1.What is Numpy?

Ans: NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. … A powerful N-dimensional array object. Sophisticated (broadcasting) functions.

2. Why NumPy is used in Python?

Ans: NumPy is a package in **Python used** for Scientific Computing. **NumPy** package is **used** to perform different operations. The ndarray (**NumPy** Array) is a multidimensional array **used** to store values of same datatype. These arrays are indexed just like Sequences, starts with zero.

---

3. What does NumPy mean in Python?

Ans: NumPy (pronounced /ˈnʌmpaɪ/ (NUM-py) or sometimes /ˈnʌmpi/ (NUM-pee)) is a library for the **Python** programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

---

4. Where is NumPy used?

Ans: NumPy is an open source numerical Python library. **NumPy** contains a multi-dimentional array and matrix data structures. It can be utilised to perform a number of mathematical operations on arrays such as trigonometric, statistical and algebraic routines. **NumPy** is an extension of Numeric and Numarray.

---

5. How to Install Numpy in Windows?

Ans:

1. **Step** 1: Download **Python** for **Windows** 10/8/7. First, download the **Python** executable binaries on your **Windows** system from the official download the page of the **Python**. …
2. **Step** 2: **Run** the **Python** executable **installer**. …
3. **Step** 3: **Install** pip on **Windows** 10/8/7. …
4. **Step** 4: **Install Numpy** in **Python** using pip on **Windows** 10/8/7.

Installation Process of Numpy..
step1: Open the terminal
step2: type **pip install numpy**

---

6. how to import numpy in python?

Ans:
import numpy as np

---

7. how to create 1D Array ?

Ans:

```
num=[1,2,3]
num = np.array(num)
print("1d array : ",num)
```

---

8. how to create 2D Array ?

Ans:

```
num2=[[1,2,3],[4,5,6]]
num2 = np.array(num2)
```

print(“\n2d array : “,num2)

9. how to create 3D Array or ND Array ?

Ans:
num3=[[[1,2,3],[4,5,6],[7,8,9]]]
num3 = np.array(num3)
print(“\n3d array : “,num3)

10. how to use shape for 1D Array ?

Ans:
num=[1,2,3] if not defined
print(‘\nshpae of 1d ‘,num.shape)

11. how to use shape for 2D Array ?

Ans:
num2=[[1,2,3],[4,5,6]] if not added
print(‘\nshpae of 2d ‘,num2.shape)

12. how to use shape for 3d or Nd Array ?

Ans: num3=[[[1,2,3],[4,5,6],[7,8,9]]] if not added
print(‘\nshpae of 3d ‘,num3.shape)

13. how to identified datatyp for numpy array?

Ans: print(‘\n data type num 1 ‘,num.dtype)
print(‘\n data type num 2 ‘,num2.dtype)
print(‘\n data type num 3 ‘,num3.dtype)

---

14. Print 5 zeros ?

Ans: arr = np.zeros(5)
print(‘single arrya’,arr)

---

15. print zeros with 2 rows and 3 columns ?

Ans: arr2 = np.zeros((2,3))
print(‘\nprint 2 rows and 3 cols : ‘,arr2)

---

16. use of eye() diagonal values ?

Ans: arr3 = np.eye(4)
print(‘\ndiaglonal values : ‘,arr3)

---

17. use of diag() square matrix ?

Ans:
arr3 = np.diag([1,2,3,4])
print(‘\n square matrix’,arr3)

---

18. Print Range Between 1 To 15 and show 4 integers random numbers

Ans: rand_arr = np.random.randint(1,15,4)
print(‘\n random number from 1 to 15 ‘,rand_arr)

---

19. Print Range Between 1 To 100 and show 4 integers random numbers

Ans: rand_arr3 = np.random.randint(1,100,20)
print('\n random number from 1 to 100 ',rand_arr3)

20.Print Range Between random number 2 row and 3 cols integers random numbers

Ans: rand_arr2 = np.random.randint([2,3])
print('\n random number 2 row and 3 cols ',rand_arr2)

21.describe the example of seed() function? and how to use it ? why seed()?

Ans: np.random.seed(123)
rand_arr4 = np.random.randint(1,100,20)
print('\nseed() showing same number only : ',rand_arr4)

---

22. What is 1d indexing? define one ex : num = np.array([5,15,25,35])?

Ans: num = np.array([5,15,25,35])
print('my array : ',num)

---

23. Print first position, last position and 2nd and 3rd position

Ans: num = np.array([5,15,25,35]) if not added
print('\n first position : ',num[0]) #5
print('\n third position : ',num[2]) #25

---

24. how to identified last number of numpy array?

Ans: num = np.array([5,15,25,35]) if not added
print('\n forth position : ',num[3])

---

25. if we don't know last number of position how to show it by pragmatically ?

Ans: num = np.array([5,15,25,35]) if not added
print('\n last indexing done by -1 position : ',num[-1])

---

26. 1D Slicing with above numpy array..print[5,15]

Ans: num = np.array([5,15,25,35]) if not added
print('\n first and third position : ',num[0:2])

---

27. 1D Slicing with above numpy arra..print[5,15,25,35]

Ans: print('\n second upto last : ',num[1:])

---

28. What is 2d indexing? define one ex : num = np.array([1,2,3],[4,5,6],[7,8,9])

Ans: num = np.array([[1,2,3],[4,5,6],[7,8,9]]) #double bracket here for 2d array
print('\n my 2d array : ',num)

---

29. print row 1 column 1 output will 5

Ans: num = np.array([[1,2,3],[4,5,6],[7,8,9]])
print('\nprint 5 :',num[1][1])

---

30. Print row 2 column 2 output will 9

Ans: num = np.array([[1,2,3],[4,5,6],[7,8,9]])
print('\nprint 9 : ',num[2][2])
num2 = np.arange(10,101)
print(num2)

---

31. create a matrix 3 * 3 with value ranging from 0 to 8

Ans: arr = np.arange(0,9).reshape(3,3)
print(arr)

---

32. create matrix 2 * 2 with value ranging from 1 to 3

Ans: arr = np.arange(0,4).reshape(2,2)
print(arr)

---

33. create matrix 2 * 2 with value ranging from 1 to 4

Ans: arr = np.arange(1,5).reshape(2,2)
print(arr)

---

34. print random number from 0 to 1

Ans: print('\n random number from 0 to 1 ',np.random.rand())

---

35. use numpy to generate array of 25 random numbers sampled from a standard normal distribution

Ans: print('\n random number 25\n ',np.random.rand(25))

---

36. insert 1 to 100 numbers and formated with 10*10

Ans: num = np.arange(1,101).reshape(10,10)
print('\n reshape after\n',num)

---

37. print size of numarray

Ans: print('\n size \n',num.size) #100

---

38. use of shape()

Ans: print('\n shape \n',num.shape) # 10,10

---

39. create an array of 20 linearly spaced point between 0 to 1

Ans: num_line = np.linspace(0,1,20)
print(num_line)
#output
#[0. 0.05263158 0.10526316 0.15789474 0.21052632 0.26315789
0.31578947 0.36842105 0.42105263 0.47368421 0.52631579 0.57894737
0.63157895 0.68421053 0.73684211 0.78947368 0.84210526 0.89473684

0.94736842 1.]

---

40. create 4*4 and print 0 to 12

Ans: num_arr = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
print(num_arr)

---

41. Print number upto 8

Ans: print('\n print 8 ',num_arr[:2]) #[[1 2 3 4][5 6 7 8]]

---

42 Print number from 9

Ans: print('\n print from 8 to 12 ',num_arr[:3]) #[[9 10 11 12]]

---

43. Print read after 5 to n

Ans: print('\n print 5 to n \n',num_arr[1:]) # [[5 6 7 8] [9 10 11 12]]

---

44. 2d slicing or describe 2d slicing with own example ?

Ans:
ex 1
[1,2]
[4,5]]
print('slicing of ',num[:2,:2])

---

45. ex 2####[[2,3]### [5,6]]

Ans: print('slicing of ',num[1:,1:])

---

46. use of reshape ex.1 starting value is 1 and ending value is 50, print 10 integer random numbers

Ans: num = np.random.randint(1,50,10)
print('\nprin random numbers 1 to 50 with 10 rand numbers : ',num)

---

47. Print shape

Ans: num = np.random.randint(1,50,10)
print('\nprint shape befor reshape : ',num.shape)

---

48. Print reshape devide into array like 2 array with 5 values

Ans: print('\nreshaped array',num.reshape(2,5))

---

49. using ravel() we can combine arrays into single

Ans: a = np.array([(1,2,3),(4,5,6)])
print(a.ravel())

---

50. Print number from 1 to 10 using arange() function

Ans: num = np.arange(1,11)
print(num)

---

51. mask convert boolean values to number #num = np.arange(1,11)

Ans: mask = num>5
print('print bigger than 5 numbers',num[mask]) #5,7,8,9,10

---

52. mask convert boolean values to number #num = np.arange(1,11)

Ans: mask = num<4
print('print lower than 4 numbers',num[mask]) #1,2,3

---

53. print value who divide by 2 and remain is 0.

Ans: print('\n divided by 2 ',num[num%2==0]) #2,4,6,8,10

54.what is the use of == operator ?

Ans: print('\n print 2 is equal to num',num[num==2]) #print [2]
print('\n print 0 is equal to num',num[num==0]) #print [] because empty 0 is not available

---

55. Linear algebra

Ans: Dot product: product of two arrays
f = np.array([1,2])
g = np.array([4,5])
1*4+2*5
np.dot(f, g)

56.use of concatenate()

Ans: x=np.array([[1,2],[3,4]])
y=np.array([[5,6]])
z=np.concatenate((x,y))
print("array of z value is : \n",z)

57.how to use of concatenate() with axis=0

Ans: x=np.array([[1,2],[3,4]])
y=np.array([[5,6]])
z=np.concatenate((x,y),axis=None) #
print("\narray of z value is with o axis : ",z) # [1 2 3 4 5 6]

58.Print the probabilty arr = [0.23, 0.09, 1.2, 1.24, 9.99] using fix() this example of fuzzy logic or probability

Ans: arr_num = [0.23, 0.09, 1.2, 1.24, 9.99]
print("Input probability : ",arr)
out_arr = np.fix(arr_num)
print("Output probability : ",out_arr)

59.what is the use sort() function ?

Ans: num_arr = np.array([[5,6,2],[9,8,1]])
print('\n sorting of numpy array : \n',np.sort(num_arr))

---

60. matrix of two arrays(array1,array2) and combine into third array (result)

Ans: array1=np.array([[1,2,3],[4,5,6],[7,8,9]],ndmin=3)
array2=np.array([[9,8,7],[6,5,4],[3,2,1]],ndmin=3)
result=np.multiply(array1,array2)
print('\n result \n',result)

61.what is the use of append() function with axis=0 ?

Ans: a=np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
b=np.array([[11, 21, 31], [42, 52, 62], [73, 83, 93]])
c=np.append(a,b,axis=0)
print('\n append function with axis = 0\n ',c)

---

62. what is the use of append() function with axis=1 ?

Ans: a=np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
b=np.array([[11, 21, 31], [42, 52, 62], [73, 83, 93]])
c=np.append(a,b,axis=1)
print('append function with axis = 1\n ',c)

63.What is meaning of axis=0 and axis=1?

Ans: axis=0 is used for reading Rows and axix=1 is used for reading Colums.

---

64. What is the use of rint() function explain with example ?

Ans: arr = [0.23, 0.09, 1.2, 1.24, 9.99]
print("Input array:",arr)
r_arr = np.rint(arr)
print("Output array:",r_arr)

---

65. What is the use of transpose() function explain with example ?

Ans: aa= np.array([[1,2],[4,5],[7,8]])
print(",aa)
bb=np.transpose(a,(1,0))
print(bb)

66. what is Numpy?

Ans:
-It is python library designed for scientif computation.
-Numpy arrays are the main way to use NumPy Library.
1-d array -vector
2-d array-matrix
n-d array-tensor

---

67. features of NumPy?

Ans: Its a array-processing package. It provides multidimensional array object, and tools for working with these arrays with high-performance.
It contains various features as follows:
o A powerful N-dimensional array object
o Sophisticated (broadcasting) functions
o Tools for integrating C/C++ and Fortran code
Useful linear algebra, Fourier transform, and random number capabilities
NumPy can also be used as an efficient multi-dimensional container of generic data.
Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a large variety of databases.

---

68. why numpy created ?

Ans: NumPy were created to do numerical and scientific computing in the most natural way with Python.

---

69. where we can use ?

Ans: Numpy is a library for the Python, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high level mathematical functions. In simple words, Numpy is an optimized version of Python lists.
You can use numpy for:
1)…Financial functions
2)…Linear Algebra
3)…Statistics
4)…Polynomials
5)…Sorting, searching etc.

---

70. How to extract items that satisfy a given condition from 1D array?

Extract all odd numbers from **arr**

```
# Input
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

# Solution
arr[arr % 2 == 1]
```

Output: array([1, 3, 5, 7, 9])

---

71. How to replace items that satisfy a condition with another value in numpy array?

```
arr[arr % 2 == 1] = -1
arr
```
Output: array([0, -1, 2, -1, 4, -1, 6, -1, 8, -1])

---

72. How to replace items that satisfy a condition without affecting the original array?

```
arr = np.arange(10)
out = np.where(arr % 2 == 1, -1, arr)
print(arr)
out
```

Output: [0 1 2 3 4 5 6 7 8 9]
array([0, -1, 2, -1, 4, -1, 6, -1, 8, -1])

---

73. How to reshape an array?

```
arr = np.arange(10)
arr.reshape(2, -1)  # Setting to -1 automatically decides the number of cols
```

Output:
```
> array([[0, 1, 2, 3, 4],
>        [5, 6, 7, 8, 9]])
```

---

74. How to stack two arrays vertically?

```
a = np.arange(10).reshape(2,-1)
b = np.repeat(1, 10).reshape(2,-1)

# Answers
# Method 1:
np.concatenate([a, b], axis=0)

# Method 2:
np.vstack([a, b])

# Method 3:
np.r_[a, b]
```

Output:
```
#> array([[0, 1, 2, 3, 4],
#>        [5, 6, 7, 8, 9],
#>        [1, 1, 1, 1, 1],
#>        [1, 1, 1, 1, 1]])
```

---

75. How to stack two arrays horizontally?

```
a = np.arange(10).reshape(2,-1)
b = np.repeat(1, 10).reshape(2,-1)

# Answers
# Method 1:
np.concatenate([a, b], axis=1)

# Method 2:
np.hstack([a, b])

# Method 3:
np.c_[a, b]
```

Output:
```
#> array([[0, 1, 2, 3, 4, 1, 1, 1, 1, 1],
#>        [5, 6, 7, 8, 9, 1, 1, 1, 1, 1]])
```

---

76. How to generate custom sequences in numpy without hardcoding?

```
np.r_[np.repeat(a, 3), np.tile(a, 3)]
```

Output:
```
#> array([1, 1, 1, 2, 2, 2, 3, 3, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3])
```

77.How to get the common items between two python numpy arrays?

```
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])
np.intersect1d(a,b)
```

Output:

```
#> array([2, 4])
```

---

78. How to remove from one array those items that exist in another?

```
a = np.array([1,2,3,4,5])
b = np.array([5,6,7,8,9])

# From 'a' remove all of 'b'
np.setdiff1d(a,b)
```

Output:

```
#> array([1, 2, 3, 4])
```

---

79. How to get the positions where elements of two arrays match?

```
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])

np.where(a == b)
```

Output:

```
#> (array([1, 3, 5, 7]),)
```

---

80. How to extract all numbers between a given range from a numpy array?

```
a = np.arange(15)

# Method 1
index = np.where((a >= 5) & (a <= 10))
a[index]

# Method 2:
index = np.where(np.logical_and(a>=5, a<=10))
a[index]
#> (array([6, 9, 10]),)

# Method 3: (thanks loganzk!)
a[(a >= 5) & (a <= 10)]
```

---

81. How to make a python function that handles scalars to work on numpy arrays?

```
def maxx(x, y):
    """Get the maximum of two items"""
    if x >= y:
        return x
    else:
        return y

pair_max = np.vectorize(maxx, otypes=[float])

a = np.array([5, 7, 9, 8, 6, 4, 5])
b = np.array([6, 3, 4, 8, 9, 7, 1])

pair_max(a, b)
```

Output:

```
#> array([ 6.,  7.,  9.,  8.,  9.,  7.,  5.])
```

---

82. How to swap two columns in a 2D numpy array?

```
# Input
arr = np.arange(9).reshape(3,3)
arr
```

```
# Solution
arr[:, [1,0,2]]
```

Output:

```
#> array([[1, 0, 2],
#>        [4, 3, 5],
#>        [7, 6, 8]])
```

---

83. How to swap two rows in a 2d numpy array?

```
# Input
arr = np.arange(9).reshape(3,3)

# Solution
arr[[1,0,2], :]
```

Output:

```
#> array([[3, 4, 5],
#>        [0, 1, 2],
#>        [6, 7, 8]])
```

---

84. How to reverse the rows of a 2D array?

```
# Input
arr = np.arange(9).reshape(3,3)

# Solution
arr[::-1]
```

Output:

```
array([[6, 7, 8],
       [3, 4, 5],
       [0, 1, 2]])
```

---

85. How to reverse the columns of a 2D array?

```
# Input
arr = np.arange(9).reshape(3,3)

# Solution
arr[:, ::-1]
```

Output:

```
#> array([[2, 1, 0],
#>        [5, 4, 3],
#>        [8, 7, 6]])
```

---

86. How to create a 2D array containing random floats between 5 and 10?

```
# Input
arr = np.arange(9).reshape(3,3)

# Solution Method 1:
rand_arr = np.random.randint(low=5, high=10, size=(5,3)) +
np.random.random((5,3))
# print(rand_arr)

# Solution Method 2:
rand_arr = np.random.uniform(5,10, size=(5,3))
print(rand_arr)
```

Output:

```
#> [[ 8.50061025  9.10531502  6.85867783]
#>  [ 9.76262069  9.87717411  7.13466701]
#>  [ 7.48966403  8.33409158  6.16808631]
#>  [ 7.75010551  9.94535696  5.27373226]
#>  [ 8.0850361   5.56165518  7.31244004]]
```

---

87. How to print only 3 decimal places in python numpy array?

```
# Input
```

```
rand_arr = np.random.random((5,3))

# Create the random array
rand_arr = np.random.random([5,3])

# Limit to 3 decimal places
np.set_printoptions(precision=3)
rand_arr[:4]
```

Output:

```
#> array([[ 0.443,  0.109,  0.97 ],
#>        [ 0.388,  0.447,  0.191],
#>        [ 0.891,  0.474,  0.212],
#>        [ 0.609,  0.518,  0.403]])
```

---

88. How to pretty print a numpy array by suppressing the scientific notation (like 1e10)?

```
# Reset printoptions to default
np.set_printoptions(suppress=False)

# Create the random array
np.random.seed(100)
rand_arr = np.random.random([3,3])/1e3
rand_arr
```

Output:

```
#> array([[  5.434049e-04,   2.783694e-04,   4.245176e-04],
#>        [  8.447761e-04,   4.718856e-06,   1.215691e-04],
#>        [  6.707491e-04,   8.258528e-04,   1.367066e-04]])
```

```
np.set_printoptions(suppress=True, precision=6)  # precision is optional
rand_arr
#> array([[ 0.000543,  0.000278,  0.000425],
#>        [ 0.000845,  0.000005,  0.000122],
#>        [ 0.000671,  0.000826,  0.000137]])
```

---

89. How to limit the number of items printed in output of numpy array?

```
>> np.set_printoptions(threshold=6)
>>> a=np.arange(15)
>>> a
```

Output:

```
array([ 0, 1, 2, ..., 12, 13, 14])
```

---

90. How to print the full numpy array without truncating?

```
np.set_printoptions(threshold=6)
a = np.arange(15)

# Solution
np.set_printoptions(threshold=np.nan)
a
#> array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

---

91. How to import a dataset with numbers and texts keeping the text intact in python numpy?

```
# Solution
url =
'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')
names = ('sepallength', 'sepalwidth', 'petallength', 'petalwidth', 'species')

# Print the first 3 rows
iris[:3]
```

Output:

```
#> array([[b'5.1', b'3.5', b'1.4', b'0.2', b'Iris-setosa'],
#>        [b'4.9', b'3.0', b'1.4', b'0.2', b'Iris-setosa'],
#>        [b'4.7', b'3.2', b'1.3', b'0.2', b'Iris-setosa']], dtype=object)
```

92. How to extract a particular column from 1D array of tuples?

```
# Input:
url =
'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris_1d = np.genfromtxt(url, delimiter=',', dtype=None)
print(iris_1d.shape)

# Solution:
species = np.array([row[4] for row in iris_1d])
species[:5]
#> (150,)
#> array([b'Iris-setosa', b'Iris-setosa', b'Iris-setosa', b'Iris-setosa',
#>        b'Iris-setosa'],
#>       dtype='|S18')
```

93. How to convert a 1d array of tuples to a 2d numpy array?

```
# Input:
url =
'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris_1d = np.genfromtxt(url, delimiter=',', dtype=None)

# Solution:
# Method 1: Convert each row to a list and get the first 4 items
iris_2d = np.array([row.tolist()[:4] for row in iris_1d])
iris_2d[:4]

# Alt Method 2: Import only the first 4 columns from source url
iris_2d = np.genfromtxt(url, delimiter=',', dtype='float', usecols=[0,1,2,3])
iris_2d[:4]
```

Output:

```
#> 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]])
```

94. How to compute the mean, median, standard deviation of a numpy array?

```
# Input
url =
'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')
sepallength = np.genfromtxt(url, delimiter=',', dtype='float', usecols=[0])

# Solution
mu, med, sd = np.mean(sepallength), np.median(sepallength), np.std(sepallength)
print(mu, med, sd)
```

Output:

```
#> 5.84333333333 5.8 0.825301291785
```

95. How to normalize an array so the values range exactly between 0 and 1?

```
# Input
url =
'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
sepallength = np.genfromtxt(url, delimiter=',', dtype='float', usecols=[0])

# Solution
Smax, Smin = sepallength.max(), sepallength.min()
S = (sepallength - Smin)/(Smax - Smin)
# or
S = (sepallength - Smin)/sepallength.ptp()  # Thanks, David Ojeda!
print(S)
```

Output:

```
#> [ 0.222  0.167  0.111  0.083  0.194  0.306  0.083  0.194  0.028  0.167
#>   0.306  0.139  0.139  0.     0.417  0.389  0.306  0.222  0.389  0.222
#>   0.306  0.222  0.083  0.222  0.139  0.194  0.194  0.25   0.25   0.111
```

```
#>   0.139  0.306  0.25   0.333  0.167  0.194  0.333  0.167  0.028  0.222
#>   0.194  0.056  0.028  0.194  0.222  0.139  0.222  0.083  0.278  0.194
#>   0.75   0.583  0.722  0.333  0.611  0.389  0.556  0.167  0.639  0.25
#>   0.194  0.444  0.472  0.5    0.361  0.667  0.361  0.417  0.528  0.361
#>   0.444  0.5    0.556  0.5    0.583  0.639  0.694  0.667  0.472  0.389
#>   0.333  0.333  0.417  0.472  0.306  0.472  0.667  0.556  0.361  0.333
#>   0.333  0.5    0.417  0.194  0.361  0.389  0.389  0.528  0.222  0.389
#>   0.556  0.417  0.778  0.556  0.611  0.917  0.167  0.833  0.667  0.806
#>   0.611  0.583  0.694  0.389  0.417  0.583  0.611  0.944  0.944  0.472
#>   0.722  0.361  0.944  0.556  0.667  0.806  0.528  0.5    0.583  0.806
#>   0.861  1.     0.583  0.556  0.5    0.944  0.556  0.583  0.472  0.722
#>   0.667  0.722  0.417  0.694  0.667  0.667  0.556  0.611  0.528  0.444]
```

---

96. How to compute the softmax score?

```
# Input
url =
'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')
sepallength = np.array([float(row[0]) for row in iris])

# Solution
def softmax(x):
    """Compute softmax values for each sets of scores in x.

https://stackoverflow.com/questions/34968722/how-to-implement-the-softmax-funct
ion-in-python"""
    e_x = np.exp(x - np.max(x))
    return e_x / e_x.sum(axis=0)

print(softmax(sepallength))
#> [ 0.002  0.002  0.001  0.001  0.002  0.003  0.001  0.002  0.001  0.002
#>   0.003  0.002  0.002  0.001  0.004  0.004  0.003  0.002  0.004  0.002
#>   0.003  0.002  0.001  0.002  0.002  0.002  0.002  0.002  0.002  0.001
#>   0.002  0.003  0.002  0.003  0.002  0.002  0.003  0.002  0.001  0.002
#>   0.002  0.001  0.001  0.002  0.002  0.002  0.002  0.001  0.003  0.002
#>   0.015  0.008  0.013  0.003  0.009  0.004  0.007  0.002  0.01   0.002
#>   0.002  0.005  0.005  0.006  0.004  0.011  0.004  0.004  0.007  0.004
#>   0.005  0.006  0.007  0.006  0.008  0.01   0.012  0.011  0.005  0.004
#>   0.003  0.003  0.004  0.005  0.003  0.005  0.011  0.007  0.004  0.003
#>   0.003  0.006  0.004  0.002  0.004  0.004  0.004  0.007  0.002  0.004
#>   0.007  0.004  0.016  0.007  0.009  0.027  0.002  0.02   0.011  0.018
#>   0.009  0.008  0.012  0.004  0.004  0.008  0.009  0.03   0.03   0.005
#>   0.013  0.004  0.03   0.007  0.011  0.018  0.007  0.006  0.008  0.018
#>   0.022  0.037  0.008  0.007  0.006  0.03   0.007  0.008  0.005  0.013
#>   0.011  0.013  0.004  0.012  0.011  0.011  0.007  0.009  0.007  0.005]
```

---

97. How to find the percentile scores of a numpy array?

```
>> url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
>>> sepallength = np.genfromtxt(url, delimiter=',', dtype='float', usecols=[0])
>>> np.percentile(sepallength, q=[5, 95])
```

Output:

```
array([4.6 , 7.255])
```

---

98. How to insert values at random positions in an array?

```
>> url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
>>> iris_2d = np.genfromtxt(url, delimiter=',', dtype='object')
>>> i, j = np.where(iris_2d)
>>> np.random.seed(100)
>>> iris_2d[np.random.choice((i), 20), np.random.choice((j), 20)] = np.nan
>>> np.random.seed(100)
>>> iris_2d[np.random.randint(150, size=20), np.random.randint(4, size=20)] = np.nan
>>> print(iris_2d[:10])
```

Output:

```
[[b'5.1' b'3.5' b'1.4' b'0.2' b'Iris-setosa']
[b'4.9' b'3.0' b'1.4' b'0.2' b'Iris-setosa']
[b'4.7' b'3.2' b'1.3' b'0.2' b'Iris-setosa']
...
```

[b’5.0′ b’3.4′ b’1.5′ b’0.2′ b’Iris-setosa’]
[b’4.4′ nan b’1.4′ b’0.2′ b’Iris-setosa’]
[b’4.9′ b’3.1′ b’1.5′ b’0.1′ b’Iris-setosa’]]
>>>

---

99. How to find the position of missing values in numpy array?

```
>> url = ‘https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data’
>>> iris_2d = np.genfromtxt(url, delimiter=’,’, dtype=’float’, usecols=[0,1,2,3])
>>> iris_2d[np.random.randint(150, size=20), np.random.randint(4, size=20)] = np.nan
>>> print(“Number of missing values: \n”, np.isnan(iris_2d[:, 0]).sum())
```

Output 1:

```
Number of missing values:
5
>>> print(“Position of missing values: \n”, np.where(np.isnan(iris_2d[:, 0])))
```

Output 2:

```
Position of missing values:
(array([ 38, 80, 106, 113, 121], dtype=int64),)
>>>
```

---

100. How to filter a numpy array based on two or more conditions?

```
>> url = ‘https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data’
>>> iris_2d = np.genfromtxt(url, delimiter=’,’, dtype=’float’, usecols=[0,1,2,3])
>>> condition = (iris_2d[:, 2] > 1.5) & (iris_2d[:, 0] < 5.0)
>>> iris_2d[condition]
```

Output:

```
array([[4.8, 3.4, 1.6, 0.2],
[4.8, 3.4, 1.9, 0.2],
[4.7, 3.2, 1.6, 0.2],
[4.8, 3.1, 1.6, 0.2],
[4.9, 2.4, 3.3, 1. ],
[4.9, 2.5, 4.5, 1.7]])
```

101.How to drop rows that contain a missing value from a numpy array?

Ans: Select the rows of iris_2d that does not have any nan value.

```
url = ‘https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data‘
iris_2d = np.genfromtxt(url, delimiter=’,’, dtype=’float’, usecols=[0,1,2,3])
iris_2d[np.random.randint(150, size=20), np.random.randint(4, size=20)] = np.nan

# Solution
# No direct numpy function for this.
# Method 1:
any_nan_in_row = np.array([~np.any(np.isnan(row)) for row in iris_2d])
iris_2d[any_nan_in_row][:5]
# Method 2: (By Rong)
iris_2d[np.sum(np.isnan(iris_2d), axis = 1) == 0][:5]
```

Output:

```
#> array([[ 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]])
```

---

102. How to find the correlation between two columns of a numpy array?

Ans: Find the correlation between SepalLength(1st column) and PetalLength(3rd column) in iris_2d

```
# Input
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='float', usecols=[0,1,2,3])
```

```
# Solution 1
np.corrcoef(iris[:, 0], iris[:, 2])[0, 1]

# Solution 2
from scipy.stats.stats import pearsonr
corr, p_value = pearsonr(iris[:, 0], iris[:, 2])
print(corr)
```

Output:
#> 0.871754157305

---

103. How to find if a given array has any null values?

Ans: Find out if iris_2d has any missing values.

```
# Inputurl = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris_2d = np.genfromtxt(url, delimiter=',', dtype='float', usecols=[0,1,2,3])
np.isnan(iris_2d).any()
#> False
```

104.How to replace all missing values with 0 in a numpy array?

Ans: Replace all ccurrences of nan with 0 in numpy array

```
> import numpy as np
>> url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
>> iris_2d = np.genfromtxt(url, delimiter=',', dtype='float', usecols=[0,1,2,3])
>> iris_2d[np.random.randint(150, size=20), np.random.randint(4, size=20)] = np.nan
>> iris_2d[np.isnan(iris_2d)] = 0
>>> iris_2d[:4]
```

Output:

```
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]])
```

---

105. How to find the count of unique values in a numpy array?

Ans: Find the unique values and the count of unique values in iris's species

```
# Import iris keeping the text column intact
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')
names = ('sepallength', 'sepalwidth', 'petallength', 'petalwidth', 'species')

# Solution
# Extract the species column as an array
species = np.array([row.tolist()[4] for row in iris])

# Get the unique values and the counts
np.unique(species, return_counts=True)
```

Output:

```
#> (array([b'Iris-setosa', b'Iris-versicolor', b'Iris-virginica'],
#>        dtype='|S15'), array([50, 50, 50]))
```

---

106. How to convert a numeric to a categorical (text) array?

Ans: Bin the petal length (3rd) column of iris_2d to form a text array, such that if petal length is:

```
# Input
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')
```

```
names = ('sepallength', 'sepalwidth', 'petallength', 'petalwidth', 'species')

# Bin petallength
petal_length_bin = np.digitize(iris[:, 2].astype('float'), [0, 3, 5, 10])

# Map it to respective category
label_map = {1: 'small', 2: 'medium', 3: 'large', 4: np.nan}
petal_length_cat = [label_map[x] for x in petal_length_bin]

# View
petal_length_cat[:4]
<#> ['small', 'small', 'small', 'small']
```

107.How to create a new column from existing columns of a numpy array?

Ans: Create a new column for volume in iris_2d, where volume is (pi x petallength x sepal_length^2)/3

```
# Import iris keeping the text column intact
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')

# Solution
# Get the species column
species = iris[:, 4]

# Approach 1: Generate Probablistically
np.random.seed(100)
a = np.array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
species_out = np.random.choice(a, 150, p=[0.5, 0.25, 0.25])

# Approach 2: Probablistic Sampling (preferred)
np.random.seed(100)
probs = np.r_[np.linspace(0, 0.500, num=50), np.linspace(0.501, .750, num=50), np.linspace(.751, 1.0, num=50)]
index = np.searchsorted(probs, np.random.random(150))
species_out = species[index]
print(np.unique(species_out, return_counts=True))

#> (array([b'Iris-setosa', b'Iris-versicolor', b'Iris-virginica'], dtype=object), array([77, 37, 36]))
Approach 2 is preferred because it creates an
```

---

108. How to get the second largest value of an array when grouped by another array?

Ans:

```
# Import iris keeping the text column intact
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')

# Solution
# Get the species and petal length columns
petal_len_setosa = iris[iris[:, 4] == b'Iris-setosa', [2]].astype('float')

# Get the second last value
np.unique(np.sort(petal_len_setosa))[-2]
#> 1.7
```

---

109. How to sort a 2D array by a column?

Ans: Sort the iris dataset based on sepallength column.

```
# Sort by column position 0: SepalLength
>>> print(iris[iris[:,0].argsort()][:20])
[[b'4.3' b'3.0' b'1.1' b'0.1' b'Iris-setosa']
 [b'4.4' b'3.2' b'1.3' b'0.2' b'Iris-setosa']
 [b'4.4' b'3.0' b'1.3' b'0.2' b'Iris-setosa']
 [b'4.4' b'2.9' b'1.4' b'0.2' b'Iris-setosa']
 [b'4.5' b'2.3' b'1.3' b'0.3' b'Iris-setosa']
 [b'4.6' b'3.6' b'1.0' b'0.2' b'Iris-setosa']
 [b'4.6' b'3.1' b'1.5' b'0.2' b'Iris-setosa']
 [b'4.6' b'3.4' b'1.4' b'0.3' b'Iris-setosa']
 [b'4.6' b'3.2' b'1.4' b'0.2' b'Iris-setosa']
 [b'4.7' b'3.2' b'1.3' b'0.2' b'Iris-setosa']
 [b'4.7' b'3.2' b'1.6' b'0.2' b'Iris-setosa']
 [b'4.8' b'3.0' b'1.4' b'0.1' b'Iris-setosa']
 [b'4.8' b'3.0' b'1.4' b'0.3' b'Iris-setosa']
 [b'4.8' b'3.4' b'1.9' b'0.2' b'Iris-setosa']
 [b'4.8' b'3.4' b'1.6' b'0.2' b'Iris-setosa']
 [b'4.8' b'3.1' b'1.6' b'0.2' b'Iris-setosa']
 [b'4.9' b'2.4' b'3.3' b'1.0' b'Iris-versicolor']
 [b'4.9' b'2.5' b'4.5' b'1.7' b'Iris-virginica']
 [b'4.9' b'3.1' b'1.5' b'0.1' b'Iris-setosa']
 [b'4.9' b'3.1' b'1.5' b'0.1' b'Iris-setosa']]
>>>
```

---

110. How to find the most frequent value in a numpy array?

Find the most frequent value of petal length (3rd column) in iris dataset.

```
# Input:
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')

# Solution:
vals, counts = np.unique(iris[:, 2], return_counts=True)
print(vals[np.argmax(counts)])
#> b'1.5'
```

---

111. How to find the position of the first occurrence of a value greater than a given value?

Find the position of the first occurrence of a value greater than 1.0 in petalwidth 4th column of iris dataset.

```
# Input:
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')

# Solution: (edit: changed argmax to argwhere. Thanks Rong!)
np.argwhere(iris[:, 3].astype(float) > 1.0)[0]
#> 50
```

---

112. How to replace all values greater than a given value to a given cutoff?

From the array a, replace all values greater than 30 to 30 and less than 10 to 10.

```
# Input
np.set_printoptions(precision=2)
np.random.seed(100)
a = np.random.uniform(1,50, 20)

# Solution 1: Using np.clip
np.clip(a, a_min=10, a_max=30)

# Solution 2: Using np.where
print(np.where(a < 10, 10, np.where(a > 30, 30, a)))
#> [ 27.63  14.64  21.8   30.    10.    10.    30.    30.    10.    29.18  30.
#>   11.25  10.08  10.    11.77  30.    30.    10.    30.    14.43]
```

113. How to get the positions of top n values from a numpy array?

Get the positions of top 5 maximum values in a given array a

```
# Input
np.random.seed(100)
a = np.random.uniform(1,50, 20)

# Solution:
print(a.argsort())
#> [18 7 3 10 15]

# Solution 2:
np.argpartition(-a, 5)[:5]
#> [15 10  3  7 18]

# Below methods will get you the values.
# Method 1:
a[a.argsort()][-5:]

# Method 2:
np.sort(a)[-5:]

# Method 3:
np.partition(a, kth=-5)[-5:]

# Method 4:
a[np.argpartition(-a, 5)][:5]
```

114. How to compute the row wise counts of all possible values in an array?

Compute the counts of unique values row-wise.

```
# Example 1:

# Input:
np.random.seed(100)
arr = np.random.randint(1,11,size=(6, 10))
arr
#> array([[ 9,  9,  4,  8,  8,  1,  5,  3,  6,  3],
#>        [ 3,  3,  2,  1,  9,  5,  1, 10,  7,  3],
#>        [ 5,  2,  6,  4,  5,  5,  4,  8,  2,  2],
#>        [ 8,  8,  1,  3, 10, 10,  4,  3,  6,  9],
#>        [ 2,  1,  8,  7,  3,  1,  9,  3,  6,  2],
#>        [ 9,  2,  6,  5,  3,  9,  4,  6,  1, 10]])
# Solution
def counts_of_all_values_rowwise(arr2d):
    # Unique values and its counts row wise
    num_counts_array = [np.unique(row, return_counts=True) for row in arr2d]

    # Counts of all values row wise
    return([[int(b[a==i]) if i in a else 0 for i in np.unique(arr2d)] for a, b in num_counts_array])

# Print
print(np.arange(1,11))
counts_of_all_values_rowwise(arr)
#> [ 1  2  3  4  5  6  7  8  9 10]

#> [[1, 0, 2, 1, 1, 1, 0, 2, 2, 0],
#>  [2, 1, 3, 0, 1, 0, 1, 0, 1, 1],
#>  [0, 3, 0, 2, 3, 1, 0, 1, 0, 0],
#>  [1, 0, 2, 1, 0, 1, 0, 2, 1, 2],
#>  [2, 2, 2, 0, 0, 1, 1, 1, 1, 0],
#>  [1, 1, 1, 1, 1, 2, 0, 0, 2, 1]]


# Example 2:
arr = np.array([np.array(list('bill clinton')), np.array(list('narendramodi')), np.array(list('jjayalalitha'))])
print(np.unique(arr))
```

```
counts_of_all_values_rowwise(arr)
#> [' ' 'a' 'b' 'c' 'd' 'e' 'h' 'i' 'j' 'l' 'm' 'n' 'o' 'r' 't' 'y']

#> [[1, 0, 1, 1, 0, 0, 0, 2, 0, 3, 0, 2, 1, 0, 1, 0],
#>  [0, 2, 0, 0, 2, 1, 0, 1, 0, 0, 1, 2, 1, 2, 0, 0],
#>  [0, 4, 0, 0, 0, 0, 1, 1, 2, 2, 0, 0, 0, 0, 1, 1]]
```

---

115. How to convert an array of arrays into a flat 1d array?

Convert array_of_arrays into a flat linear 1d array.

```
# Input:
arr1 = np.arange(3)
arr2 = np.arange(3,7)
arr3 = np.arange(7,10)

array_of_arrays = np.array([arr1, arr2, arr3])
print('array_of_arrays: ', array_of_arrays)

# Solution 1
arr_2d = np.array([a for arr in array_of_arrays for a in arr])

# Solution 2:
arr_2d = np.concatenate(array_of_arrays)
print(arr_2d)
#> array_of_arrays:  [array([0, 1, 2]) array([3, 4, 5, 6]) array([7, 8, 9])]
#> [0 1 2 3 4 5 6 7 8 9]
```

---

116. How to generate one-hot encodings for an array in numpy?

Compute the one-hot encodings (dummy binary variables for each unique value in the array)

```
# Input:
np.random.seed(101)
arr = np.random.randint(1,4, size=6)
arr
#> array([2, 3, 2, 2, 2, 1])

# Solution:
def one_hot_encodings(arr):
    uniqs = np.unique(arr)
    out = np.zeros((arr.shape[0], uniqs.shape[0]))
    for i, k in enumerate(arr):
        out[i, k-1] = 1
    return out

one_hot_encodings(arr)
#> array([[ 0.,  1.,  0.],
#>        [ 0.,  0.,  1.],
#>        [ 0.,  1.,  0.],
#>        [ 0.,  1.,  0.],
#>        [ 0.,  1.,  0.],
#>        [ 1.,  0.,  0.]])

# Method 2:
(arr[:, None] == np.unique(arr)).view(np.int8)
```

---

117. How to create row numbers grouped by a categorical variable?

Create row numbers grouped by a categorical variable. Use the following sample from iris species as input.

```
# Input:
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
species = np.genfromtxt(url, delimiter=',', dtype='str', usecols=4)
np.random.seed(100)
```

```
species_small = np.sort(np.random.choice(species, size=20))
species_small
#> array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
#>        'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor',
#>        'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
#>        'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
#>        'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
#>        'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
#>        'Iris-virginica'],
#>       dtype='<U15')
```

---

118. How to create ground ids based on a given categorical variable?

Create group ids based on a given categorical variable. Use the following sample from iris species as input.

```
# Input:
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
species = np.genfromtxt(url, delimiter=',', dtype='str', usecols=4)
np.random.seed(100)
species_small = np.sort(np.random.choice(species, size=20))
species_small
#> array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
#>        'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor',
#>        'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
#>        'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
#>        'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
#>        'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
#>        'Iris-virginica'],
#>       dtype='<U15')


# Solution:
output = [np.argwhere(np.unique(species_small) == s).tolist()[0][0] for val in np.unique(species_small) for s in species_small[species_small==val]]

# Solution: For Loop version
output = []
uniqs = np.unique(species_small)

for val in uniqs:  # uniq values in group
    for s in species_small[species_small==val]:  # each element in group
        groupid = np.argwhere(uniqs == s).tolist()[0][0]  # groupid
        output.append(groupid)

print(output)
#> [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2]
```

---

119. How to rank items in an array using numpy?

Create the ranks for the given numeric array a.

```
np.random.seed(10)
a = np.random.randint(20, size=10)
print('Array: ', a)

# Solution
print(a.argsort().argsort())
print('Array: ', a)
#> Array:  [ 9  4 15  0 17 16 17  8  9  0]
#> [4 2 6 0 8 7 9 3 5 1]
#> Array:  [ 9  4 15  0 17 16 17  8  9  0]
```

---

120. How to rank items in a multidimensional array using numpy?

Create a rank array of the same shape as a given numeric array a.

```
# Input:
np.random.seed(10)
```

```
a = np.random.randint(20, size=[2,5])
print(a)

# Solution
print(a.ravel().argsort().argsort().reshape(a.shape))
#> [[ 9  4 15  0 17]
#>  [16 17  8  9  0]]
#> [[4 2 6 0 8]
#>  [7 9 3 5 1]]
```

---

121. How to find the maximum value in each row of a numpy array 2d?

Compute the maximum for each row in the given array

```
# Input
np.random.seed(100)
a = np.random.randint(1,10, [5,3])
a

# Solution 1
np.amax(a, axis=1)

# Solution 2
np.apply_along_axis(np.max, arr=a, axis=1)
#> array([9, 8, 6, 3, 9])
```

---

122. How to compute the min-by-max for each row for a numpy array 2d?

Compute the min-by-max for each row for given 2d numpy array.

```
# Input
np.random.seed(100)
a = np.random.randint(1,10, [5,3])
a

# Solution
np.apply_along_axis(lambda x: np.min(x)/np.max(x), arr=a, axis=1)
#> array([ 0.44444444,  0.125     ,  0.5       ,  1.        ,  0.11111111])
```

---

123. How to find the duplicate records in a numpy array?

Find the duplicate entries (2nd occurrence onwards) in the given numpy array and mark them as True. First time occurrences should be False

```
# Input
np.random.seed(100)
a = np.random.randint(0, 5, 10)

## Solution
# There is no direct function to do this as of 1.13.3

# Create an all True array
out = np.full(a.shape[0], True)

# Find the index positions of unique elements
unique_positions = np.unique(a, return_index=True)[1]

# Mark those positions as False
out[unique_positions] = False

print(out)
#> [False  True False  True False False  True  True  True  True]
```

---

124. How to find the grouped mean in numpy?

Find the mean of a numeric column grouped by a categorical column in a 2D numpy array

```
# Input
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = np.genfromtxt(url, delimiter=',', dtype='object')
names = ('sepallength', 'sepalwidth', 'petallength', 'petalwidth', 'species')


# Solution
# No direct way to implement this. Just a version of a workaround.
numeric_column = iris[:, 1].astype('float')  # sepalwidth
grouping_column = iris[:, 4]  # species

# List comprehension version
[[group_val, numeric_column[grouping_column==group_val].mean()] for group_val in
np.unique(grouping_column)]

# For Loop version
output = []
for group_val in np.unique(grouping_column):
    output.append([group_val, numeric_column[grouping_column==group_val].mean()])

output
#> [[b'Iris-setosa', 3.418],
#>  [b'Iris-versicolor', 2.770],
#>  [b'Iris-virginica', 2.974]]
```

125.How to convert a PIL image to numpy?

.Import the image from the following URL and convert it to a numpy array.

```
from io import BytesIO
from PIL import Image
import PIL, requests

# Import image from URL
URL = 'https://upload.wikimedia.org/wikipedia/commons/8/8b/Denali_Mt_McKinley.jpg'
response = requests.get(URL)

# Read it as Image
I = Image.open(BytesIO(response.content))

# Optionally resize
I = I.resize([150,150])

# Convert to numpy array
arr = np.asarray(I)

# Optionaly Convert it back to an image and show
im = PIL.Image.fromarray(np.uint8(arr))
Image.Image.show(im)
```

126.How to drop all missing values from a numpy array?

Drop all nan values from a 1D numpy array

```
a = np.array([1,2,3,np.nan,5,6,7,np.nan])
a[~np.isnan(a)]
#> array([ 1.,  2.,  3.,  5.,  6.,  7.])
```

---

127. How to compute the euclidean distance between two arrays?

Compute the euclidean distance between two arrays a and b.

```
# Input
a = np.array([1,2,3,4,5])
b = np.array([4,5,6,7,8])

# Solution
dist = np.linalg.norm(a-b)
```

```
dist
#> 6.7082039324993694
```

128.How to find all the local maxima (or peaks) in a 1d array?

Find all the peaks in a 1D numpy array a. Peaks are points surrounded by smaller values on both sides.

```
a = np.array([1, 3, 7, 1, 2, 6, 0, 1])
doublediff = np.diff(np.sign(np.diff(a)))
peak_locations = np.where(doublediff == -2)[0] + 1
peak_locations
#> array([2, 5])
```

---

129. How to subtract a 1d array from a 2d array, where each item of 1d array subtracts from respective row?

Subtract the 1d array b_1d from the 2d array a_2d, such that each item of b_1d subtracts from the respective row of a_2d.

```
# Input
a_2d = np.array([[3,3,3],[4,4,4],[5,5,5]])
b_1d = np.array([1,2,3])

# Solution
print(a_2d - b_1d[:,None])
#> [[2 2 2]
#>  [2 2 2]
#>  [2 2 2]]
```

---

130. How to find the index of n’th repetition of an item in an array

Find the index of 5th repetition of number 1 in x.

```
x = np.array([1, 2, 1, 1, 3, 4, 3, 1, 1, 2, 1, 1, 2])
n = 5

# Solution 1: List comprehension
[i for i, v in enumerate(x) if v == 1][n-1]

# Solution 2: Numpy version
np.where(x == 1)[0][n-1]
#> 8
```

---

131. How to convert numpy’s datetime64 object to datetime’s datetime object?

Convert numpy’s datetime64 object to datetime’s datetime object

```
# Input: a numpy datetime64 object
dt64 = np.datetime64('2018-02-25 22:10:10')

# Solution
from datetime import datetime
dt64.tolist()

# or

dt64.astype(datetime)
#> datetime.datetime(2018, 2, 25, 22, 10, 10)
```

---

131. How to compute the moving average of a numpy array?

Compute the moving average of window size 3, for the given 1D array.

```
# Solution
# Source: https://stackoverflow.com/questions/14313510/how-to-calculate-moving-average-using-numpy
def moving_average(a, n=3) :
    ret = np.cumsum(a, dtype=float)
    ret[n:] = ret[n:] - ret[:-n]
    return ret[n - 1:] / n

np.random.seed(100)
Z = np.random.randint(10, size=10)
print('array: ', Z)
# Method 1
moving_average(Z, n=3).round(2)

# Method 2:  # Thanks AlanLRH!
# np.ones(3)/3 gives equal weights. Use np.ones(4)/4 for window size 4.
np.convolve(Z, np.ones(3)/3, mode='valid') .


#> array:  [8 8 3 7 7 0 4 2 5 2]
#> moving average:  [ 6.33  6.    5.67  4.67  3.67  2.    3.67  3.  ]
```

---

132. How to create a numpy array sequence given only the starting point, length and the step?

Create a numpy array of length 10, starting from 5 and has a step of 3 between consecutive numbers

```
length = 10
start = 5
step = 3

def seq(start, length, step):
    end = start + (step*length)
    return np.arange(start, end, step)

seq(start, length, step)
#> array([ 5,  8, 11, 14, 17, 20, 23, 26, 29, 32])
```

---

133. How to fill in missing dates in an irregular series of numpy dates?

Given an array of a non-continuous sequence of dates. Make it a continuous sequence of dates, by filling in the missing dates.

```
# Input
dates = np.arange(np.datetime64('2018-02-01'), np.datetime64('2018-02-25'), 2)
print(dates)

# Solution ----------------
filled_in = np.array([np.arange(date, (date+d)) for date, d in zip(dates, np.diff(dates))]).reshape(-1)

# add the last day
output = np.hstack([filled_in, dates[-1]])
output

# For loop version -------
out = []
for date, d in zip(dates, np.diff(dates)):
    out.append(np.arange(date, (date+d)))

filled_in = np.array(out).reshape(-1)

# add the last day
output = np.hstack([filled_in, dates[-1]])
output
#> ['2018-02-01' '2018-02-03' '2018-02-05' '2018-02-07' '2018-02-09'
#>  '2018-02-11' '2018-02-13' '2018-02-15' '2018-02-17' '2018-02-19'
#>  '2018-02-21' '2018-02-23']

#> array(['2018-02-01', '2018-02-02', '2018-02-03', '2018-02-04',
```

```
#>       '2018-02-05', '2018-02-06', '2018-02-07', '2018-02-08',
#>       '2018-02-09', '2018-02-10', '2018-02-11', '2018-02-12',
#>       '2018-02-13', '2018-02-14', '2018-02-15', '2018-02-16',
#>       '2018-02-17', '2018-02-18', '2018-02-19', '2018-02-20',
#>       '2018-02-21', '2018-02-22', '2018-02-23'], dtype='datetime64[D]')
```

---

134. How to create strides from a given 1D array?

From the given 1d array arr, generate a 2d matrix using strides, with a window length of 4 and strides of 2, like [[0,1,2,3], [2,3,4,5], [4,5,6,7]..]

```
def gen_strides(a, stride_len=5, window_len=5):
    n_strides = ((a.size-window_len)//stride_len) + 1
    # return np.array([a[s:(s+window_len)] for s in np.arange(0, a.size, stride_len)[:n_strides]])
    return np.array([a[s:(s+window_len)] for s in np.arange(0, n_strides*stride_len, stride_len)])

print(gen_strides(np.arange(15), stride_len=2, window_len=4))
#> [[ 0  1  2  3]
#>  [ 2  3  4  5]
#>  [ 4  5  6  7]
#>  [ 6  7  8  9]
#>  [ 8  9 10 11]
#>  [10 11 12 13]]
```

---

135. Create a 4X2 integer array and Prints its attributes

The element must be a type of unsigned int16. And print the following Attributes: –

- The shape of an array.
- Array dimensions.
- The Length of each element of the array in bytes.

```
import numpy

firstArray = numpy.empty([4,2], dtype = numpy.uint16)
print("Printing Array")
print(firstArray)

print("Printing numpy array Attributes")
print("1> Array Shape is: ", firstArray.shape)
print("2>. Array dimensions are ", firstArray.ndim)
print("3>. Length of each element of array in bytes is ", firstArray.itemsize)
```

---

136. Create a 5X2 integer array from a range between 100 to 200 such that the difference between each element is 10

```
import numpy

print("Creating 5X2 array using numpy.arange")
sampleArray = numpy.arange(100, 200, 10)
sampleArray = sampleArray.reshape(5,2)
print (sampleArray)
```

---

137. Following is the provided numpy array. return array of items in the third column from all rows

```
import numpy
sampleArray = numpy.array([[11 ,22, 33], [44, 55, 66], [77, 88, 99]])


Expected Output:
Printing Input Array
[[11 22 33]
 [44 55 66]
 [77 88 99]]

 Printing array of items in the third column from all rows
[22 55 88]
```

```
import numpy

sampleArray = numpy.array([[11 ,22, 33], [44, 55, 66], [77, 88, 99]])
print("Printing Input Array")
print(sampleArray)

print("\n Printing array of items in the third column from all rows")
newArray = sampleArray[...,1]
print(newArray)
```

---

138. Following is the given numpy array return array of odd rows and even columns

```
import numpy

sampleArray = numpy.array([[3 ,6, 9, 12], [15 ,18, 21, 24],
[27 ,30, 33, 36], [39 ,42, 45, 48], [51 ,54, 57, 60]])
Expected Output:

Printing Input Array
[[ 3  6  9 12]
 [15 18 21 24]
 [27 30 33 36]
 [39 42 45 48]
 [51 54 57 60]]

 Printing array of odd rows and even columns
[[ 6 12]
 [30 36]
 [54 60]]


import numpy

sampleArray = numpy.array([[3 ,6, 9, 12], [15 ,18, 21, 24],
[27 ,30, 33, 36], [39 ,42, 45, 48], [51 ,54, 57, 60]])
print("Printing Input Array")
print(sampleArray)

print("\n Printing array of odd rows and even columns")
newArray = sampleArray[::2, 1::2]
print(newArray)
```

---

139. Add the following two NumPy arrays and Modify a result array by calculating the square of each element

```
import numpy

arrayOne = numpy.array([[5, 6, 9], [21 ,18, 27]])
arrayTwo = numpy.array([[15 ,33, 24], [4 ,7, 1]])
Expected Output:

addition of two arrays is

[[20 39 33]
 [25 25 28]]

Result array after calculating the square root of all elements

[[ 400 1521 1089]
 [ 625  625  784]]
import numpy

arrayOne = numpy.array([[5, 6, 9], [21 ,18, 27]])
arrayTwo = numpy.array([[15 ,33, 24], [4 ,7, 1]])

resultArray  = arrayOne + arrayTwo
```

```
print("addition of two arrays is \n")
print(resultArray)

for num in numpy.nditer(resultArray, op_flags = ['readwrite']):
   num[...] = num*num
print("\nResult array after calculating the square root of all elements\n")
print(resultArray)
```

---

140. Split the array into four equal-sized sub-arrays

Note: Create an 8X3 integer array from a range between 10 to 34 such that the difference between each element is 1 and then Split the array into four equal-sized sub-arrays.
Expected Output:

```
Creating 8X3 array using numpy.arange
[[10 11 12]
 [13 14 15]
 [16 17 18]
 [19 20 21]
 [22 23 24]
 [25 26 27]
 [28 29 30]
 [31 32 33]]

Dividing 8X3 array into 4 sub array

[array([[10, 11, 12],[13, 14, 15]]),
array([[16, 17, 18],[19, 20, 21]]),
array([[22, 23, 24],[25, 26, 27]]),
array([[28, 29, 30],[31, 32, 33]])]
```

```
import numpy

print("Creating 8X3 array using numpy.arange")
sampleArray = numpy.arange(10, 34, 1)
sampleArray = sampleArray.reshape(8,3)
print (sampleArray)

print("\nDividing 8X3 array into 4 sub array\n")
subArrays = numpy.split(sampleArray, 4)
print(subArrays)
```

---

141. Sort following NumPy array

- 7.1- by the second row and
- 7.2-by the second column

```
import numpy
sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])
```

Expected Output:

```
Printing Original array
[[34 43 73]
 [82 22 12]
 [53 94 66]]

Sorting Original array by secoond row
[[73 43 34]
 [12 22 82]
 [66 94 53]]

Sorting Original array by secoond column
[[82 22 12]
 [34 43 73]
 [53 94 66]]
```

```
import numpy

print("Printing Original array")
sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])
print (sampleArray)
```

```
sortArrayByRow = sampleArray[:,sampleArray[1,:].argsort()]
print("Sorting Original array by secoond row")
print(sortArrayByRow)

print("Sorting Original array by secoond column")
sortArrayByColumn = sampleArray[sampleArray[:,1].argsort()]
print(sortArrayByColumn)
```

---

142. Following is the 2-D array. Print max from axis 0 and min from axis 1

```
import numpy
sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])
Expected Output:
Printing Original array
[[34 43 73]
 [82 22 12]
 [53 94 66]]
Printing amin Of Axis 1
[34 12 53]
Printing amax Of Axis 0
[82 94 73]


import numpy

print("Printing Original array")
sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])
print (sampleArray)


minOfAxisOne = numpy.amin(sampleArray, 1)
print("Printing amin Of Axis 1")
print(minOfAxisOne)

maxOfAxisOne = numpy.amax(sampleArray, 0)
print("Printing amax Of Axis 0")
print(maxOfAxisOne)
```

---

143. Following is the input NumPy array delete column two and insert following new column in its place.

```
import numpy
sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])

newColumn = numpy.array([[10,10,10]])
Expected Output:

Printing Original array
[[34 43 73]
 [82 22 12]
 [53 94 66]]

Array after deleting column 2 on axis 1

[[34 73]
 [82 12]
 [53 66]]
Array after inserting column 2 on axis 1

[[34 10 73]
 [82 10 12]
 [53 10 66]]
Solution:
import numpy

print("Printing Original array")
sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])
print (sampleArray)

print("Array after deleting column 2 on axis 1")
```

```
sampleArray = numpy.delete(sampleArray , 1, axis = 1)
print (sampleArray)

arr = numpy.array([[10,10,10]])

print("Array after inserting column 2 on axis 1")
sampleArray = numpy.insert(sampleArray , 1, arr, axis = 1)
print (sampleArray)
```

---

144. Create a two 2-D array and Plot it using matplotlib

```
import numpy

print("Printing Original array")
sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])
print (sampleArray)

print("Array after deleting column 2 on axis 1")
sampleArray = numpy.delete(sampleArray , 1, axis = 1)
print (sampleArray)

arr = numpy.array([[10,10,10]])

print("Array after inserting column 2 on axis 1")
sampleArray = numpy.insert(sampleArray , 1, arr, axis = 1)
print (sampleArray)
```