**1.** Write a Python program to print the NumPy version in your system. 

**2.** Write a Python program to convert a list of numeric value into a one-dimensional NumPy array.    
Expected Output:  
Original List: [12.23, 13.32, 100, 36.32]   
One-dimensional numpy array: [ 12.23 13.32 100. 36.32] 

**3.** Create a 3x3 matrix with values ranging from 2 to 10.    
Expected Output:  
[[ 2 3 4]   
[ 5 6 7]   
[ 8 9 10]] 

**4.**Write a Python program to create a null vector of size 10 and update sixth value to 11.   
[ 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]   
Update sixth value to 11   
[ 0. 0. 0. 0. 0. 0. 11. 0. 0. 0.]

**5.** Write a Python program to create a array with values ranging from 12 to 38.   
Expected Output:  
[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] 

**6.** Write a Python program to reverse an array (first element becomes last).    
Original array:   
[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]   
Reverse array:   
[37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12] 

**7.** Write a Python program to an array converted to a float type.    
Sample output:  
Original array   
[1, 2, 3, 4]   
Array converted to a float type:   
[ 1. 2. 3. 4.] 

**8.**Write a Python program to create a 2d array with 1 on the border and 0 inside.    
Expected Output:  
Original array:   
[[ 1. 1. 1. 1. 1.]   
[ 1. 1. 1. 1. 1.]   
[ 1. 1. 1. 1. 1.]   
[ 1. 1. 1. 1. 1.]   
[ 1. 1. 1. 1. 1.]]   
1 on the border and 0 inside in the array   
[[ 1. 1. 1. 1. 1.]   
[ 1. 0. 0. 0. 1.]   
[ 1. 0. 0. 0. 1.]   
[ 1. 0. 0. 0. 1.]   
[ 1. 1. 1. 1. 1.]]

**9.**Write a Python program to add a border (filled with 0's) around an existing array.   
Expected Output:  
Original array:   
[[ 1. 1. 1.]   
[ 1. 1. 1.]   
[ 1. 1. 1.]]   
1 on the border and 0 inside in the array   
[[ 0. 0. 0. 0. 0.]   
[ 0. 1. 1. 1. 0.]   
[ 0. 1. 1. 1. 0.]   
[ 0. 1. 1. 1. 0.]   
[ 0. 0. 0. 0. 0.]]

**10.**Write a Python program to create a 8x8 matrix and fill it with a checkerboard pattern.   
Checkerboard pattern:  
[[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]   
[1 0 1 0 1 0 1 0]]

**11.**Write a Python program to convert a list and tuple into arrays.   
List to array:  
[1 2 3 4 5 6 7 8]   
Tuple to array:  
[[8 4 6]   
[1 2 3]]

**12.**Write a Python program to append values to the end of an array.   
Expected Output:  
Original array:  
[10, 20, 30]   
After append values to the end of the array:  
[10 20 30 40 50 60 70 80 90]

**13.**Write a Python program to create an empty and a full array.   
Expected Output:  
[ 6.93270651e-310 1.59262180e-316 6.93270559e-310 6.93270665e-310]  
[ 6.93270667e-310 6.93270671e-310 6.93270668e-310 6.93270483e-310]   
[ 6.93270668e-310 6.93270671e-310 6.93270370e-310 6.93270488e-310]]   
[[6 6 6]   
[6 6 6]   
[6 6 6]] 

**14.**Write a Python program to convert the values of Centigrade degrees into Fahrenheit degrees. Centigrade values are stored into a NumPy array.   
Sample Array [0, 12, 45.21 ,34, 99.91]  
Expected Output:  
Values in Fahrenheit degrees:  
[ 0. 12. 45.21 34. 99.91]   
Values in Centigrade degrees:   
[-17.77777778 -11.11111111 7.33888889 1.11111111 37.72777778] 

**15.** Write a Python program to find the real and imaginary parts of an array of complex numbers.   
Expected Output:  
Original array [ 1.00000000+0.j 0.70710678+0.70710678j]   
Real part of the array:   
[ 1. 0.70710678]   
Imaginary part of the array:  
[ 0. 0.70710678] 

**16.** Write a Python program to find the number of elements of an array, length of one array element in bytes and total bytes consumed by the elements.   
Expected Output:  
Size of the array: 3  
Length of one array element in bytes: 8   
Total bytes consumed by the elements of the array: 24 

**17.** Write a Python program to test whether each element of a 1-D array is also present in a second array.   
Expected Output:  
Array1: [ 0 10 20 40 60]  
Array2: [0, 40]   
Compare each element of array1 and array2   
[ True False False True False] 

**18.** Write a Python program to find common values between two arrays.   
Expected Output:  
Array1: [ 0 10 20 40 60]   
Array2: [10, 30, 40]  
Common values between two arrays:  
[10 40] 

**19.** Write a Python program to get the unique elements of an array.   
Expected Output:  
Original array:  
[10 10 20 20 30 30]   
Unique elements of the above array:  
[10 20 30]   
Original array:  
[[1 1]   
[2 3]]  
Unique elements of the above array:  
[1 2 3] 

**20.** Write a Python program to find the set difference of two arrays. The set difference will return the sorted, unique values in array1 that are not in array2.   
Expected Output:  
Array1: [ 0 10 20 40 60 80]  
Array2: [10, 30, 40, 50, 70, 90]   
Set difference between two arrays:  
[ 0 20 60 80]

**21.** Write a Python program to find the set exclusive-or of two arrays. Set exclusive-or will return the sorted, unique values that are in only one (not both) of the input arrays.   
Array1: [ 0 10 20 40 60 80]   
Array2: [10, 30, 40, 50, 70]   
Unique values that are in only one (not both) of the input arrays:   
[ 0 20 30 50 60 70 80]

**22.** Write a Python program to find the union of two arrays. Union will return the unique, sorted array of values that are in either of the two input arrays.   
Array1: [ 0 10 20 40 60 80]  
Array2: [10, 30, 40, 50, 70]  
Unique sorted array of values that are in either of the two input arrays:  
[ 0 10 20 30 40 50 60 70 80]

**23.** Write a Python program to test if all elements in an array evaluate to True.   
Note: 0 evaluates to False in python.

**24.** Write a Python program to test whether any array element along a given axis evaluates to True.  
Note: 0 evaluates to False in python.

**25.** Write a Python program to construct an array by repeating.

Sample array: [1, 2, 3, 4]  
Expected Output:  
Original array   
[1, 2, 3, 4]   
Repeating 2 times   
[1 2 3 4 1 2 3 4]  
Repeating 3 times   
[1 2 3 4 1 2 3 4 1 2 3 4]

**26.** Write a Python program to repeat elements of an array.   
Expected Output:  
[3 3 3 3]   
[1 1 2 2 3 3 4 4]

**27.** Write a Python program to find the indices of the maximum and minimum values along the given axis of an array.   
Original array: [1 2 3 4 5 6]   
Maximum Values: 5   
Minimum Values: 0 

**28.** Write a Python program compare two arrays using numpy.   
Array a: [1 2]  
Array b: [4 5]  
a > b   
[False False]  
a >= b   
[False False]   
a < b   
[ True True]   
a <= b   
[ True True]

**29.** Write a Python program to sort an along the first, last axis of an array.   
Sample array: [[2,5],[4,4]]  
Expected Output:  
Original array:  
[[4 6]   
[2 1]]   
Sort along the first axis:  
[[2 1]   
[4 6]]   
Sort along the last axis:  
[[1 2]   
[4 6]] 

**30.** Write a Python program to sort pairs of first name and last name return their indices. (first by last name, then by first name).   
first\_names = (Betsey, Shelley, Lanell, Genesis, Margery)  
last\_names = (Battle, Brien, Plotner, Stahl, Woolum)  
Expected Output:  
[1 3 2 4 0] 

**31.** Write a Python program to get the values and indices of the elements that are bigger than 10 in a given array.   
Original array:  
[[ 0 10 20]   
[20 30 40]]  
Values bigger than 10 = [20 20 30 40]  
Their indices are (array([0, 1, 1, 1]), array([2, 0, 1, 2])) 

**32.** Write a Python program to save a NumPy array to a text file. 

**33.** Write a Python program to find the memory size of a NumPy array.   
Expected Output:  
128 bytes

**34.** Write a Python program to create an array of ones and an array of zeros.   
Expected Output:  
Create an array of zeros  
Default type is float  
[[ 0. 0.]]   
Type changes to int  
[[0 0]]   
Create an array of ones  
Default type is float   
[[ 1. 1.]]   
Type changes to int  
[[1 1]]

**35.** Write a Python program to change the dimension of an array.   
Expected Output:  
6 rows and 0 columns  
(6,)  
(3, 3) -> 3 rows and 3 columns  
[[1 2 3]   
[4 5 6]   
[7 8 9]]  
Change array shape to (3, 3) -> 3 rows and 3 columns   
[[1 2 3]   
[4 5 6]   
[7 8 9]]

**36.** Write a Python program to create a contiguous flattened array.   
Original array:  
[[10 20 30]   
[20 40 50]]   
New flattened array:   
[10 20 30 20 40 50]

**37.** Write a Python program to create a 2-dimensional array of size 2 x 3 (composed of 4-byte integer elements), also print the shape, type and data type of the array.   
Expected Output:  
  
(2, 3)  
int32 

**38.** Write a Python program to create a new shape to an array without changing its data.   
Reshape 3x2:   
[[1 2]   
[3 4]   
[5 6]]   
Reshape 2x3:  
[[1 2 3]   
[4 5 6]]

**39.** Write a Python program to change the data type of an array.   
Expected Output:  
[[ 2 4 6]  
[ 6 8 10]]   
Data type of the array x is: int32   
New Type: float64   
[[ 2. 4. 6.]   
[ 6. 8. 10.]]

**40.** Write a Python program to create a new array of 3\*5, filled with 2.   
Expected Output:  
[[2 2 2 2 2]  
[2 2 2 2 2]  
[2 2 2 2 2]]  
[[2 2 2 2 2]  
[2 2 2 2 2]   
[2 2 2 2 2]]

**41.** Write a Python program to create an array of 10's with the same shape and type of an given array.   
Sample array: x = np.arange(4, dtype=np.int64)  
Expected Output:  
[10 10 10 10]

**42.** Write a Python program to create a 3-D array with ones on a diagonal and zeros elsewhere.   
Expected Output:  
[[ 1. 0. 0.]  
[ 0. 1. 0.]   
[ 0. 0. 1.]]

**43.** Write a Python program to create a 2-D array whose diagonal equals [4, 5, 6, 8] and 0's elsewhere.   
Expected Output:  
[[4 0 0 0]   
[0 5 0 0]   
[0 0 6 0]   
[0 0 0 8]]

**44.** Write a Python program to create a 1-D array going from 0 to 50 and an array from 10 to 50.   
Expected Output:  
Array from 0 to 50:   
[ 0 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]   
Array from 10 to 50:   
[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] 

**45.** Write a Python program to Create a 1-D array of 30 evenly spaced elements between 2.5. and 6.5, inclusive.   
Expected Output:  
[ 2.5 2.63793103 2.77586207 2.9137931 3.05172414 3.18965517   
3.32758621 3.46551724 3.60344828 3.74137931 3.87931034 4.01724138   
4.15517241 4.29310345 4.43103448 4.56896552 4.70689655 4.84482759  
4.98275862 5.12068966 5.25862069 5.39655172 5.53448276 5.67241379  
5.81034483 5.94827586 6.0862069 6.22413793 6.36206897 6.5 ]

**46.** Write a Python program to to create a 1-D array of 20 element spaced evenly on a log scale between 2. and 5., exclusive.   
Expected Output:  
[ 100. 141.25375446 199.5262315 281.83829313  
398.10717055 562.34132519 794.32823472 1122.0184543   
1584.89319246 2238.72113857 3162.27766017 4466.83592151  
6309.5734448 8912.50938134 12589.25411794 17782.79410039  
25118.8643151 35481.33892336 50118.72336273 70794.57843841]

**47.** Write a Python program to create an array which looks like below array.   
Expected Output:  
[[ 0. 0. 0.]  
[ 1. 0. 0.]  
[ 1. 1. 0.]  
[ 1. 1. 1.]]

**48.** Write a Python program to create an array which looks like below array.   
Expected Output:  
[[ 2 3 4]   
[ 5 6 7]   
[ 0 9 10]   
[ 0 0 13]]

**49.** Write a Python program to collapse a 3-D array into one dimension array.   
Expected Output:  
3-D array:   
[[ 1. 0. 0.]   
[ 0. 1. 0.]   
[ 0. 0. 1.]]  
One dimension array:   
[ 1. 0. 0. 0. 1. 0. 0. 0. 1.]

**50.** Write a Python program to find the 4th element of a specified array.   
Expected Output:  
[[ 2 4 6]  
[ 6 8 10]]  
Forth e1ement of the array:  
6

**51.** Write a Python program to interchange two axes of an array.   
Sample array: [[1 2 3]]   
Expected Output:  
[[1]   
[2]   
[3]]

**52.** Write a Python program to create an array of 10's with the same shape and type of an given array.   
Expected Output:  
(3, 4, 2)   
(4, 2, 3)

**53.** Write a Python program to create an array of 10's with the same shape and type of an given array.   
Move the following 3rd array axes to first position.  
(2,3,4,5)  
Sample Expected Output:  
(2, 5, 3, 4)

**54.** Write a Python program to convert specified inputs to arrays with at least one dimension.   
Expected Output:  
[ 12.]   
[[ 0. 1. 2.]  
[ 3. 4. 5.]]  
[array([1]), array([3, 4])]

**55.** Write a Python program to view inputs as arrays with at least two dimensions, three dimensions.   
Expected Output:  
View inputs as arrays with at least two dimensions:  
[10]   
[[ 0. 1.]  
[ 2. 3.]]   
View inputs as arrays with at least three dimensions:  
[[[15]]]   
[[[ 0.]   
[ 1.]   
[ 2.]]] 

**56.** Write a Python program to insert a new axis within a 2-D array.   
2-D array of shape (3, 4).  
Expected Output:  
New shape will be will be (3, 1, 4).

**57.** Write a Python program to remove single-dimensional entries from a specified shape.   
Specified shape: (3, 1, 4)  
Expected Output: (3, 4)

**58.** Write a Python program to concatenate two 2-dimensional arrays.   
Expected Output:  
Sample arrays: ([[0, 1, 3], [5, 7, 9]], [[0, 2, 4], [6, 8, 10]]  
Expected Output:  
[[ 0 1 3 0 2 4]  
[ 5 7 9 6 8 10]] 

**59.** Write a Python program to convert 1-D arrays as columns into a 2-D array.   
Sample array: (10,20,30), (40,50,60)  
Expected Output:  
[[10 40]  
[20 50]  
[30 60]]

**60.** Write a Python program to convert (in sequence depth wise (along third axis)) two 1-D arrays into a 2-D array.   
Sample array: (10,20,30), (40,50,60)  
Expected Output:  
[[[10 40]]  
[[20 50]]  
[[30 60]]]

**61.** Write a Python program to split an array of 14 elements into 3 arrays, each of which has 2, 4, and 8 elements in the original order.   
Expected Output:  
Original array: [ 1 2 3 4 5 6 7 8 9 10 11 12 13 14]  
After splitting:   
[array([1, 2]), array([3, 4, 5, 6]), array([ 7, 8, 9, 10, 11, 12, 13, 14])]

**62.** Write a Python program to split an of array of shape 4x4 it into two arrays along the second axis.   
Sample array :  
[[ 0 1 2 3]  
[ 4 5 6 7]   
[ 8 9 10 11]   
[12 13 14 15]]  
Expected Output:  
[array([[ 0, 1],  
[ 4, 5],   
[ 8, 9],   
[12, 13]]), array([[ 2, 3],  
[ 6, 7],  
[10, 11],   
[14, 15]]), array([], shape=(4, 0), dtype=int64)]

**63.** Write a Python program to get the number of nonzero elements in an array.   
Expected Output:  
Original array:   
[[ 0 10 20]   
[20 30 40]]  
Number of non zero elements in the above array:   
5 

**64.** Write a Python program to create a 5x5 matrix with row values ranging from 0 to 4.   
Original 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.]]   
Row values ranging from 0 to 4.  
[[ 0. 1. 2. 3. 4.]  
[ 0. 1. 2. 3. 4.]   
[ 0. 1. 2. 3. 4.]  
[ 0. 1. 2. 3. 4.]  
[ 0. 1. 2. 3. 4.]]

**65.** Write a Python program to test if specified values are present in an array.   
Expected Output:  
Original array:  
[[ 1.12 2. 3.45]   
[ 2.33 5.12 6. ]]   
True   
False   
True   
False   
True

**66.** Write a Python program to create a vector of size 10 with values ranging from 0 to 1, both excluded.   
Expected Output:  
[ 0.09090909 0.18181818 0.27272727 0.36363636 0.45454545 0.54545455   
0.63636364 0.72727273 0.81818182 0.90909091]

**67.** Write a Python program to make an array immutable (read-only).   
Expected Output:  
Test the array is read-only or not:  
Try to change the value of the first element:  
Traceback (most recent call last):  
File "19236bd0-0bd9-11e7-a232-c706d0968eb6.py", line 6, in   
x[0] = 1  
ValueError: assignment destination is read-only

**68.** Write a Python program (using numpy) to sum of all the multiples of 3 or 5 below 100.   
Expected Output:  
[ 3 5 6 9 10 12 15 18 20 21 24 25 27 30 33 35 36 39 40 42 45 48 50 51 54  
55 57 60 63 65 66 69 70 72 75 78 80 81 84 85 87 90 93 95 96 99]  
2318 

**69.** Write a Python program to create an array with 10^3 elements.   
Expected Output:  
[ 0. 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.   
- - - - - - - - - - - - - - - - - - - -  
972. 973. 974. 975. 976. 977. 978. 979. 980. 981. 982. 983.  
984. 985. 986. 987. 988. 989. 990. 991. 992. 993. 994. 995.   
996. 997. 998. 999.] 

**70.** Write a Python program to create display every element of an numpy array.   
Expected Output:  
0 1 2 3 4 5 6 7 8 9 10 11

**71.** Write a Python program to create and display every element of an numpy array in Fortran order.   
Expected Output:  
Elements of the array in Fortan array:   
0 4 8 1 5 9 2 6 10 3 7 11 

**72.** Write a Python program to create a 5x5x5 cube of 1's.   
Expected Output:  
[[[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]]   
  
[[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]]   
  
[[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]]   
  
[[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]]   
  
[[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]   
[1 1 1 1 1]]]

**73.** Write a Python program to create an array of (3, 4) shape, multiply every element value by 3 and display the new array.   
Expected Output:  
Original array elements:   
[[ 0 1 2 3]   
[ 4 5 6 7]   
[ 8 9 10 11]]   
New array elements:  
[[ 0 3 6 9]   
[12 15 18 21]   
[24 27 30 33]]

**74.** Write a Python program to combine a one and a two dimensional array together and display their elements.   
Expected Output:  
One dimensional array:  
[0 1 2 3]   
Two dimensional array:   
[[0 1 2 3]   
[4 5 6 7]]   
0:0   
1:1   
2:2   
3:3  
0:4  
1:5  
2:6  
3:7

**75.** Write a Python program to create an array of zeros and three column types (integer, float, character).   
Expected Output:  
[(1, 2., b'Albert Einstein') (2, 2., b'Edmond Halley')  
(3, 3., b'Gertrude B. Elion')] 

**76.** Write a Python program to create a function cube which cubes all the elements of an array.   
Expected Output:  
[ 1 8 27]

**77.** Write a Python program to create an array of (3, 4) shape and convert the array elements in smaller chunks.   
Expected Output:  
Original array elements:  
[[ 0 1 2 3]   
[ 4 5 6 7]  
[ 8 9 10 11]]  
[0 4 8]   
[1 5 9]   
[ 2 6 10]  
[ 3 7 11]

**78.** Write a Python program to create a record array from a (flat) list of arrays.   
Sample arrays: [1,2,3,4], ['Red', 'Green', 'White', 'Orange'], [12.20,15,20,40]  
Expected Output:  
(1, 'Red', 12.2)  
(2, 'Green', 15.0)  
(3, 'White', 20.0)

**79.** Write a Python program to generate a generic 2D Gaussian-like array.   
Expected Output:  
2D Gaussian-like array:  
[[ 0.36787944 0.44822088 0.51979489 0.57375342 0.60279818 0.60279818  
0.57375342 0.51979489 0.44822088 0.36787944]   
[ 0.44822088 0.54610814 0.63331324 0.69905581 0.73444367 0.73444367  
0.69905581 0.63331324 0.54610814 0.44822088]   
.......... [ 0.44822088 0.54610814 0.63331324 0.69905581 0.73444367 0.73444367   
0.69905581 0.63331324 0.54610814 0.44822088]   
[ 0.36787944 0.44822088 0.51979489 0.57375342 0.60279818 0.60279818   
0.57375342 0.51979489 0.44822088 0.36787944]]

**80.** Write a Python program to convert a NumPy array into Python list structure.   
Expected Output:  
Original array elements:  
[[0 1]  
[2 3]   
[4 5]]   
Array to list:   
[[0, 1], [2, 3], [4, 5]]

**81.** Write a Python program to access an array by column.   
Expected Output:  
Original array elements:  
[[0 1]   
[2 3]   
[4 5]]   
Access an array by column:  
First column:  
[0 1]   
Second column:   
[2 3]   
Third column:  
[4 5]

**82.** Write a Python program to convert a numpy array of float values to a numpy array of integer values.   
Expected Output:  
Original array elements:  
[[ 12. 12.51]  
[ 2.34 7.98]   
[ 25.23 36.5 ]]  
Conver float values to intger values:  
[[12 12]   
[ 2 7]   
[25 36]]

**83.** Write a Python program to convert a NumPy array into Python list structure.   
Expected Output:  
Original array elements:  
[ 0.26153123 0.52760141 0.5718299 0.5927067 0.7831874 0.69746349   
0.35399976 0.99469633 0.0694458 0.54711478]   
Print array values with precision 3:   
[ 0.262 0.528 0.572 0.593 0.783 0.697 0.354 0.995 0.069 0.547]

**84.** Write a Python program to suppresses the use of scientific notation for small numbers in numpy array.    
Expected Output:  
Original array elements:  
[ 1.60000000e-10 1.60000000e+00 1.20000000e+03 2.35000000e-01]   
Print array values with precision 3:   
[ 0. 1.6 1200. 0.235]

**85.** Write a Python program to create a numpy array of 10 integers from a generator.   
  
Expected Output:  
[0 1 2 3 4 5 6 7 8 9]

**86.** Write a Python program to how to add an extra column to an numpy array.   
  
Expected Output:  
[[ 10 20 30 100]  
[ 40 50 60 200]]

**87.** Write a Python program to find unique rows in a numpy array.   
  
Expected Output:  
Original array:  
[[20 20 20 0]  
[ 0 20 20 20]  
[ 0 20 20 20]  
[20 20 20 0]   
[10 20 20 20]]  
Unique rows of the above array:  
[[ 0 20 20 20]   
[10 20 20 20]  
[20 20 20 0]]

**88.** Write a Python program to replace all elements of numpy array that are greater than specified array.   
  
Expected Output:  
Original array:  
[[ 0.42436315 0.48558583 0.32924763]   
[ 0.7439979 0.58220701 0.38213418]  
[ 0.5097581 0.34528799 0.1563123 ]]  
Replace all elements of the said array with .5 which are greater than. 5  
[[ 0.42436315 0.48558583 0.32924763]   
[ 0.5 0.5 0.38213418]   
[ 0.5 0.34528799 0.1563123 ]]

**89.** Write a Python program to remove specific elements in a numpy array.   
  
Expected Output:  
Original array:   
[ 10 20 30 40 50 60 70 80 90 100]  
Delete first, fourth and fifth elements:  
[ 20 30 60 70 80 90 100]

**90.** Write a Python program to remove the negative values in a numpy array with 0.   
  
Expected Output:  
Original array:  
[-1 -4 0 2 3 4 5 -6]  
Replace the negative values of the said array with 0:  
[0 0 0 2 3 4 5 0]

**91.** Write a Python program to remove all rows in a numpy array that contain non-numeric values.   
Expected Output:  
Original array:   
[[ 1. 2. 3.]   
[ 4. 5. nan]  
[ 7. 8. 9.]   
[ 1. 0. 1.]]   
Remove all non-numeric elements of the said array  
[[ 1. 2. 3.]   
[ 7. 8. 9.]   
[ 1. 0. 1.]] 

**92.** Write a Python program to remove all rows in a numpy array that contain non-numeric values.   
Sample array :  
a = np.array([97, 101, 105, 111, 117])  
b = np.array(['a','e','i','o','u'])  
Note: Select the elements from the second array corresponding to elements in the first array that are greater than 100 and less than 110  
Expected Output:  
Original arrays   
[ 97 101 105 111 117]   
['a' 'e' 'i' 'o' 'u']   
Elements from the second array corresponding to elements in the first   
array that are greater than 100 and less than 110:   
['e' 'i']

**93.** Write a Python program to get the magnitude of a vector in numpy.   
Expected Output:  
Original array:  
[1 2 3 4 5]  
Magnitude of the vector:  
7.4161984871

**94.** Write a Python program to count the frequency of unique values in numpy array.   
Expected Output:  
Original array:  
[10 10 20 10 20 20 20 30 30 50 40 40]   
Frequency of unique values of the said array:  
[[10 20 30 40 50]   
[ 3 4 2 2 1]]

**95.** Write a Python program to check whether the numpy array is empty or not.   
Expected Output:  
2   
0

**96.** Write a Python program to divide each row by a vector element.   
Expected Output:  
Original array:  
[[20 20 20]   
[30 30 30]   
[40 40 40]]  
Vector:   
[20 30 40]  
[[ 1. 1. 1.]  
[ 1. 1. 1.]   
[ 1. 1. 1.]]

**97.** Write a Python program to print all the values of an array.   
Expected Output:  
[[ 0. 0. 0. 0.]  
[ 0. 0. 0. 0.]  
[ 0. 0. 0. 0.]  
[ 0. 0. 0. 0.]]

**98.** Write a Python program to convert the raw data in an array to a binary string and then create an array.   
Expected Output:  
Original array:  
[ 10. 20. 30.]  
Binary string array:  
b'\x00\x00\x00\x00\x00\x00$@\x00\x00\x00\x00\x00\x004@\x00\x00\x00\x00\x00\x00>@'   
Array using fromstring():   
[ 10. 20. 30.]

**99.** Write a Python program to sum and compute the product of a numpy array elements.   
Expected Output:  
Original array:   
[ 10. 20. 30.]   
Sum of the array elements:  
60.0   
Product of the array elements:   
6000.0 

**100.** Write a Python program to take values from a source array and put them at specified indices of another array.   
Expected Output:  
[ 10. 10. 20. 30. 30.]   
Put 0 and 40 in first and fifth position of the above array   
Array x after put two values: [ 0. 10. 20. 30. 40.] 