Q1.Create two numpy arrays. The first array, called arr1 should be of one dimension and consist of six elements of float type. The second array, called arr2 should be of two dimensions and each sub-array should consist of 5 elements of string type. Display these two numpy arrays.

In [1]:
```
import numpy as np
arr1 = np.array([1.1, 2.5, 3.8, 4.4, 5.2,6.8])
arr2=np.array([['apple','banana','orange','cherry','mango'], ['papaya','kiwi','strawberry','grapes','plums']])
print(arr1)
print(arr2)
```

```
[1.1 2.5 3.8 4.4 5.2 6.8]
[['apple' 'banana' 'orange' 'cherry' 'mango']
 ['papaya' 'kiwi' 'strawberry' 'grapes' 'plums']]
```

Q2.For the arrays arr1 and arr2 created in Q.No.1, display the following properties: data type and dimensions. Reshape arr1 into a two dimensional array of 2 rows and three columns, and display the reshaped array. Similarly, reshape arr2 into a single dimensional array and display the reshaped array.

In [2]:
```
print("type is: ",type(arr1))
print("dtype is: ",arr1.dtype)
print("type is: ",type(arr2))
print("dtype is: ",arr2.dtype)
arr1=arr1.reshape(2,3)
print(arr1)
arr2=arr2.flatten()
print(arr2)
```

```
type is:  <class 'numpy.ndarray'>
dtype is:  float64
type is:  <class 'numpy.ndarray'>
dtype is:  <U10
[[1.1 2.5 3.8]
 [4.4 5.2 6.8]]
['apple' 'banana' 'orange' 'cherry' 'mango' 'papaya' 'kiwi' 'strawberry'
 'grapes' 'plums']
```

Q3.Create a two dimensional array called arr3, of 3 rows and 4 columns which is initialized to zeroes. Create another single dimensional array called arr4, with 6 elements and which is initialized to ones. The type of arr4 is int.

In [4]:
```
import numpy as np
arr3=np.zeros([3,4],dtype=float)
arr4=np.ones(6,dtype=int)
print(arr3)
print(arr4)
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
[1 1 1 1 1 1]
```

Q4.Create a list called list1 with the elements 10,20,30,40,50. Now, create a numpy array called arr5 from list1, using the method asarray. The data type of arr5 is float. Similarly, create another array called arr6, from list1, using the method fromiter, whose data type is int. Display arr5 and arr6.

In [5]:
```
list1 = [10,20,30,40,50]
print(list1)
arr5=np.array(list1,dtype=float)
print(arr5)
arr6=np.fromiter(list1,dtype=int,count=-1)
print(arr6)
```

```
[10, 20, 30, 40, 50]
[10. 20. 30. 40. 50.]
[10 20 30 40 50]
```

Q5.Create a numpy array called arr7 using the method arange. The starting value of this array is 12 and ending value is 99, with a step size of 3. Use the linspace method to create another array called arr8, where the starting value is 12 and ending value is 99 and which contains 50 evenly spaced elements.

In [6]:
```
arr7=np.arange(12,100,3)
print(arr7)
arr8=np.linspace(12,100,num=50,endpoint=True)
print(arr8)
```

```
[12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81
 84 87 90 93 96 99]
[ 12.          13.79591837  15.59183673  17.3877551   19.18367347
  20.97959184  22.7755102   24.57142857  26.36734694  28.16326531
  29.95918367  31.75510204  33.55102041  35.34693878  37.14285714
  38.93877551  40.73469388  42.53061224  44.32653061  46.12244898
  47.91836735  49.71428571  51.51020408  53.30612245  55.10204082
  56.89795918  58.69387755  60.48979592  62.28571429  64.08163265
  65.87755102  67.67346939  69.46938776  71.26530612  73.06122449
  74.85714286  76.65306122  78.44897959  80.24489796  82.04081633
  83.83673469  85.63265306  87.42857143  89.2244898   91.02040816
  92.81632653  94.6122449   96.40816327  98.20408163 100.        ]
```

Q6.In the array arr2 defined in Q.No. 1, what is the output of the following statements: print(arr2), print(arr2[0:1]), print(arr2[1][3]), print (arr2[1])

In [7]:
```
import numpy as np
arr2=np.array([['apple','banana','orange','cherry','mango'], ['papaya','kiwi','strawberry','grapes','plums']])
print(arr2)
print(arr2[0:1])
print(arr2[1][3])
print(arr2[1])
```

```
[['apple' 'banana' 'orange' 'cherry' 'mango']
 ['papaya' 'kiwi' 'strawberry' 'grapes' 'plums']]
[['apple' 'banana' 'orange' 'cherry' 'mango']]
grapes
['papaya' 'kiwi' 'strawberry' 'grapes' 'plums']
```

Q7.Create a two dimensional array called arr9 with the following elements: [101, 322, 768], [233, 124, 390], [534, 122, 216]. From arr9, create an array called arr10 which contains the elements 101, 322, 390, 122 using advanced integer indexing. Create another array called arr11 which contains elements of first and second rows, and second and third columns using slicing operations.

In [13]:
```
arr9 = np.array([[101,322,768],[233,124,390],[534,122,216]],dtype=int)
arr10 = arr9[[0,0,1,2],[0,1,2,1]]
arr11 = arr9[0:2][1:3]
print('array arr9 is :\n',arr9)
print('\n')
print('Advanced Integer Indexing :',arr10)
print('\n')
print('Slicing Operation :',arr11)
```

```
array arr9 is :
 [[101 322 768]
 [233 124 390]
 [534 122 216]]


Advanced Integer Indexing : [101 322 390 122]


Slicing Operation : [[233 124 390]]
```

Q8.From array arr9 created in Q.No. 7, find all elements greater than 200 using boolean indexing, find all non-NaN elements, and find all non-float elements. Find the result of adding arrays arr9 and another array of 3 elements, using the concept of broadcasting.

In [17]:
```
arr12 = arr9[arr9 > 200]
print('In arr9, Elements are greater than 200 :',arr12)
print('\n')
arr13 = arr9[~np.isnan(arr9)]
print('Non-NaN Elements in arr9 :',arr13)
print('\n')
arr14 = arr9[arr9.dtype != 'float']
print('Non-Float Elements in arr9 :',arr14)
print('\n')
arr15 = np.array([[arr12],[arr13],[arr14]])
arr16 = arr9 + arr15
print('The Broadcasting Method :',arr16)
```

```
In arr9, Elements are greater than 200 : [322 768 233 390 534 216]


Non-NaN Elements in arr9 : [101 322 768 233 124 390 534 122 216]


Non-Float Elements in arr9 : [[[101 322 768]
  [233 124 390]
  [534 122 216]]]


The Broadcasting Method : [[array([423, 869, 334, 491, 635, 317])
  array([ 644, 1090,  555,  712,  856,  538])
  array([1090, 1536, 1001, 1158, 1302,  984])]
 [array([ 334,  555, 1001,  466,  357,  623,  767,  355,  449])
  array([225, 446, 892, 357, 248, 514, 658, 246, 340])
  array([ 491,  712, 1158,  623,  514,  780,  924,  512,  606])]
 [array([[[ 635,  856, 1302],
          [ 767,  658,  924],
          [1068,  656,  750]]]) array([[[223, 444, 890],
                                        [355, 246, 512],
                                        [656, 244, 338]]])

  array([[[317, 538, 984],
          [449, 340, 606],
          [750, 338, 432]]])]]
```

```
C:\Users\91830\AppData\Local\Temp\ipykernel_30108\1271294.py:11: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
  arr15 = np.array([[arr12],[arr13],[arr14]])
```

Q9.For the array arr9, find its transpose. Find the result of joining the arrays arr1 and arr4 along axis 0.

In [21]:
```
print('The Transpose of arr9 is given below :')
print(arr9.T)
print('The joining of two arrays are arr1 and arr4 given below :')
x=np.concatenate((arr1,arr4),axis=0)
print(x)
```

```
The Transpose of arr9 is given below :
[[101 233 534]
 [322 124 122]
 [768 390 216]]
The joining of two arrays are arr1 and arr4 given below :
[1.1 2.5 3.8 4.4 5.2 6.8 1.  1.  1.  1.  1.  1. ]
```

Q10.Using any two 2-dimensional arrays, demonstrate the working of the methods, hstack, vstack, hsplit and vsplit.

In [35]:
```
import numpy as np
arr1=np.array([[1000,5000],[10000,40000]])
a=np.hstack(arr1)
b=np.vstack(arr1)
c=np.hsplit(arr1,1)
d=np.vsplit(arr1,1)
print(a)
print(b)
print(c)
print(d)
```

```
[ 1000  5000 10000 40000]
[[ 1000  5000]
 [10000 40000]]
[array([[ 1000,  5000],
       [10000, 40000]])]
[array([[ 1000,  5000],
       [10000, 40000]])]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: