```
import numpy as np
```

Generating Dataset

Our dataset will be page view dataset where column represents the days (DAY 1 to DAY 5), and the row represents the students who visits the pages. (Students array will be added later.)

TASK 1 [+4].

Generate a random array with values ranging between 0-1000. The array should have 10 rows and 5 columns.

Use the seed of 5 so that it will always generate the same set of random numbers.

```
#expected outcome is given below

     array([[867, 206, 701, 998, 118],
            [400,  73,   8, 740, 743],
            [958, 411, 624, 670, 720],
            [519, 204, 911, 113, 437],
            [999,  80,  27,  44, 205],
            [715,  65, 431, 542, 980],
            [ 86, 637, 146, 254, 377],
            [649, 362,  41, 446, 897],
            [210, 103, 144, 974,   5],
            [826, 768, 976, 900, 110]])
```

```
#add your code here

import numpy as np

np.random.seed(5)

array = np.random.randint(1000, size = (10, 5))

print(array)
```

```
     [[867 206 701 998 118]
      [400  73   8 740 743]
      [958 411 624 670 720]
      [519 204 911 113 437]
      [999  80  27  44 205]
      [715  65 431 542 980]
      [ 86 637 146 254 377]
      [649 362  41 446 897]
      [210 103 144 974   5]
      [826 768 976 900 110]]
```

Slicing

In the following four task, you will apply slicing/indexing on the dataset.

TASK 2 [+4].

Select first 5 rows:

```
#expected outcome is given below

     array([[867, 206, 701, 998, 118],
            [400,  73,   8, 740, 743],
            [958, 411, 624, 670, 720],
            [519, 204, 911, 113, 437],
            [999,  80,  27,  44, 205]])
```

```
#add your code here

slicedArray = array[0:5]
print(slicedArray)
```

```
     [[867 206 701 998 118]
      [400  73   8 740 743]
      [958 411 624 670 720]
      [519 204 911 113 437]
      [999  80  27  44 205]]
```

TASK 3 [+4].

Select last 2 columns:

```
#expected outcome is given below

     array([[998, 118],
            [740, 743],
            [670, 720],
            [113, 437],
            [ 44, 205],
            [542, 980],
            [254, 377],
            [446, 897],
            [974,   5],
            [900, 110]])
```

```
#add your code here

slicedArray2 = array[:, -2:]
print(slicedArray2)
```

```
     [[998 118]
      [740 743]
      [670 720]
      [113 437]
      [ 44 205]
      [542 980]
      [254 377]
      [446 897]
      [974   5]
      [900 110]]
```

TASK 4 [+4].

Select the 3rd and 4th row, and 2nd and 3rd columns:

```
#expected outcome is given below

     array([[411, 624],
            [204, 911]])
```

```
#add your code here

slicedArray3 = array[2:4, 1:3]
print(slicedArray3)
```

```
     [[411 624]
      [204 911]]
```

Boolean Indexing

Assume that the rows in `pageViews` array correspond to specific students given in the following array:

```
#expected outcome is given below

     array(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'], dtype='<U1')
```

```
#add your code here

students = np.array(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'])
print(students)
```

```
     ['A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J']
```

TASK 5 [+6].

Find the visits of students D and H:

```
#expected outcome is given below
```

```
#add your code here

print(array[(students == 'D') | (students == 'H')])
```

```
[[519 204 911 113 437]
 [649 362  41 446 897]]
```

TASK 6 [+8].

Find the visits of students B and F during the first three days:

```
#expected outcome is given below
```

```
array([[400,  73,   8],
       [715,  65, 431]])
```

```
#add your code here


print(array[((students == 'B') | (students == 'F'))][:, :3])
```

```
[[400  73   8]
 [715  65 431]]
```

TASK 7 [+10].

Find all students' visits except student C and H:

You must use `~` *for this task.*

```
#expected outcome is given below
```

```
array([[867, 206, 701, 998, 118],
       [400,  73,   8, 740, 743],
       [519, 204, 911, 113, 437],
       [999,  80,  27,  44, 205],
       [715,  65, 431, 542, 980],
       [ 86, 637, 146, 254, 377],
       [210, 103, 144, 974,   5],
       [826, 768, 976, 900, 110]])
```

```
#add your code here

print(array[~((students == 'C') | (students == 'H'))])
```

```
[[867 206 701 998 118]
 [400  73   8 740 743]
 [519 204 911 113 437]
 [999  80  27  44 205]
 [715  65 431 542 980]
 [ 86 637 146 254 377]
 [210 103 144 974   5]
 [826 768 976 900 110]]
```

Calculating Basic Stats

TASK 8 [+5].

Compute the daily average across all students. Here you can use **np.average** . Be careful with the axis value.

```
#expected outcome is given below
```

```
array([622.9, 290.9, 400.9, 568.1, 459.2])
```

```
#add your code here
daily_average = np.average(array, axis=0) #I wrote 0 here to calculate rows

print(daily_average)
```

```
[622.9 290.9 400.9 568.1 459.2]
```

TASK 9 [+5].

Compute the average per student across all days. Here you can use **np.average** . Be careful with the axis value.

```
#expected outcome is given below
```

```
    array([578. , 392.8, 676.6, 436.8, 271. , 546.6, 300. , 479. , 287.2,
           716. ])
```

```
#add your code here
average_per_student = np.average(array, axis=1) #I wrote 0 here to calculate columns

print(average_per_student)
```

```
    [578.  392.8 676.6 436.8 271.  546.6 300.  479.  287.2 716. ]
```

TASK 10 [+15].

Return the name of the student (which is a letter) who have the lowest average page views.

Here you MUST use the argmin function. The value returned by argmin can be used as an index for `students` list to obtain the final result.

```
#expected outcome is given below
```

```
    'E'
```

```
#add your code here



# Find the index of the student with the lowest average page views using argmin
lowest_avg_index = np.argmin(average_per_student)

# Get the name of the student with the lowest average page views
lowest_avg_student_name = students[lowest_avg_index]

# Print the name of the student with the lowest average page views
print("The student with the lowest average page views is:", lowest_avg_student_name)
```

```
    The student with the lowest average page views is: E
```

TASK 11 [+15].

Find the number of days student A visited more pages than student D.

Hint: Generate a boolean list at the end, and apply `sum()` on this boolean list to obtain the result.

```
#expected outcome is given below
```

```
    3
```

```
#add your code here

# Sample lists representing the number of pages visited by students A and D
student_A_pages = array[students == 'A']
student_D_pages = array[students == 'D']

# Create a boolean list to check if A visited more pages than D on each day
more_pages_than_D = [a > d for a, d in zip(student_A_pages, student_D_pages)]

# Use the sum() function to count the number of True values in the boolean list
days_visited_more = sum(more_pages_than_D)

# Print the result
print("Student A visited more pages than student D on", days_visited_more, "days.")
```

```
    Student A visited more pages than student D on [1 1 0 1 0] days.
```

TASK 12 [+20].

Return the list of students whose total visits until the 4th day (that is 1st, 2nd and 3rd days) is less than their visits on the 4th day.

You **MUST** use `cumsum` to obtain the total visits till the 4th day.

```
#expected outcome is given below
```

```
     array(['B', 'I'], dtype='<U1')
```

```
#add your code here


# Calculate the cumulative sum of visits until the 4th day
sum_cumulative = np.cumsum(array[:, 0:3], axis = 1)
total_until_fourthday = sum_cumulative[:, 2]
fourth_day = array[:, 3]


weird_students = [students[i] for i in range(len(fourth_day)) if fourth_day[i] > total_until_fourthday[i]]
print(weird_students) # :)
```

```
     ['B', 'I']
```