```
import numpy as np
```

Q-1: Create a random 3x4 matrix with value between 0-100. And perform below tasks

i. Sort this matrix. np.sort()
ii. Sort this matrix based on values in 2nd column.
iii. Sort this matrix based on max value in each row.
iv. Sort based on elements value.

See examples:

```
arr =
    [[92 90 74]
    [ 6 63 93]
    [15 93 96]
    [70 60 48]]


i. np.sort
[[74 90 92]
 [ 6 63 93]
 [15 93 96]
 [48 60 70]]

ii. based on 2nd column
[[70 60 48]
 [ 6 63 93]
 [92 90 74]
 [15 93 96]]

iii. based on row max- ascending
[[15 93 96]
 [ 6 63 93]
 [92 90 74]
 [70 60 48]]

iv. based on elements value
[[ 6 15 48]
 [60 63 70]
 [74 90 92]
 [93 93 96]]
```

```
# code here
arr = np.random.randint(0, 100, (4,3))
print(arr)
#1
print(np.sort(arr))
#2
print(arr[arr[:, 1].argsort()])
#3
print(np.array(sorted(arr, key = lambda x: max(x))))
#4
print(np.sort(arr, axis = None).reshape(4,3))
```

```
    [[94 87 26]
     [45 73 90]
     [70 30 81]
     [97  7 83]]
    [[26 87 94]
     [45 73 90]
     [30 70 81]
     [ 7 83 97]]
    [[97  7 83]
     [70 30 81]
```

```
       [45 73 90]
       [94 87 26]]
      [[70 30 81]
       [45 73 90]
       [94 87 26]
       [97  7 83]]
      [[ 7 26 30]
       [45 70 73]
       [81 83 87]
       [90 94 97]]
```

Q-2: There is an array of marks of 5 students in 4 subjects. Further you are asked to perform below task.

i. Add marks every student of an extra subject in the same array.
ii. Add two new students marks in respective 5 subjects.(one subject added in above task)
iii. Add extra column with sum of all subjects(5-subjects) marks
iv. Sort the array(non-ascending order) on total marks column--one added in above task. Show top 2 rows.

Note: Change dimension of arrays during concatenation or appending if required.

Given Array-

```
marks = [[13, 10,  9, 33],
       [63, 46, 90, 42],
       [39, 76, 13, 29],
       [82,  9, 29, 78],
       [67, 61, 59, 36]]

extra_subject = [41, 87, 72, 36, 92]
# Two extra students record-
rec1 = [77, 83, 98, 95, 89]
rec2 = [92, 71, 52, 61, 53]
```

```
# code here
marks = np.array([[13, 10,  9, 33],
       [63, 46, 90, 42],
       [39, 76, 13, 29],
       [82,  9, 29, 78],
       [67, 61, 59, 36]])
extra_subject = np.array([41, 87, 72, 36, 92]).reshape(-1, 1)
marks = np.concatenate([marks, extra_subject], axis = 1)
#2
rec1 = np.array([77, 83, 98, 95, 89]).reshape(1, -1)
rec2 = np.array( [92, 71, 52, 61, 53]).reshape(1, -1)
marks = np.concatenate([marks, rec1, rec2], axis = 0)
#3
sum_of_stud = marks.sum(axis = 1, keepdims=True)
marks = np.concatenate([marks, sum_of_stud], axis = 1)
#4
print(np.array(sorted(marks, key = lambda x: x[-1], reverse=True)))
```

```
     [[ 77  83  98  95  89 442]
      [ 92  71  52  61  53 329]
      [ 63  46  90  42  87 328]
      [ 67  61  59  36  92 315]
      [ 82   9  29  78  36 234]
      [ 39  76  13  29  72 229]
      [ 13  10   9  33  41 106]]
```

Q-3: Find unique arrays from a 2D array column wise and row wise.

```
arr = np.array([[1,2,3,3,1,1],
                [0,9,1,2,8,8],
                [1,2,3,8,8,8],
                [1,2,3,3,1,1]])
```

Expected Result-

```
Row Wise
[[0 9 1 2 8 8]
 [1 2 3 3 1 1]
 [1 2 3 8 8 8]]

Col Wise
[[1 1 2 3 3]
 [0 8 9 1 2]
 [1 8 2 3 8]
 [1 1 2 3 3]]
```

```
# code here
arr = np.array([[1,2,3,3,1,1],
                [0,9,1,2,8,8],
                [1,2,3,8,8,8],
                [1,2,3,3,1,1]])
#1
print(np.unique(arr, axis = 0))
#2
print(np.unique(arr, axis = 1))
```

```
[[0 9 1 2 8 8]
 [1 2 3 3 1 1]
 [1 2 3 8 8 8]]
[[1 1 2 3 3]
 [0 8 9 1 2]
 [1 8 2 3 8]
 [1 1 2 3 3]]
```

Q-4: Flip given 2-D array along both axes at the same time.

```
# code here
arr = np.array([[1,2,3,3,1,1],
                [0,9,1,2,8,8],
                [1,2,3,8,8,8],
                [1,2,3,3,1,1],
                [0,6,1,2,11,7],
                [10,19,14,-2,8,-1]])
np.flip(arr, axis = [0, 1])
```

```
array([[-1,  8, -2, 14, 19, 10],
       [ 7, 11,  2,  1,  6,  0],
       [ 1,  1,  3,  3,  2,  1],
       [ 8,  8,  8,  3,  2,  1],
       [ 8,  8,  2,  1,  9,  0],
       [ 1,  1,  3,  3,  2,  1]])
```

Q-5: Get row numbers of NumPy array having element larger than X.

```
arr = [[1,2,3,4,5],
       [10,-3,30,4,5],
       [3,2,5,-4,5],
       [9,7,3,6,5]]

X = 6
```

```
# code here
arr = np.array([[1,2,3,4,5],
      [10,-3,30,4,5],
      [3,2,5,-4,5],
      [9,7,3,6,5]] )
X = 6

np.where(np.any(arr > X, axis = 1))
```

```
(array([1, 3], dtype=int64),)
```

Q-6: How to convert an array of arrays into a flat 1d array?

```
# These arrays are given.
arr1 = np.arange(3)
arr2 = np.arange(3,7)
arr3 = np.arange(7,10)
```

```
# code here
np.concatenate([arr1, arr2, arr3])
```

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

Q-7: You are given a array. You have to find the minimum and maximum array element and remove that from the array.

```
import numpy as np

np.random.seed(400)
arr = np.random.randint(100, 1000, 200).reshape((1, 200))
```

```
# code here
np.random.seed(400)
arr = np.random.randint(100, 1000, 200).reshape((1, 200))
print(arr)
max_element = arr.argmax()
min_element = arr.argmin()
print(arr.max(), arr.min())
arr = np.delete(arr, max_element, axis = 1)
arr = np.delete(arr, min_element, axis = 1)
print(arr.max(), arr.min())
```

```
[[448 563 418 240 507 362 345 236 719 291 298 639 458 387 262 613 267 882
  181 425 790 635 889 818 872 967 277 470 336 920 917 295 557 830 506 385
  353 975 592 997 137 340 222 215 472 459 617 649 935 956 914 932 645 952
  921 490 527 972 278 307 840 958 246 449 251 957 103 627 920 824 356 825
  173 323 372 960 710 464 244 782 763 635 436 774 171 469 178 458 624 211
  771 270 308 231 952 514 699 702 433 900 373 318 998 265 503 320 230 324
  922 967 620 743 527 117 566 804 123 946 587 227 853 757 944 328 855 930
  325 729 426 514 296 879 575 936 705 209 191 743 510 513 628 559 658 528
  395 525 922 136 496 225 895 975 263 908 420 711 800 976 786 235 930 859
  618 226 695 460 218 483 490 803 621 453 193 607 677 637 728 724 534 748
  291 194 761 875 687 569 228 482 781 554 654 739 885 197 266 228 892 207
  883 588]]
998 103
997 117
```

Q-8: You are given an arrays. You have to limit this array's elements between 100 to 200. $arr \in [100, 200]$. So replace those values accordingly with the minimum and maximum value. Then sort the array and perform the cumulative sum of that array.

```
# code here
arr = np.random.randint(0, 2000, 500)
np.cumsum(np.sort(np.clip(arr, 100, 200)))
```

```
array([  100,   200,   300,   400,   500,   600,   700,   800,   900,
        1000,  1100,  1200,  1300,  1400,  1500,  1600,  1700,  1800,
        1900,  2000,  2100,  2205,  2320,  2436,  2553,  2677,  2811,
        2947,  3094,  3242,  3395,  3552,  3712,  3873,  4044,  4221,
        4399,  4592,  4786,  4986,  5186,  5386,  5586,  5786,  5986,
        6186,  6386,  6586,  6786,  6986,  7186,  7386,  7586,  7786,
        7986,  8186,  8386,  8586,  8786,  8986,  9186,  9386,  9586,
        9786,  9986, 10186, 10386, 10586, 10786, 10986, 11186, 11386,
       11586, 11786, 11986, 12186, 12386, 12586, 12786, 12986, 13186,
       13386, 13586, 13786, 13986, 14186, 14386, 14586, 14786, 14986,
       15186, 15386, 15586, 15786, 15986, 16186, 16386, 16586, 16786,
```

```
       16986, 17186, 17386, 17586, 17786, 17986, 18186, 18386, 18586,
       18786, 18986, 19186, 19386, 19586, 19786, 19986, 20186, 20386,
       20586, 20786, 20986, 21186, 21386, 21586, 21786, 21986, 22186,
       22386, 22586, 22786, 22986, 23186, 23386, 23586, 23786, 23986,
       24186, 24386, 24586, 24786, 24986, 25186, 25386, 25586, 25786,
       25986, 26186, 26386, 26586, 26786, 26986, 27186, 27386, 27586,
       27786, 27986, 28186, 28386, 28586, 28786, 28986, 29186, 29386,
       29586, 29786, 29986, 30186, 30386, 30586, 30786, 30986, 31186,
       31386, 31586, 31786, 31986, 32186, 32386, 32586, 32786, 32986,
       33186, 33386, 33586, 33786, 33986, 34186, 34386, 34586, 34786,
       34986, 35186, 35386, 35586, 35786, 35986, 36186, 36386, 36586,
       36786, 36986, 37186, 37386, 37586, 37786, 37986, 38186, 38386,
       38586, 38786, 38986, 39186, 39386, 39586, 39786, 39986, 40186,
       40386, 40586, 40786, 40986, 41186, 41386, 41586, 41786, 41986,
       42186, 42386, 42586, 42786, 42986, 43186, 43386, 43586, 43786,
       43986, 44186, 44386, 44586, 44786, 44986, 45186, 45386, 45586,
       45786, 45986, 46186, 46386, 46586, 46786, 46986, 47186, 47386,
       47586, 47786, 47986, 48186, 48386, 48586, 48786, 48986, 49186,
       49386, 49586, 49786, 49986, 50186, 50386, 50586, 50786, 50986,
       51186, 51386, 51586, 51786, 51986, 52186, 52386, 52586, 52786,
       52986, 53186, 53386, 53586, 53786, 53986, 54186, 54386, 54586,
       54786, 54986, 55186, 55386, 55586, 55786, 55986, 56186, 56386,
       56586, 56786, 56986, 57186, 57386, 57586, 57786, 57986, 58186,
       58386, 58586, 58786, 58986, 59186, 59386, 59586, 59786, 59986,
       60186, 60386, 60586, 60786, 60986, 61186, 61386, 61586, 61786,
       61986, 62186, 62386, 62586, 62786, 62986, 63186, 63386, 63586,
       63786, 63986, 64186, 64386, 64586, 64786, 64986, 65186, 65386,
       65586, 65786, 65986, 66186, 66386, 66586, 66786, 66986, 67186,
       67386, 67586, 67786, 67986, 68186, 68386, 68586, 68786, 68986,
       69186, 69386, 69586, 69786, 69986, 70186, 70386, 70586, 70786,
       70986, 71186, 71386, 71586, 71786, 71986, 72186, 72386, 72586,
       72786, 72986, 73186, 73386, 73586, 73786, 73986, 74186, 74386,
       74586, 74786, 74986, 75186, 75386, 75586, 75786, 75986, 76186,
       76386, 76586, 76786, 76986, 77186, 77386, 77586, 77786, 77986,
       78186, 78386, 78586, 78786, 78986, 79186, 79386, 79586, 79786,
       79986, 80186, 80386, 80586, 80786, 80986, 81186, 81386, 81586,
       81786, 81986, 82186, 82386, 82586, 82786, 82986, 83186, 83386,
       83586, 83786, 83986, 84186, 84386, 84586, 84786, 84986, 85186,
       85386, 85586, 85786, 85986, 86186, 86386, 86586, 86786, 86986,
       87186, 87386, 87586, 87786, 87986, 88186, 88386, 88586, 88786,
       88986, 89186, 89386, 89586, 89786, 89986, 90186, 90386, 90586,
       90786, 90986, 91186, 91386, 91586, 91786, 91986, 92186, 92386,
       92586, 92786, 92986, 93186, 93386, 93586, 93786, 93986, 94186,
       94386, 94586, 94786, 94986, 95186, 95386, 95586, 95786, 95986,
       96186, 96386, 96586, 96786, 96986], dtype=int32)
```

Q-9: You are given a array ($arr \in [0, 1]$). First you have round off the elements upto 3 decimal places and compare that

- 0th percentile == minimum value of the array
- 100th percentile == maximum value of the array
- also find the difference betwen 51th percenile and 50th percentile values

```
# code here
arr = np.random.random(100)
arr = np.around(arr, decimals = 3)

print("0th percentile == minimum value of the array: ", (np.percentile(arr, 0) == arr.min()))
print("100th percentile == maximum value of the array: ", (np.percentile(arr, 100) == arr.max()))
print("difference betwen 51th percenile and 50th percentile values: ", (np.percentile(arr, 51) - np.percentile(arr, 50)))
```

```
0th percentile == minimum value of the array:  True
100th percentile == maximum value of the array:  True
difference betwen 51th percenile and 50th percentile values:  0.014919999999999933
```

Start coding or generate with AI.