	5.0	9	5.0	7.0	

In [112]:
```
df
```

Out[112]:

	Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	B Assigni
0	022-14-19987	BS(CS)	Abdul Basit	28	8.0	3.0	8	7.0	9.0	
1	022-14-110233	BS(CS)	Adeel Ahmed	17	0.0	5.0	5	8.0	10.0	
2	022-14-110585	BS(CS)	Afrah Zareen	18	5.0	2.0	5	8.0	10.0	
3	022-14-19718	BS(CS)	Ahmed Ali Raza	14	7.0	2.0	7	NaN	2.0	
4	022-14-110648	BS(CS)	Ahsan Ali Vohra	27	7.0	6.0	7	7.0	9.0	
5	022-14-110232	BS(CS)	Ameer Hamza	25	9.0	6.0	9	8.0	10.0	
6	022-14-	BS(CS)	Anas Ali	28	5.0	6.0	6	8.0	10.0	

In [113]:
```
df.dropna(how='all')
```

Out[113]:

	Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	B Assigni
0	022-14-19987	BS(CS)	Abdul Basit	28	8.0	3.0	8	7.0	9.0	
1	022-14-110233	BS(CS)	Adeel Ahmed	17	0.0	5.0	5	8.0	10.0	
2	022-14-110585	BS(CS)	Afrah Zareen	18	5.0	2.0	5	8.0	10.0	
3	022-14-19718	BS(CS)	Ahmed Ali Raza	14	7.0	2.0	7	NaN	2.0	
4	022-14-110648	BS(CS)	Ahsan Ali Vohra	27	7.0	6.0	7	7.0	9.0	
5	022-14-110232	BS(CS)	Ameer Hamza	25	9.0	6.0	9	8.0	10.0	
6	022-14-	BS(CS)	Anas Ali	28	5.0	6.0	6	8.0	10.0	

In [117]:
```
import sys
df.to_csv(sys.stdout,sep='\t')
```

	Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	Best of Assignments	Total	Sessional (50)	Final (50)	Total (100)	Grade
0	022-14-19987	BS(CS)	Abdul Basit	28	8.0	3.0	8	7.0	9.0	9	45	25.0	70	B
1	022-14-110233	BS(CS)	Adeel Ahmed	17	0.0	5.0	5	8.0	10.0	10	32	18.0	50	F
2	022-14-110585	BS(CS)	Afrah Zareen	18	5.0	2.0	5	8.0	10.0	10	33	30.0	63	C
3	022-14-19718	BS(CS)	Ahmed Ali Raza	14	7.0	2.0	7	2.0	2	23	23.0	46	F		
4	022-14-110648	BS(CS)	Ahsan Ali Vohra	27	7.0	6.0	7	7.0	9.0	9	43	34.0	77	B	
5	022-14-110232	BS(CS)	Ameer Hamza	25	9.0	6.0	9	8.0	10.0	10	44	27.0	71	B	
6	022-14-110588	BS(CS)	Anas Ali Khan	28	5.0	6.0	6	8.0	10.0	10	44	30.0	74	B	
7	022-14-110388	BS(CS)	Aneebullah Niazi	26	9.0	6.0	9	8.0	10.0	10	45	40.0	85	A	
8	022-14-110601	BS(CS)	Areesha Sohail	19	9.0	4.0	9	7.0	9.0	9	37	24.0	61	C	
9	022-14-110599	BS(CS)	Arsalan	28	8.0	6.0	8	8.0	8	44	40.0	84	A		
10	022-14-110214	BS(CS)	Fatima Haider Warsi	30	8.0	7.0	8	8.0		8	46	45.0	91	A	
11	022-14-110591	BS(CS)	Habib Ullah	28	8.0	5.0	8	5.0		5	41	35.0	76	B	
12	022-15-110994	BS(CS)	Hafiza Tooba Akbani	23	7.0	5.0	7	8.0	10.0	10	40	33.0	73	B	
13	022-14-110600	BS(CS)	Hamza Abdul Jabbar	24	8.0	4.0	8	8.0	10.0	10	42	25.0	67	C	
14	022-14-110389	BS(CS)	Hareem Afshan	21	7.0	4.0	7	6.0	8.0	8	36	29.0	65	C	
15	022-14-19916	BS(CS)	Haseeb Sajid	18	0.0		0	0	18		18	F			
16	022-14-110596	BS(CS)	Hassam Ahmed	23	5.0	5.0	5	7.0	9.0	9	37	35.0	72	B	
17	022-14-110396	BS(CS)	Khalid Anwer	20	8.0	5.0	8	9.0	11.0	11	39	31.0	70	B	
18	022-14-110223	BS(CS)	Madiha Jabeen	16	5.0	2.0	5	8.0	10.0	10	31	29.0	60	C	
19	022-14-110222	BS(CS)	Mohammad Hunain	27	9.0	6.0	9	8.0	10.0	10	46	45.0	91	A	
20	022-14-110412	BS(CS)	Muhammad Aamir	24	7.0	6.0	7	7.0	9.0	9	40	31.0	71	B	
21	022-14-110593	BS(CS)	Muhammad Abdul Rehman Siddiqui	20	9.0	6.0	9	9.0	11.0	11	40	20.0	60	C	
22	022-14-110398	BS(CS)	Muhammad Abdullah	20	7.0	6.0	7	7.0		7	34	23.0	57	F	
23	022-14-19983	BS(CS)	Muhammad Ali Iqbal	26	5.0		5	8.0	10.0	10	41	21.0	62	C	
24	022-14-110215	BS(CS)	Muhammad Bilal	28	5.0	4.0	5	9.0	11.0	11	44	33.0	77	B	
25	022-14-110105	BS(CS)	Muhammad Faraz	26	9.0	6.0	9	8.0	10.0	10	45	41.0	86	A	
26	022-14-110370	BS(CS)	Muhammad Ghazali Faridi	27	6.0		6	6.0	8.0	8	41	24.0	65	C	

27	022-14-110452	BS(CS)	Muhammad Osama Khan	27	8.0	6.0	8	8.0	10.0	10	45	26.0	71	B
28	022-14-110387	BS(CS)	Muhammad Saqib Intizar	22	6.0	6.0	6	9.0	11.0	11	39	18.0	57	F
29	022-14-110217	BS(CS)	Muhammad Shahroz Khurshid	17	7.0	1.0	7	6.0		6	30	9.0	39	F
30	022-14-110401	BS(CS)	Muhammad Shozab	23	8.0	6.0	8	9.0		9	40	32.0	72	B
31	022-14-110231	BS(CS)	Muhammad Taha Hasnain	27	7.0	5.0	7	10.0		10	44	29.0	73	B
32	022-14-110035	BS(CS)	Muhammad Wajahat Khan	9	0.0	2.0	2	8.0	10.0	10	21	8.0	29	F
33	022-14-19919	BS(CS)	Muhammad Younus Baig	20	4.0	2.0	4	8.0	10.0	10	34	17.0	51	F
34	022-14-110413	BS(CS)	Nazeer Bin Zafar	25	9.0	6.0	9	8.0	10.0	10	44	42.0	86	A
35	022-14-19923	BS(CS)	Rabi Ahmed	20	8.0	5.0	8	10.0	10	38	28.0	66	C	
36	022-14-110582	BS(CS)	Rida Nasim	25	9.0	5.0	9	9.0		9	43	27.0	70	B
37	022-14-110230	BS(CS)	Sadaf Nosheen	27	9.0	5.0	9	9.0		9	45	30.0	75	B
38	022-14-110229	BS(CS)	Sadaquat Rafique	9	6.0	6.0	6	5.0	7.0	7	22	17.0	39	F
39	022-14-110107	BS(CS)	Sania Iqbal	28	4.0	4.0	4	9.0	11.0	11	43	25.0	68	C
40	022-14-110584	BS(CS)	Sharif Taqi	27	9.0	6.0	9	8.0	10.0	10	46	35.0	81	A
41	022-14-110225	BS(CS)	Shariqa Ahmad	20	7.0	6.0	7	8.0	10.0	10	37	20.0	57	F
42	022-14-110587	BS(CS)	Sumbul Rehman	28	5.0	6.0	6	8.0	10.0	10	44	23.0	67	C
43	022-14-110451	BS(CS)	Syed Faizan Uddin	28	9.0	4.0	9	6.0	8.0	8	45	34.0	79	B
44	022-14-110589	BS(CS)	Syed Sohaib	25	7.0	5.0	7	9.0		9	41	22.0	63	C
45	022-14-110400	BS(CS)	Syeda Sabika Raza	27	9.0	6.0	9	9.0		9	45	35.0	80	A
46	022-14-19911	BS(CS)	Usman Khan	25	8.0	5.0	8	8.0	10.0	10	43	22.0	65	C
47	022-14-110219	BS(CS)	Waqar Ahmed	11	9.0	5.0	9	5.0	7.0	7	27	19.0	46	F

In [118]:
```
!pip install bs4
```

```
Collecting bs4
  Using cached bs4-0.0.1.tar.gz (1.1 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.11.2-py3-none-any.whl (129 kB)
     ---------------------------------- 129.4/129.4 kB 637.2 kB/s eta 0:00:
00
Collecting soupsieve>1.2
  Downloading soupsieve-2.3.2.post1-py3-none-any.whl (37 kB)
Installing collected packages: soupsieve, beautifulsoup4, bs4
  Running setup.py install for bs4: started
  Running setup.py install for bs4: finished with status 'done'
Successfully installed beautifulsoup4-4.11.2 bs4-0.0.1 soupsieve-2.3.2.post1

  DEPRECATION: bs4 is being installed using the legacy 'setup.py install' met
hod, because it does not have a 'pyproject.toml' and the 'wheel' package is n
ot installed. pip 23.1 will enforce this behaviour change. A possible replace
ment is to enable the '--use-pep517' option. Discussion can be found at http
s://github.com/pypa/pip/issues/8559 (https://github.com/pypa/pip/issues/8559)

[notice] A new release of pip available: 22.3.1 -> 23.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [ ]:
```
```

In [1]:
```
#Review of previous lecture
#Series : How to create series, extract specific elemnent, indexing
#DataFrame: Creating dataframe, extracting columns, filter, iloc/ loc, apply,re
```

In [2]:
```
#WebScrapping libraries
#BeautifulSoup
#Scrappy
```

In [3]:
```
#HTML - Hyper tex markup language
#Hierarchical representation of a page
#Tags, attributes
```

In [4]:
```
url = 'http://www.espncricinfo.com/rankings/content/page/211271.html'
```

In [5]:
```
from requests import get
from bs4 import BeautifulSoup
```

In [6]:
```
response = get(url) #Getting webpage from the website
```

In [7]:
```
print(response.text) #Lets see what we got after fetching the URL contents
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.
w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!-- hostname: web03, edition-view: espncricinfo-en-pk, country: pk, cluste
r: pak, created: 2023-02-12 06:15:30 -->
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:fb="http://www.facebook.co
m/2008/fbml" xmlns:og="http://opengraphprotocol.org/schema/" xmlns:fb="htt
p://developers.facebook.com/schema/" >
<head>
 <script type="text/javascript">var _sf_startpt=(new Date()).getTime()</scr
ipt>
 <meta name="google-site-verification" content="ZxdgH3XglRg0Bsy-Ho2RnO3EE4n
Rs53FloLS6fkt_nc" />
 <meta
        name="viewport"
        content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
user-scalable=0"
      />
 <title>ICC Cricket Rankings - Current ICC Rankings for Tests, ODIs, T20 Cr
```

In [8]:
```
html_soup = BeautifulSoup(response.text, 'html.parser')
```

In [13]:
```
ranking = html_soup.find_all('table', class_ = 'StoryengineTable')
```

In [ ]:
```
#Lets explore a bit
```

In [25]:
```
len(ranking)
```

Out[25]:
```
5
```

In [26]:
```
ranking
```

Out[26]:
```
[<table class="StoryengineTable">
 <caption>08 January 2023</caption>
 <tr class="head">
 <th class="left">Pos</th>
 <th class="left">Team</th>
 <th class="left">Matches</th>
 <th class="left">Points</th>
 <th class="left">Rating</th>
 </tr>
 <tr>
 <td class="left">1</td>
 <td class="left">Australia</td>
 <td class="left">29</td>
 <td class="left">3668</td>
 <td class="left">126</td>
 </tr>
 <tr>
 <td class="left">2</td>
 <td class="left">India</td>
```

In [18]:
```
print(ranking[0].caption.text)
```

```
08 January 2023
```

In [20]:
```
trs = ranking[0].find_all('tr')
```

In [28]:
```
trs[1]
```

Out[28]:
```
<tr>
<td class="left">1</td>
<td class="left">Australia</td>
<td class="left">29</td>
<td class="left">3668</td>
<td class="left">126</td>
</tr>
```

In [38]:
```
#Print ranking for only one format
for th in trs[0].find_all('th'): #Heading row
        print('{:13}'.format(th.text), end=" ")
print() #Move to next line
for tr in trs[1:]:  #All rows
    for td in tr.find_all('td'): #All columns
        print('{:13}'.format(td.text), end=" ")
    print()
```

```
Pos           Team          Matches       Points        Rating
1             Australia     29            3668          126
2             India         32            3690          115
3             England       47            5017          107
4             South Africa  29            2952          102
5             New Zealand   30            2965          99
6             Pakistan      30            2638          88
7             Sri Lanka     26            2282          88
8             West Indies   28            2198          79
9             Bangladesh    25            1161          46
10            Zimbabwe      6             148           25
```

In [ ]:
```
#Now print rankings for all the format
```

In [50]:
```
def print_ranking(trs):
    for th in trs[0].find_all('th'): #Heading row
        print('{:20}'.format(th.text), end=" ")
    print() #Move to next line
    for tr in trs[1:]:  #All rows
        for td in tr.find_all('td'): #All columns
            print('{:20}'.format(td.text), end=" ")
        print()
```

In [52]:
```
for r in ranking: #For all the rankings
    print(r.caption.text)
    print_ranking(r.find_all('tr'))
    print('\n\n\n')
```

```
08 January 2023
Pos                  Team                 Matches              Points
Rating
1                    Australia            29                   3668
126
2                    India                32                   3690
115
3                    England              47                   5017
107
4                    South Africa         29                   2952
102
5                    New Zealand          30                   2965
99
6                    Pakistan             30                   2638
88
7                    Sri Lanka            26                   2282
88
8                    West Indies          28                   2198
79
```

In [59]:
```
#Now we will print the format details as well
format=html_soup.find_all('div' , class_="ciPhotoContainer")[0].find_all('h3')
```

In [58]:
```
format
```

Out[58]:
```
[<div class="ciPhotoContainer">
 <p class="news-body">
 <h1> <font size="4">ICC rankings for Tests, ODIs, T20 &amp;amp; Women's ODI an
d T20</font> </h1>
 </p>
 <p class="news-body">
 <h3>ICC Test Rankings</h3>
 </p>
 <p class="news-body">
 <!--- START TEST CHAMPIONSHIP --->
 </p>
 <p class="news-body"></p>
 <p class="news-body">
 <table class="StoryengineTable">
 <caption>08 January 2023</caption>
 <tr class="head">
 <th class="left">Pos</th>
 <th class="left">Team</th>
 <th class="left">Matches</th>
```

In [61]:
```
i=0
for r in ranking: #For all the rankings
    print(r.caption.text)
    print(format[i].text)
    print_ranking(r.find_all('tr'))
    print('\n\n\n')
    i+=1
```

```
08 January 2023
ICC Test Rankings
Pos                 Team                Matches             Points
Rating
1                   Australia           29                  3668
126
2                   India               32                  3690
115
3                   England             47                  5017
107
4                   South Africa        29                  2952
102
5                   New Zealand         30                  2965
99
6                   Pakistan            30                  2638
88
7                   Sri Lanka           26                  2282
88
8                   West Indies         28                  2198
```

In [63]:
```
#Let's do the same thing with enumerate
for i,r in enumerate(ranking): #For all the rankings
    print(r.caption.text)
    print(format[i].text)
    print_ranking(r.find_all('tr'))
    print('\n\n\n')
```

```
08 January 2023
ICC Test Rankings
Pos                 Team                Matches             Points
Rating
1                   Australia           29                  3668
126
2                   India               32                  3690
115
3                   England             47                  5017
107
4                   South Africa        29                  2952
102
5                   New Zealand         30                  2965
99
6                   Pakistan            30                  2638
88
7                   Sri Lanka           26                  2282
88
8                   West Indies         28                  2198
```

In [74]:
```
def csv_ranking(trs):
    data = ""
    for th in trs[0].find_all('th'): #Heading row
        data+=th.text+","
    data = data[:-1] #Remove last comma
    data+='\n' #Move to next line
    for tr in trs[1:]:  #All rows
        for td in tr.find_all('td'): #All columns
            data+=td.text+","
        data = data[:-1]
        data+='\n'
    return data
```

In [78]:
```
for i,r in enumerate(ranking):
    file = format[i].text
    data = csv_ranking(r.find_all('tr'))
    with open(file+".csv","w") as f:
        f.writelines(data)
```

In [79]:
```
import pandas as pd
```

```
D:\Anaconda3\lib\site-packages\pandas\compat\_optional.py:138: UserWarning: P
andas requires version '2.7.0' or newer of 'numexpr' (version '2.6.9' current
ly installed).
  warnings.warn(msg, UserWarning)
```

In [80]:
```
frame = pd.read_csv('ICC Twenty20 Rankings.csv')
```

In [81]:
```
frame
```

Out[81]:

	Pos	Team	Matches	Points	Rating
0	1	India	69	18445	267
1	2	England	49	13029	266
2	3	Pakistan	55	14168	258
3	4	South Africa	41	10510	256
4	5	New Zealand	53	13371	252
...	...	...	...	...	...
80	81	Thailand	10	0	0
81	82	Eswatini	18	0	0
82	83	Seychelles	10	0	0
83	84	Estonia	12	0	0
84	85	Cameroon	13	0	0

85 rows × 5 columns

In [ ]:

In [48]:
```
#Pandas revision
```

In [1]:
```
import pandas as pd
```

```
D:\Anaconda3\lib\site-packages\pandas\compat\_optional.py:138: UserWarning: P
andas requires version '2.7.0' or newer of 'numexpr' (version '2.6.9' current
ly installed).
  warnings.warn(msg, UserWarning)
```

In [2]:
```
frame = pd.read_csv('students.csv')
```

In [6]:
```
frame['scaled marks']=frame['Total (100)'].apply(lambda x:x+5)
```

In [7]:
```
frame
```

Out[7]:

	Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	B Assign
0	022-14-19987	BS(CS)	Abdul Basit	28	8.0	3.0	8	7.0	9.0	
1	022-14-110233	BS(CS)	Adeel Ahmed	17	NaN	5.0	5	8.0	10.0	
2	022-14-110585	BS(CS)	Afrah Zareen	18	5.0	2.0	5	8.0	10.0	
3	022-14-19718	BS(CS)	Ahmed Ali Raza	14	7.0	2.0	7	NaN	2.0	
4	022-14-110648	BS(CS)	Ahsan Ali Vohra	27	7.0	6.0	7	7.0	9.0	
5	022-14-110232	BS(CS)	Ameer Hamza	25	9.0	6.0	9	8.0	10.0	
6	022-14-	BS(CS)	Anas Ali	28	5.0	6.0	6	8.0	10.0	

In [8]:
```
#Lambda function
```

In [11]:
```
y = lambda x:2*x  #lambda arg: operation
```

In [10]:
```
y(6)
```

Out[10]: 12

In [12]:
```
frame['Total (100)'].count()
```

Out[12]: 48

In [13]:
```
frame['Total (100)'].argmin()
```

Out[13]: 15

In [24]:
```
def min(row):
    if row['Assignment 1']>row['Assignment 2']:
        return row['Assignment 2']
    else:
        return row['Assignment 1']
frame.apply(min,axis=1)
```

Out[24]:
```
0      7.0
1      8.0
2      8.0
3      NaN
4      7.0
5      8.0
6      8.0
7      8.0
8      7.0
9      8.0
10     8.0
11     5.0
12     8.0
13     8.0
14     6.0
15     NaN
16     7.0
17     9.0
18     8.0
19     8.0
20     7.0
21     9.0
22     7.0
23     8.0
24     9.0
25     8.0
26     6.0
27     8.0
28     9.0
29     6.0
30     9.0
31    10.0
32     8.0
33     8.0
34     8.0
35     NaN
36     9.0
37     9.0
38     5.0
39     9.0
40     8.0
41     8.0
42     8.0
43     6.0
44     9.0
45     9.0
46     8.0
47     5.0
dtype: float64
```

In [25]:
```
#Assignment: Read data in different formats
```

In [26]:
```
frame
```

Out[26]:

	Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	B Assignı
0	022-14-19987	BS(CS)	Abdul Basit	28	8.0	3.0	8	7.0	9.0	
1	022-14-110233	BS(CS)	Adeel Ahmed	17	NaN	5.0	5	8.0	10.0	
2	022-14-110585	BS(CS)	Afrah Zareen	18	5.0	2.0	5	8.0	10.0	
3	022-14-19718	BS(CS)	Ahmed Ali Raza	14	7.0	2.0	7	NaN	2.0	
4	022-14-110648	BS(CS)	Ahsan Ali Vohra	27	7.0	6.0	7	7.0	9.0	
5	022-14-110232	BS(CS)	Ameer Hamza	25	9.0	6.0	9	8.0	10.0	
6	022-14-	BS(CS)	Anas Ali	28	5.0	6.0	6	8.0	10.0	

In [28]:
```
frame = pd.read_csv('students.csv',index_col='Student Code')
display(frame)
```

Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	Best c Assignment
022-14-19987	BS(CS)	Abdul Basit	28	8.0	3.0	8	7.0	9.0	
022-14-110233	BS(CS)	Adeel Ahmed	17	NaN	5.0	5	8.0	10.0	1
022-14-110585	BS(CS)	Afrah Zareen	18	5.0	2.0	5	8.0	10.0	1
022-14-19718	BS(CS)	Ahmed Ali Raza	14	7.0	2.0	7	NaN	2.0	
022-14-110648	BS(CS)	Ahsan Ali Vohra	27	7.0	6.0	7	7.0	9.0	
022-14-	BS(CS)	Ameer	25	9.0	6.0	9	8.0	10.0	1

In [49]:
```
#Removing outliers
```

In [37]:
```
import numpy as np
mean = frame['Total (100)'].mean()
std = frame['Total (100)'].std()
lower_range = mean - 3*std
upper_range = mean + 3*std
```

In [38]:
```
filtered_data = frame[frame['Total (100)'] < upper_range][frame['Total (100)']
```

In [39]:
```
filtered_data
```

Out[39]:

Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	Best o Assignment
022-14-19987	BS(CS)	Abdul Basit	28	8.0	3.0	8	7.0	9.0	
022-14-110233	BS(CS)	Adeel Ahmed	17	NaN	5.0	5	8.0	10.0	1
022-14-110585	BS(CS)	Afrah Zareen	18	5.0	2.0	5	8.0	10.0	1
022-14-19718	BS(CS)	Ahmed Ali Raza	14	7.0	2.0	7	NaN	2.0	
022-14-110648	BS(CS)	Ahsan Ali Vohra	27	7.0	6.0	7	7.0	9.0	
022-14-	BS(CS)	Ameer	25	9.0	6.0	9	8.0	10.0	1

In [41]:
```
frame['Assignment 1'].fillna(frame['Assignment 1'].mean(),inplace=True)
```

In [42]:
```
frame
```

Out[42]:

Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	Best o Assignment
022-14-19987	BS(CS)	Abdul Basit	28	8.0	3.0	8	7.000000	9.0	
022-14-110233	BS(CS)	Adeel Ahmed	17	NaN	5.0	5	8.000000	10.0	1
022-14-110585	BS(CS)	Afrah Zareen	18	5.0	2.0	5	8.000000	10.0	1
022-14-19718	BS(CS)	Ahmed Ali Raza	14	7.0	2.0	7	7.755556	2.0	
022-14-110648	BS(CS)	Ahsan Ali Vohra	27	7.0	6.0	7	7.000000	9.0	
022-14-	BS(CS)	Ameer	25	9.0	6.0	9	8.000000	10.0	1

In [ ]:
```
#Database
```

In [44]:
```
import sqlite3 as s3
```

In [45]:
```
query = "create table Student(sId varchar,sname varchar)" #Create a table
```

In [71]:
```
con = s3.connect('student.sqlite') #Make the connection
```

In [47]:
```
con.execute(query) #Execute query
```

Out[47]:
```
<sqlite3.Cursor at 0x22e17cc3e30>
```

In [50]:
```
data = [
    (1,'Ali'),
    (2,'Fatima'),
    (3,'Asad')
]
```

In [52]:
```
query = "INSERT INTO Student values(?,?)"
```

In [53]:
```
con.executemany(query,data)
```

Out[53]:
```
<sqlite3.Cursor at 0x22e17c83e30>
```

In [54]:
```
con.commit()
```

In [72]:
```
query = "select sId, sName from Student"
```

In [73]:
```
cursor = con.execute(query)
```

In [74]:
```
rows = cursor.fetchall()
```

In [75]:
```
rows
```

Out[75]:
```
[('1', 'Ali'), ('2', 'Fatima'), ('3', 'Asad')]
```

In [76]:
```
cursor.description
```

Out[76]:
```
(('sId', None, None, None, None, None, None),
 ('sname', None, None, None, None, None, None))
```

In [70]:
```
#DDL : Create, alter
#DML : Select, Insert, delete, update
```

In [69]:
```
con.close()
```

In [80]:
```
#Data cleaning
```

In [77]:
```
frame.columns
```

Out[77]:
```
Index(['Degree', 'Student Name', 'Mid', 'Quiz 1', 'Quiz 2', 'Best of Quizze
s',
       'Assignment 1', 'Assignment 2', 'Best of Assignments',
       'Total Sessional (50)', 'Final (50)', 'Total (100)', 'Grade'],
      dtype='object')
```

In [79]:
```
frame['Quiz 1'].notnull()
```

Out[79]:
```
Student Code
022-14-19987      True
022-14-110233    False
022-14-110585     True
022-14-19718      True
022-14-110648     True
022-14-110232     True
022-14-110588     True
022-14-110388     True
022-14-110601     True
022-14-110599     True
022-14-110214     True
022-14-110591     True
022-15-110994     True
022-14-110600     True
022-14-110389     True
022-14-19916     False
022-14-110596     True
022-14-110396     True
022-14-110223     True
022-14-110222     True
022-14-110412     True
022-14-110593     True
022-14-110398     True
022-14-19983      True
022-14-110215     True
022-14-110105     True
022-14-110370     True
022-14-110452     True
022-14-110387     True
022-14-110217     True
022-14-110401     True
022-14-110231     True
022-14-110035    False
022-14-19919      True
022-14-110413     True
022-14-19923      True
022-14-110582     True
022-14-110230     True
022-14-110229     True
022-14-110107     True
022-14-110584     True
022-14-110225     True
022-14-110587     True
022-14-110451     True
022-14-110589     True
022-14-110400     True
022-14-19911      True
022-14-110219     True
Name: Quiz 1, dtype: bool
```

In [81]:
```
frame['Quiz 1'].isnull()
```

Out[81]:
```
Student Code
022-14-19987     False
022-14-110233     True
022-14-110585    False
022-14-19718     False
022-14-110648    False
022-14-110232    False
022-14-110588    False
022-14-110388    False
022-14-110601    False
022-14-110599    False
022-14-110214    False
022-14-110591    False
022-15-110994    False
022-14-110600    False
022-14-110389    False
022-14-19916      True
022-14-110596    False
022-14-110396    False
022-14-110223    False
022-14-110222    False
022-14-110412    False
022-14-110593    False
022-14-110398    False
022-14-19983     False
022-14-110215    False
022-14-110105    False
022-14-110370    False
022-14-110452    False
022-14-110387    False
022-14-110217    False
022-14-110401    False
022-14-110231    False
022-14-110035     True
022-14-19919     False
022-14-110413    False
022-14-19923     False
022-14-110582    False
022-14-110230    False
022-14-110229    False
022-14-110107    False
022-14-110584    False
022-14-110225    False
022-14-110587    False
022-14-110451    False
022-14-110589    False
022-14-110400    False
022-14-19911     False
022-14-110219    False
Name: Quiz 1, dtype: bool
```

In [86]:
```
pd.set_option('display.max_rows', None)
```

In [87]:
```
frame
```

Out[87]:

Student Code	Degree	Student Name	Mid	Quiz 1	Quiz 2	Best of Quizzes	Assignment 1	Assignment 2	Best o Assignment
022-14-19987	BS(CS)	Abdul Basit	28	8.0	3.0	8	7.000000	9.0	
022-14-110233	BS(CS)	Adeel Ahmed	17	NaN	5.0	5	8.000000	10.0	1
022-14-110585	BS(CS)	Afrah Zareen	18	5.0	2.0	5	8.000000	10.0	1
022-14-19718	BS(CS)	Ahmed Ali Raza	14	7.0	2.0	7	7.755556	2.0	
022-14-110648	BS(CS)	Ahsan Ali Vohra	27	7.0	6.0	7	7.000000	9.0	
022-14-	BS(CS)	Ameer	25	9.0	6.0	9	8.000000	10.0	1

In [89]:
```
print(frame.count())
print(frame.dropna().count())
```

```
Degree                  48
Student Name            48
Mid                     48
Quiz 1                  45
Quiz 2                  45
Best of Quizzes         48
Assignment 1            48
Assignment 2            36
Best of Assignments     48
Total Sessional (50)    48
Final (50)              47
Total (100)             48
Grade                   48
dtype: int64
Degree                  32
Student Name            32
Mid                     32
Quiz 1                  32
Quiz 2                  32
Best of Quizzes         32
Assignment 1            32
Assignment 2            32
Best of Assignments     32
Total Sessional (50)    32
Final (50)              32
Total (100)             32
Grade                   32
dtype: int64
```

In [90]:
```
print(frame.count())
print(frame.dropna(how='all').count())
```

```
Degree                  48
Student Name            48
Mid                     48
Quiz 1                  45
Quiz 2                  45
Best of Quizzes         48
Assignment 1            48
Assignment 2            36
Best of Assignments     48
Total Sessional (50)    48
Final (50)              47
Total (100)             48
Grade                   48
dtype: int64
Degree                  48
Student Name            48
Mid                     48
Quiz 1                  45
Quiz 2                  45
Best of Quizzes         48
Assignment 1            48
Assignment 2            36
Best of Assignments     48
Total Sessional (50)    48
Final (50)              47
Total (100)             48
Grade                   48
dtype: int64
```

In [125]:
```
df = pd.DataFrame({'name':['ali','asif','rashid',None],'age':[1,None,3,None],
```

In [126]:
```
df
```

Out[126]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	3.0	3.0
3	None	NaN	25.0

In [127]:
```
df.dropna(how='all')
```

Out[127]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	3.0	3.0
3	None	NaN	25.0

In [128]:
```
df.dropna(thresh=2) #atleast two columns should have the data
```

Out[128]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	3.0	3.0

In [129]:
```
df.dropna(thresh=3,axis=1) #atleast three rows have data to survive a column
```

Out[129]:

	name	marks
0	ali	NaN
1	asif	12.0
2	rashid	3.0
3	None	25.0

In [130]:
```
df
```

Out[130]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	3.0	3.0
3	None	NaN	25.0

In [136]:
```
df[df['name'].notnull()]
```

Out[136]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	3.0	3.0

In [137]:
```
df
```

Out[137]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	3.0	3.0
3	None	NaN	25.0

In [138]:
```
df.fillna({  #specifying different filled values for each column
    'name':'No Name',
    'age':18,
    'marks':0
})
```

Out[138]:

	name	age	marks
0	ali	1.0	0.0
1	asif	18.0	12.0
2	rashid	3.0	3.0
3	No Name	18.0	25.0

In [139]:
```
df.fillna(0) #Filled all missing values with 0
```

Out[139]:

	name	age	marks
0	ali	1.0	0.0
1	asif	0.0	12.0
2	rashid	3.0	3.0
3	0	0.0	25.0

In [140]:
```
df['age'].fillna(df['age'].mean()) #Filled with mean vale
```

Out[140]:
```
0    1.0
1    2.0
2    3.0
3    2.0
Name: age, dtype: float64
```

In [141]:
```
df
```

Out[141]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	3.0	3.0
3	None	NaN	25.0

In [142]:
```
df.fillna(method='ffill')
```

Out[142]:

	name	age	marks
0	ali	1.0	NaN
1	asif	1.0	12.0
2	rashid	3.0	3.0
3	rashid	3.0	25.0

In [143]:
```
df = pd.DataFrame({'name':['ali','asif','rashid',None],'age':[1,None,None,None
```

In [144]:
```
df
```

Out[144]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	NaN	3.0
3	None	NaN	25.0

In [145]:
```
df.fillna(method='ffill',limit=2)
```

Out[145]:

	name	age	marks
0	ali	1.0	NaN
1	asif	1.0	12.0
2	rashid	1.0	3.0
3	rashid	NaN	25.0

In [146]:
```
df = pd.DataFrame({'name':['ali','asif','rashid','rashid'],'age':[1,None,None,
```

In [147]:
```
df
```

Out[147]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	NaN	3.0
3	rashid	NaN	3.0

In [149]:
```
df.drop_duplicates()
```

Out[149]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	NaN	3.0

In [150]:
```
df = pd.DataFrame({'name':['ali','asif','rashid','rashid'],'age':[1,None,None,
```

In [151]:
```
df
```

Out[151]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	NaN	3.0
3	rashid	NaN	31.0

In [152]:
```
df.drop_duplicates(['name'])
```

Out[152]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	NaN	3.0

In [153]:
```
df.drop_duplicates(['name'],keep='last')
```

Out[153]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
3	rashid	NaN	31.0

In [154]:
```
father_name={
    'ali':'Hamid',
    'asif':'Asghar',
    'rashid':'Naeem'
}
```

In [155]:
```
df
```

Out[155]:

	name	age	marks
0	ali	1.0	NaN
1	asif	NaN	12.0
2	rashid	NaN	3.0
3	rashid	NaN	31.0

In [157]:
```
df['Father Name'] = df['name'].map(father_name)
```

In [158]:
```
df
```

Out[158]:

	name	age	marks	Father Name
0	ali	1.0	NaN	Hamid
1	asif	NaN	12.0	Asghar
2	rashid	NaN	3.0	Naeem
3	rashid	NaN	31.0	Naeem

In [162]:
```
df['Full Name'] = df['name'].map(lambda name: name+' ' +father_name[name])
```

In [163]:
```
df
```

Out[163]:

	name	age	marks	Father Name	Full Name
0	ali	1.0	NaN	Hamid	ali Hamid
1	asif	NaN	12.0	Asghar	asif Asghar
2	rashid	NaN	3.0	Naeem	rashid Naeem
3	rashid	NaN	31.0	Naeem	rashid Naeem

In [ ]:
```

```

In [1]:
```
import pandas as pd
```

```
D:\Anaconda3\lib\site-packages\pandas\compat\_optional.py:138: UserWarning: P
andas requires version '2.7.0' or newer of 'numexpr' (version '2.6.9' current
ly installed).
  warnings.warn(msg, UserWarning)
```

In [2]:
```
df = pd.DataFrame(
{
    'name':['Ali','Zahid','Ubaid'],
    'age':[18,20,18]
}
)
```

In [4]:
```
df['age']=df['age'].replace({18:25 })
```

In [5]:
```
df
```

Out[5]:

	name	age
0	Ali	25
1	Zahid	20
2	Ubaid	25

In [17]:
```
df.iloc[0,1]=27
```

In [18]:
```
df
```

Out[18]:

	name	age
0	Ali	27
1	Zahid	20
2	Ubaid	25

In [19]:
```
df['name'].replace(['Ali','Zahid'],['No name', ' '])
```

Out[19]:
```
0    No name
1
2      Ubaid
Name: name, dtype: object
```

In [20]:
```
df['name'].replace({'Ali':'No name','Zahid': ' '})
```

Out[20]:
```
0    No name
1
2      Ubaid
Name: name, dtype: object
```

In [21]:
```
df['name'].replace(['Ali','Zahid'],'No name')
```

Out[21]:
```
0    No name
1    No name
2      Ubaid
Name: name, dtype: object
```

In [22]:
```
df
```

Out[22]:

	name	age
0	Ali	27
1	Zahid	20
2	Ubaid	25

In [23]:
```
#Map
```

In [26]:
```
df['age'].map(lambda age: 'Older' if age>25  else 'Younger')
```

Out[26]:
```
0      Older
1    Younger
2    Younger
Name: age, dtype: object
```

In [27]:
```
#Assignment: What is the difference between map and apply
```

In [28]:
```
import numpy as np
```

In [29]:
```
df = pd.DataFrame(
    np.arange(0,12).reshape(3,4),
    index=['A','B','C'],
    columns = ['C1','C2','C3','C4']
)
```

In [34]:
```
display(df)
print(df.loc['A'])
print(df.iloc[1])
```

	C1	C2	C3	C4
A	0	1	2	3
B	4	5	6	7
C	8	9	10	11

```
C1    0
C2    1
C3    2
C4    3
Name: A, dtype: int32
C1    4
C2    5
C3    6
C4    7
Name: B, dtype: int32
```

In [42]:
```
#We can also apply map to index. Map can be used to convert a specific column
```

In [35]:
```
df.index
```

Out[35]:
```
Index(['A', 'B', 'C'], dtype='object')
```

In [36]:
```
index_to_country = {
    'A': 'Australia',
    'B': 'Brazil',
    'C': 'China'
}
```

In [39]:
```
df.index = df.index.map(index_to_country)
```

In [40]:
```
df
```

Out[40]:

	C1	C2	C3	C4
Australia	0	1	2	3
Brazil	4	5	6	7
China	8	9	10	11

In [41]:
```
df.loc['Australia']
```

Out[41]:
```
C1    0
C2    1
C3    2
C4    3
Name: Australia, dtype: int32
```

In [43]:
```
#Other index methods
```

In [44]:
```
df.rename(index=str.upper, columns=str.lower) #Changing columns / index to upp
```

Out[44]:

	c1	c2	c3	c4
AUSTRALIA	0	1	2	3
BRAZIL	4	5	6	7
CHINA	8	9	10	11

In [46]:
```
df.rename(index={'China': 'Coulambia'},
          columns={'C1': 'C11'}) #Changing specific column and index
```

Out[46]:

	C11	C2	C3	C4
Australia	0	1	2	3
Brazil	4	5	6	7
Coulambia	8	9	10	11

In [47]:
```
df.rename(index={'China': 'Columabia'}, inplace=True) #doing inplace
```

In [48]:
```
df
```

Out[48]:

	C1	C2	C3	C4
Australia	0	1	2	3
Brazil	4	5	6	7
Columabia	8	9	10	11

In [54]:
```
#Discretization and binning: To place items into respective buckets/ bins if t
```

In [ ]:
```
88+ -> A
75-87 -> B
60-74 -> C
<60 -> F
```

In [58]:
```
marks = [90, 22, 25, 67, 81, 83, 87]
bins = [0,60,75, 88,100]
cats = pd.cut(marks, bins)
```

In [59]:
```
cats.codes
```

Out[59]:
```
array([3, 0, 0, 1, 2, 2, 2], dtype=int8)
```

In [60]:
```
cats.categories
```

Out[60]:
```
IntervalIndex([(0, 60], (60, 75], (75, 88], (88, 100]], dtype='interval[int6
4, right]')
```

In [ ]:
```
(,) - closed interval, exclusive
[ , ] - open interval, inclusive
```

In [62]:
```
pd.value_counts(cats)
```

Out[62]:
```
(75, 88]     3
(0, 60]      2
(60, 75]     1
(88, 100]    1
dtype: int64
```

In [63]:
```
cats = pd.cut(marks, bins,labels=['F','C','B','A'])
```

In [68]:
```
pd.value_counts(cats)
```

Out[68]:
```
B    3
F    2
C    1
A    1
dtype: int64
```

In [69]:
```
print(cats.categories)
print(cats.codes)
```

```
Index(['F', 'C', 'B', 'A'], dtype='object')
[3 0 0 1 2 2 2]
```

In [70]:
```
marks
```

Out[70]:
```
[90, 22, 25, 67, 81, 83, 87]
```

In [72]:
```
#descriptive statistics, outliers removal
```

In [71]:
```
df.describe()
```

Out[71]:

	C1	C2	C3	C4
count	3.0	3.0	3.0	3.0
mean	4.0	5.0	6.0	7.0
std	4.0	4.0	4.0	4.0
min	0.0	1.0	2.0	3.0
25%	2.0	3.0	4.0	5.0
50%	4.0	5.0	6.0	7.0
75%	6.0	7.0	8.0	9.0
max	8.0	9.0	10.0	11.0

In [73]:
```
df[np.abs(df['C2']) > 5]
```

Out[73]:

	C1	C2	C3	C4
Columabia	8	9	10	11

In [75]:
```
#Capping data outside the range
#Capping putting the value on the edge values
df[np.abs(df['C2']) > 5] = np.sign(df) * 5   # -7 -> -5, 8->5
```

In [76]:
```
df
```

Out[76]:

	C1	C2	C3	C4
Australia	0	1	2	3
Brazil	4	5	6	7
Columabia	5	5	5	5

In [77]:
```
#Merging two data frames: It's similar to SQL joins. We join two data frame bas
```

In [81]:
```
df1 = pd.DataFrame(
    {
        'Id':[1,2,3],
        'Names':['John','Peter','Silbert']
    }
)
```

In [80]:
```
df2 = pd.DataFrame(
    {
        'Id':[1,2,3],
        'Salary':['150','170','190']
    }
)
```

In [83]:
```
display(df1)
display(df2)
```

	Id	Names
0	1	John
1	2	Peter
2	3	Silbert

	Id	Salary
0	1	150
1	2	170
2	3	190

In [82]:
```
pd.merge(df1,df2)
```

Out[82]:

	Id	Names	Salary
0	1	John	150
1	2	Peter	170
2	3	Silbert	190

In [84]:
```
df2 = pd.DataFrame(
    {
        'Emp_Id':[1,2,3],
        'Salary':['150','170','190']
    }
)
```

In [85]:
```
#When the common column name doesn't match
```

In [86]:
```
pd.merge(df1, df2, left_on='Id', right_on='Emp_Id')
```

Out[86]:

	Id	Names	Emp_Id	Salary
0	1	John	1	150
1	2	Peter	2	170
2	3	Silbert	3	190

In [87]:
```
#Different types of joins
```

In [90]:
```
df2 = pd.DataFrame(
    {
        'Id':[1,2,3,4],
        'Salary':['150','170','190','90']
    }
)
```

In [91]:
```
pd.merge(df1,df2,how='right')
```

Out[91]:

	Id	Names	Salary
0	1	John	150
1	2	Peter	170
2	3	Silbert	190
3	4	NaN	90

In [92]:
```
#Merging based on indexes
```

In [101]:
```
df2 = pd.DataFrame(
    {
        'Salary':['150','170','190','90']
    },
            index=[1,2,3,4],
)
```

In [97]:
```
display(df1)
display(df2)
```

	Id	Names
0	1	John
1	2	Peter
2	3	Silbert

	Salary
1	150
2	170
3	190
4	90

In [96]:
```
pd.merge(df1,df2,left_on='Id',right_index=True)
```

Out[96]:

	Id	Names	Salary
0	1	John	150
1	2	Peter	170
2	3	Silbert	190

In [102]:
```
#Concatenating two dataframes
```

In [103]:
```
df1
```

Out[103]:

	Id	Names
0	1	John
1	2	Peter
2	3	Silbert

In [104]:
```
df2 = pd.DataFrame(
    {
        'Id':[4,5],
        'Names':['Ryo','Pawan']
    }
)
```

In [107]:
```
pd.concat([df1,df2]) #Row-wise concatenation
```

Out[107]:

	Id	Names
0	1	John
1	2	Peter
2	3	Silbert
0	4	Ryo
1	5	Pawan

In [108]:
```
pd.concat([df1,df2],axis=1) #Column-wise concatenation
```

Out[108]:

	Id	Names	Id	Names
0	1	John	4.0	Ryo
1	2	Peter	5.0	Pawan
2	3	Silbert	NaN	NaN

In [111]:
```
#combine first
```

In [125]:
```
df1 = pd.DataFrame(
    {
        'Id':[1,2,3],
        'Names':['John','Peter','Silbert'],
        'Ages':[np.nan,'40',np.nan]
    }
)
```

In [126]:
```
df2 = pd.DataFrame(
    {
        'Id':[1,2,3],
        'Ages':['23',np.nan,'40']
    }
)
```

In [127]:
```
display(df1)
display(df2)
```

	Id	Names	Ages
0	1	John	NaN
1	2	Peter	40
2	3	Silbert	NaN

	Id	Ages
0	1	23
1	2	NaN
2	3	40

In [128]:
```
df1.combine_first(df2)
```

Out[128]:

	Ages	Id	Names
0	23	1	John
1	40	2	Peter
2	40	3	Silbert

In [116]:
```
#Assignment: Combine First example from book
```

In [129]:
```
#Matplotlib: A library for data visualization. You can draw bar graphs, pie ch
```

In [130]:
```
import matplotlib.pyplot as plt
```

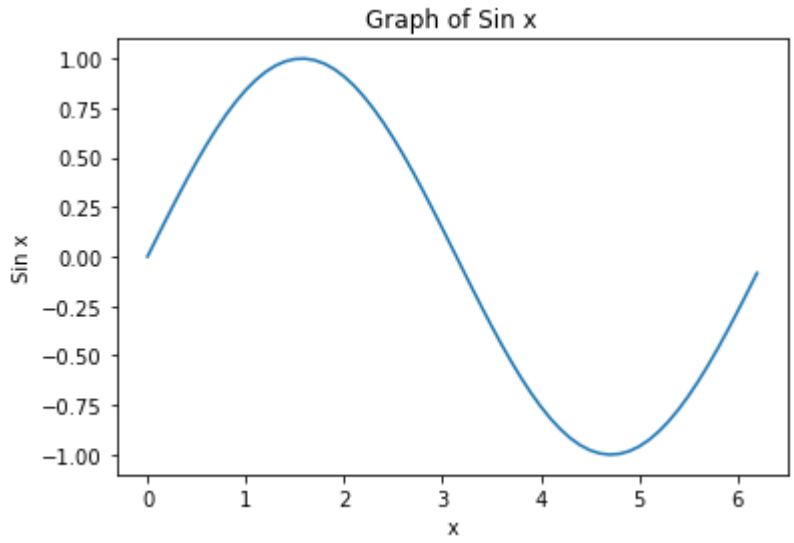
In [131]:
```
%matplotlib inline
```

In [135]:
```
x=np.arange(0,2*3.14,0.1)
```

In [136]:
```
y=np.sin(x)
```

In [140]:
```
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('Sin x')
plt.title('Graph of Sin x')
```

Out[140]:
```
Text(0.5, 1.0, 'Graph of Sin x')
```

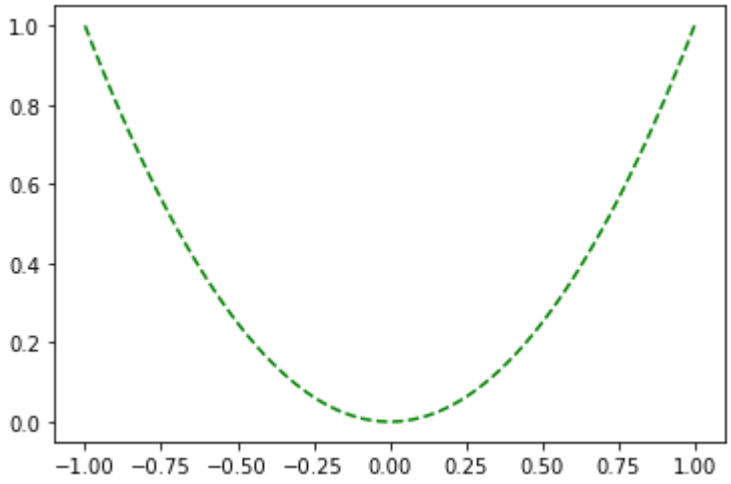


In [145]:
```
x = np.linspace(-1,1,100)
```

In [146]:
```
y=x**2
```

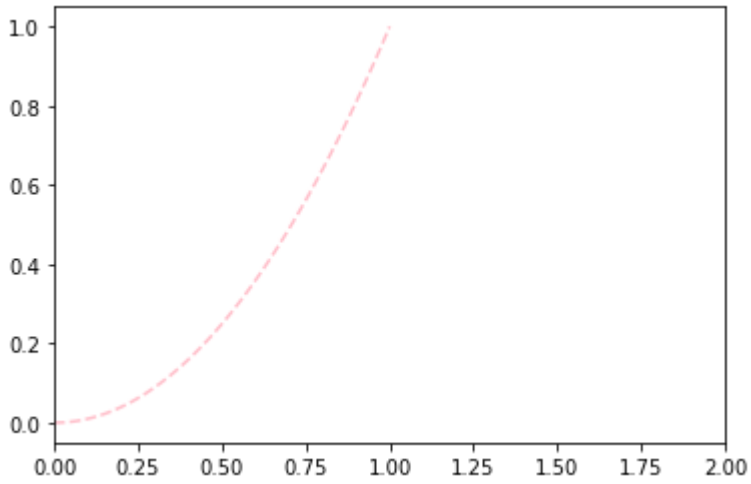
In [148]:
```
plt.plot(x,y,'g--')
```

Out[148]:
```
[<matplotlib.lines.Line2D at 0x1f658f53908>]
```



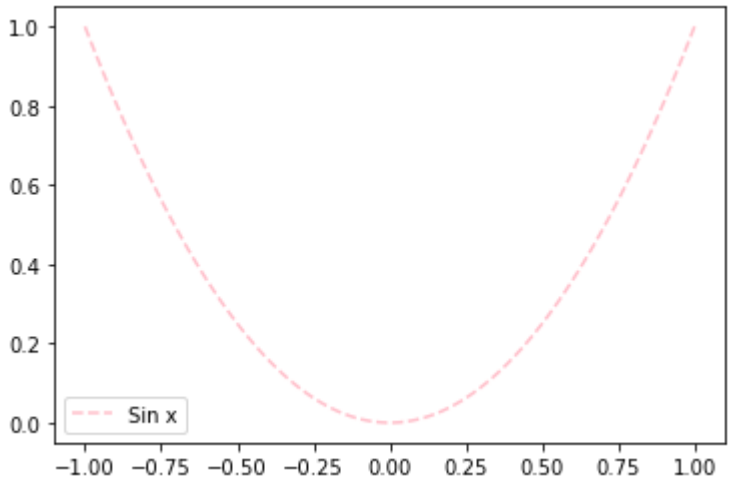
In [151]:
```
plt.plot(x,y,color='pink',linestyle='dashed')
plt.xlim(0,2)
```

Out[151]:
```
(0, 2)
```



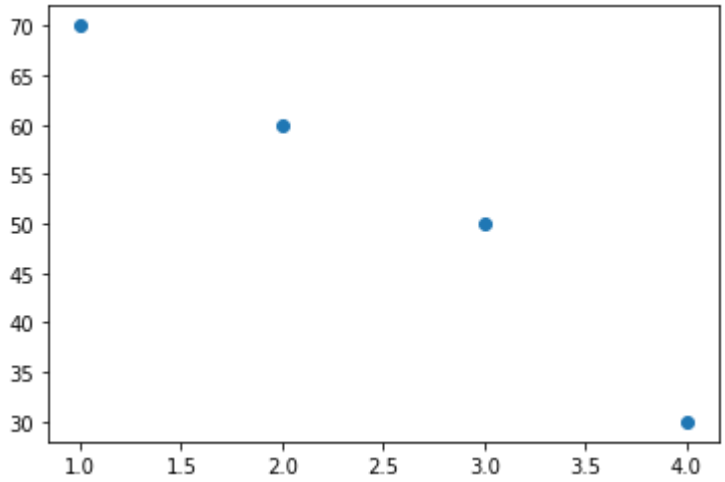
In [152]:
```
plt.plot(x,y,color='pink',linestyle='dashed',label='Sin x')
plt.legend()
```

Out[152]: <matplotlib.legend.Legend at 0x1f6590b01d0>



In [155]:
```
#Scatter
plt.scatter([1,2,3,4],[70,60,50,30])
```

Out[155]: <matplotlib.collections.PathCollection at 0x1f66135ac88>



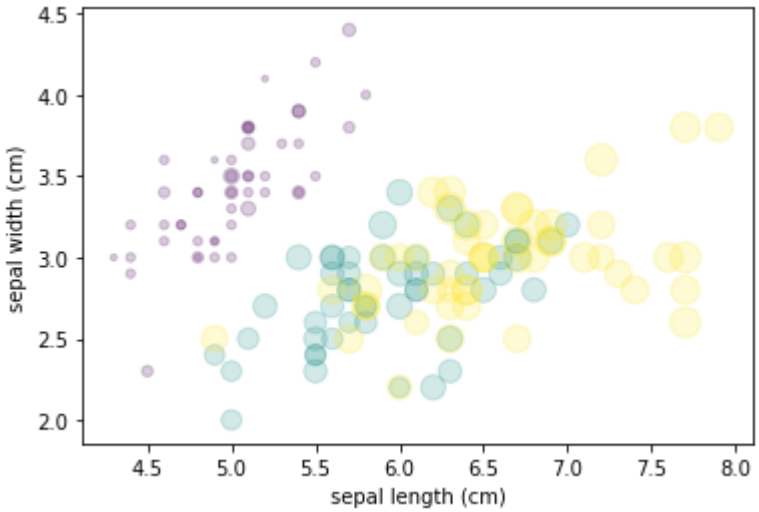
In [156]:
```
from sklearn.datasets import load_iris
iris = load_iris()  #Load the Iris data
iris #Show the iris data: data/ features and target
```

Out[156]:
```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
        [4.8, 3.4, 1.6, 0.2],
        [4.8, 3. , 1.4, 0.1],
        [4.3, 3. , 1.1, 0.1],
        [5.8, 4. , 1.2, 0.2],
        [5.7, 4.4, 1.5, 0.4],
        [5.4, 3.9, 1.3, 0.4],
        [5.1, 3.5, 1.4, 0.3],
        [5.7, 3.8, 1.7, 0.3],
```

In [161]:
```
features = iris.data.T #Transpose the data
plt.scatter(features[0], features[1], alpha=0.2, s=100*features[3], c=iris.tar
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])

#Position - feature[0], feature[1]
#Size - Feature[3]
#Color - Target (Virginica,Setosa, ...)
#alpha - transparency
```

Out[161]:
```
Text(0, 0.5, 'sepal width (cm)')
```



In [ ]:
```
```