In [ ]:
```
import pandas as pd
import numpy as np
```

Write a code to generate the following series

```
[ 0  7 14 21 28 35 42 49 56 63 70 77 84 91 98]

Generate the random number that contains multiples of 7
[[56 56 56 28]
 [21 21 98 98]
 [28 98 91 49]]
```

In [ ]:
```
import numpy as np
df=np.arange(0,99,7)
p=np.random.choice(df,size=(3,4))
print(df)
print(p)
```

Write a Code to convert a 1-D array to a 3-D array

```
One Dimension Array
 [ 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]

Multi-Dimension Array
 [[[ 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]]]
```

In [2]:
```
p=np.arange(0,27).reshape(3,3,3)
print(p)
```

```
[[[ 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]]]
```

Combine the given two arrays as below

Given Array

```
[[1 2 3]
 [4 5 6]]
[[ 7  8  9]
 [10 11 12]]
```

Resultant Array

```
[[ 1  2  3  7  8  9]
 [ 4  5  6 10 11 12]]
```

In [3]:

```
arr1=np.array([[1,2,3],[4,5,6]])
arr2=np.array([[7,8,9],[10,11,12]])
p=np.concatenate((arr1,arr2),axis=1)
print(p)
```

```
[[ 1  2  3  7  8  9]
 [ 4  5  6 10 11 12]]
```

Generate of given count of equally spaced 10 numbers within a range of 0 to 100

```
[  0.          11.11111111  22.22222222  33.33333333  44.44444444
  55.55555556  66.66666667  77.77777778  88.88888889 100.        ]
```

In [4]:

```
a=np.linspace(0,100,10)
print(a)
```

```
[  0.          11.11111111  22.22222222  33.33333333  44.44444444
  55.55555556  66.66666667  77.77777778  88.88888889 100.        ]
```

Generate the matrix of with 5 rows and 5 columns and assign to one value "999"

In [5]:

```
p=np.random.randint(100,size=(5,5))
print(p)
p[4,4]=999
print(p)
```

```
[[90 19 92 69  3]
 [ 1 68 35 34 13]
 [87 86 19 89 30]
 [16 12 48 45 46]
```

```
 [21 30 76  2  1]]
[[ 90  19  92  69   3]
 [  1  68  35  34  13]
 [ 87  86  19  89  30]
 [ 16  12  48  45  46]
 [ 21  30  76   2 999]]
```

Generate the series as below

```
Drygrapes    26
Cashew       48
Walnut       65
Fig          64
Dates        88
Name: Jan Month Sales of Dry Fruits, dtype: int64
Drygrapes    50
Cashew       38
Walnut       62
Fig          78
Dates        93
Name: Feb Month Sales of Dry Fruits, dtype: int64
```

In [8]:

```
import pandas as pd
Jan = pd.Series([26,48,65,64,88],index=["Drygrapes","Cashew","Walnut","Fig","Dates"],na
Feb = pd.Series([50,38,62,78,93],index=["Drygrapes","Cashew","Walnut","Fig","Dates"],na
print(Jan)
print(Feb)
```

```
Drygrapes    26
Cashew       48
Walnut       65
Fig          64
Dates        88
Name: Jan Month Sales of Dry Fruits, dtype: int64
Drygrapes    50
Cashew       38
Walnut       62
Fig          78
Dates        93
Name: Feb Month Sales of Dry Fruits, dtype: int64
```

Find the sales difference of two months Jan and Feb and generate the output as below

```
Sales Difference for the month of Jan and Feb

Drygrapes    24
Cashew      -10
Walnut       -3
Fig          14
Dates         5
Name: Difference in Sales, dtype: int64
```

In [9]:

```
p=pd.Series((Feb-Jan),name="Difference in sales")
print("Sales Difference for the month of Jan and Feb")
print(p)
```

```
Sales Difference for the month of Jan and Feb
Drygrapes    24
Cashew      -10
Walnut       -3
Fig          14
Dates         5
Name: Difference in sales, dtype: int64
```

Display the Dryfruits that have an increased its sales in the Feb month compared to Jan month

```
Displaying the dry fruits that have an increase in sales in Feb month when compared to Jan Month

Drygrapes    24
Fig          14
Dates         5
Name: Difference in Sales, dtype: int64
```

In [10]:

```
sales_increase=Feb-Jan
fruits_increase=sales_increase[sales_increase> 0]
print(fruits_increase)
```

```
Drygrapes    24
Fig          14
Dates         5
dtype: int64
```

Calculate the percentage of increase in sales

Formula = (Sales in Jan month - Sales in Feb Month)/Sales of Feb month * 100

And Display the result as below

```
Calculating the percentage of increase in sales

Cashew              NaN
Dates          5.376344
Drygrapes     48.000000
Fig           17.948718
Walnut              NaN
dtype: float64
```

In [11]:

```
print("Calculating the percentage of increase in sales")
Percent_increase=((Jan-Feb) / Feb) * 100
display(Percent_increase)
```

```
Calculating the percentage of increase in sales
Drygrapes   -48.000000
Cashew       26.315789
Walnut        4.838710
Fig         -17.948718
Dates        -5.376344
dtype: float64
```

Use the above generate series and generate the below dataframe.

	January	Febuary	Percentage of increase in sales Jan to Feb
Drygrapes	26	50	48.000000
Cashew	48	38	NaN
Walnut	65	62	NaN
Fig	64	78	17.948718
Dates	88	93	5.376344

In [12]:

```
Percent_increase=((Feb-Jan) / Feb)*100
df=pd.DataFrame({
    "January": Jan,
    "February":Feb,
    "Percentage of increase in sales Jan to Feb" : Percent_increase,
})
display(df)
```

	January	February	Percentage of increase in sales Jan to Feb
Drygrapes	26	50	48.000000
Cashew	48	38	-26.315789
Walnut	65	62	-4.838710
Fig	64	78	17.948718
Dates	88	93	5.376344

Rename the column name of Percentage of sales as below.

	January	Febuary	Profit_Percentage
Drygrapes	26	50	48.000000
Cashew	48	38	NaN
Walnut	65	62	NaN
Fig	64	78	17.948718
Dates	88	93	5.376344

In [13]:
```
percent_increase =pd.Series([48.000000, float("nan"),float( "nan") , 17.948718, 5.37634
df=pd.DataFrame({
    "January": Jan,
    "February": Feb,
    "Profit_Percentage":percent_increase
})
display(df)
```

	January	February	Profit_Percentage
0	NaN	NaN	48.000000
1	NaN	NaN	NaN
2	NaN	NaN	NaN
3	NaN	NaN	17.948718
4	NaN	NaN	5.376344
Cashew	48.0	38.0	NaN
Dates	88.0	93.0	NaN
Drygrapes	26.0	50.0	NaN
Fig	64.0	78.0	NaN
Walnut	65.0	62.0	NaN

Replace the NAN values with '0' as below

	January	Febuary	Profit_Percentage
Drygrapes	26	50	48.000000
Cashew	48	38	0.000000
Walnut	65	62	0.000000
Fig	64	78	17.948718
Dates	88	93	5.376344

In [14]:
```
df=pd.DataFrame({
    "January": Jan,
    "February": Feb,
    "Percentage of Increase": percent_increase

})
df=df.fillna(0)
display(df)
```

	January	February	Percentage of Increase
0	0.0	0.0	48.000000
1	0.0	0.0	0.000000
2	0.0	0.0	0.000000
3	0.0	0.0	17.948718
4	0.0	0.0	5.376344
Cashew	48.0	38.0	0.000000
Dates	88.0	93.0	0.000000
Drygrapes	26.0	50.0	0.000000
Fig	64.0	78.0	0.000000
Walnut	65.0	62.0	0.000000

Round of the profit percentage to two decimal values as below.

	January	Febuary	Profit_Percentage
Drygrapes	26	50	48.00
Cashew	48	38	0.00
Walnut	65	62	0.00
Fig	64	78	17.95
Dates	88	93	5.38

In [15]:
```
df["Percentage of Increase"]=df["Percentage of Increase"].round(2)
display(df)
```

	January	February	Percentage of Increase
0	0.0	0.0	48.00
1	0.0	0.0	0.00
2	0.0	0.0	0.00
3	0.0	0.0	17.95
4	0.0	0.0	5.38
Cashew	48.0	38.0	0.00
Dates	88.0	93.0	0.00
Drygrapes	26.0	50.0	0.00
Fig	64.0	78.0	0.00
Walnut	65.0	62.0	0.00

Generate the Dataframe as below which describes about the step count of 5 persons.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Jack	1020	2400	2800	1056	1089	2800	1080
Lawrence	2500	1000	1500	2300	3500	1500	2800
Susen	450	900	500	1089	2000	500	1080
Kiran	3000	2890	1890	3500	4500	1890	2890
George	5000	3500	4955	4256	5000	4955	3855

In [5]:

```
import pandas as pd

data={
    "Day 1": [1020, 2400, 450, 3000, 5000],
    "Day 2": [2400, 1000, 900, 2890, 3500],
    "Day 3": [2800, 1500, 500, 1890, 4955],
    "Day 4": [1056, 2300, 1089, 3500, 4256],
    "Day 5": [1089, 3500, 2000, 4500, 5000],
    "Day 6": [2800, 1500, 500, 1890, 4955],
    "Day 7": [1080, 2800, 1080, 2890, 3855]
}
index=["Jack","Lawrence","Susen","Kiran","George"]
df=pd.DataFrame(data, index=index)
display(df)
```

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Jack	1020	2400	2800	1056	1089	2800	1080
Lawrence	2400	1000	1500	2300	3500	1500	2800
Susen	450	900	500	1089	2000	500	1080
Kiran	3000	2890	1890	3500	4500	1890	2890
George	5000	3500	4955	4256	5000	4955	3855

Add the column "Total Step count" as below which consist of total step count for 7 days and if the sum value is float convert that to integer

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Total_Step_Count_Week
Jack	1020	2400	2800	1056	1089	2800	1080	12245
Lawrence	2500	1000	1500	2300	3500	1500	2800	15100
Susen	450	900	500	1089	2000	500	1080	6519
Kiran	3000	2890	1890	3500	4500	1890	2890	20560
George	5000	3500	4955	4256	5000	4955	3855	31521

In [6]:
```
df["Total_Step_Count"]= df.sum(axis=1)
display(df)
```

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Total_Step_Count
Jack	1020	2400	2800	1056	1089	2800	1080	12245
Lawrence	2400	1000	1500	2300	3500	1500	2800	15000
Susen	450	900	500	1089	2000	500	1080	6519
Kiran	3000	2890	1890	3500	4500	1890	2890	20560
George	5000	3500	4955	4256	5000	4955	3855	31521

Calculate the average step count for each person as below

```
Jack         1749.285714
Lawrence     2157.142857
Susen         931.285714
Kiran        2937.142857
George       4503.000000
dtype: float64
```

In [7]:
```
df["Average_Step_Count"]=df.iloc[:,:7].mean(axis=1)
print(df["Average_Step_Count"])
```

```
Jack         1749.285714
Lawrence     2142.857143
Susen         931.285714
Kiran        2937.142857
George       4503.000000
Name: Average_Step_Count, dtype: float64
```

Average step count need to rounded off to one decimal and add it as a new column to the dataframe.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Total_Step_Count_Week	Average_Step_Count
Jack	1020	2400	2800	1056	1089	2800	1080	12245	1749.0
Lawrence	2500	1000	1500	2300	3500	1500	2800	15100	2157.0
Susen	450	900	500	1089	2000	500	1080	6519	931.0
Kiran	3000	2890	1890	3500	4500	1890	2890	20560	2937.0
George	5000	3500	4955	4256	5000	4955	3855	31521	4503.0

In [8]:
```
display(df)
```

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Total_Step_Count	Average_Step_Count
Jack	1020	2400	2800	1056	1089	2800	1080	12245	1749.285714
Lawrence	2400	1000	1500	2300	3500	1500	2800	15000	2142.857143
Susen	450	900	500	1089	2000	500	1080	6519	931.285714
Kiran	3000	2890	1890	3500	4500	1890	2890	20560	2937.142857
George	5000	3500	4955	4256	5000	4955	3855	31521	4503.000000

Find the minimum step count of each person and which has to be added as the new column to the dataframe it type must be an integer

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Total_Step_Count_Week	Average_Step_Count	Minimum_Step_Count
Jack	1020	2400	2800	1056	1089	2800	1080	12245	1749.0	1020
Lawrence	2500	1000	1500	2300	3500	1500	2800	15100	2157.0	1000
Susen	450	900	500	1089	2000	500	1080	6519	931.0	450
Kiran	3000	2890	1890	3500	4500	1890	2890	20560	2937.0	1890
George	5000	3500	4955	4256	5000	4955	3855	31521	4503.0	3500

In [9]:
```
df["Minimum_Step_Count"]=df.iloc[:,:7].min(axis=1)
display(df)
```

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Total_Step_Count	Average_Step_Count	Minimum_S
Jack	1020	2400	2800	1056	1089	2800	1080	12245	1749.285714	
Lawrence	2400	1000	1500	2300	3500	1500	2800	15000	2142.857143	
Susen	450	900	500	1089	2000	500	1080	6519	931.285714	
Kiran	3000	2890	1890	3500	4500	1890	2890	20560	2937.142857	
George	5000	3500	4955	4256	5000	4955	3855	31521	4503.000000	

Find the maximum step count of each person and which has to be added as the new column to the dataframe it type must be an integer

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Total_Step_Count_Week	Average_Step_Count	Minimum_Step_Count	Maximum_Step_Count
Jack	1020	2400	2800	1056	1089	2800	1080	12245	1749.0	1020	2800
Lawrence	2500	1000	1500	2300	3500	1500	2800	15100	2157.0	1000	3500
Susen	450	900	500	1089	2000	500	1080	6519	931.0	450	2000
Kiran	3000	2890	1890	3500	4500	1890	2890	20560	2937.0	1890	4500
George	5000	3500	4955	4256	5000	4955	3855	31521	4503.0	3500	5000

In [10]:
```
df["Maximum_Step_Count"] = df.iloc[:,:7].max(axis=1)
display(df)
```

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Total_Step_Count	Average_Step_Count	Minimum_S
Jack	1020	2400	2800	1056	1089	2800	1080	12245	1749.285714	
Lawrence	2400	1000	1500	2300	3500	1500	2800	15000	2142.857143	
Susen	450	900	500	1089	2000	500	1080	6519	931.285714	
Kiran	3000	2890	1890	3500	4500	1890	2890	20560	2937.142857	
George	5000	3500	4955	4256	5000	4955	3855	31521	4503.000000	

Display the name of the person whose average step count is minimum.

```
'Susen'
```

In [12]:
```
def get_row_with_min_value(df, column_name):
    min_row = df[df[column_name] == df[column_name].min()]
    row_name = min_row.index[0]
    return row_name
get_row_with_min_value(df, "Average_Step_Count")
```

Out[12]:
```
'Susen'
```

Load the student database "student_exercise"

In [23]:
```
import pandas as pd
cc=pd.read_csv("student_exercise.csv")
print(cc)
```

```
    id         name  class  mark  gender
0    1     John Deo   Four    75  female
1    2     Max Ruin  Three    85    male
2    3       Arnold  Three    55    male
3    4   Krish Star   Four    60  female
4    5    John Mike   Four    60  female
5    6    Alex John   Four    55    male
6    7  My John Rob  Fifth    78    male
7    8       Asruid   Five    85    male
8    9      Tes Qry    Six    78    male
9   10     Big John   Four    55  female
10  11       Ronald    Six    89  female
11  12        Recky    Six    94  female
12  13          Kty  Seven    88  female
13  14         Bigy  Seven    88  female
14  15     Tade Row   Four    88    male
15  16        Gimmy   Four    88    male
16  17        Tumyu    Six    54    male
17  18        Honny   Five    75    male
18  19        Tinny   Nine    18    male
19  20       Jackly   Nine    65  female
20  21   Babby John   Four    69  female
21  22       Reggid  Seven    55  female
22  23        Herod  Eight    79    male
23  24    Tiddy Now  Seven    78    male
24  25     Giff Tow  Seven    88    male
25  26       Crelea  Seven    79    male
26  27     Big Nose  Three    81  female
27  28    Rojj Base  Seven    86  female
28  29  Tess Played  Seven    55    male
29  30    Reppy Red    Six    79  female
30  31  Marry Toeey   Four    88    male
31  32    Binn Rott  Seven    90  female
32  33    Kenn Rein    Six    96  female
33  34     Gain Toe  Seven    69    male
34  35   Rows Noump    Six    88  female
```

Display the columns of the dataframes

In [24]:
```
print(cc.columns.values)
```

```
['id' 'name' 'class' 'mark' 'gender']
```

Display the descriptive statistics of numeric columns in the DataFrame

In [25]:
```
cc.describe()
```

Out[25]:

	id	mark
count	35.000000	35.000000
mean	18.000000	74.657143
std	10.246951	16.401117

	id	mark
min	1.000000	18.000000
25%	9.500000	62.500000
50%	18.000000	79.000000
75%	26.500000	88.000000
max	35.000000	96.000000

Display the details of top 5 student based on their marks

In [26]:
```
cc.nlargest(5, ['mark'])
```

Out[26]:

	id	name	class	mark	gender
32	33	Kenn Rein	Six	96	female
11	12	Recky	Six	94	female
31	32	Binn Rott	Seven	90	female
10	11	Ronald	Six	89	female
12	13	Kty	Seven	88	female

Display the first 5 records of NAME column of the DataFrame

In [27]:
```
cc.head(5)
```

Out[27]:

	id	name	class	mark	gender
0	1	John Deo	Four	75	female
1	2	Max Ruin	Three	85	male
2	3	Arnold	Three	55	male
3	4	Krish Star	Four	60	female
4	5	John Mike	Four	60	female

filter rows where the "Mark" column is greater than 60

In [30]:
```
cc[(cc['mark']>60)]
```

Out[30]:

	id	name	class	mark	gender
0	1	John Deo	Four	75	female

	id	name	class	mark	gender
1	2	Max Ruin	Three	85	male
6	7	My John Rob	Fifth	78	male
7	8	Asruid	Five	85	male
8	9	Tes Qry	Six	78	male
10	11	Ronald	Six	89	female
11	12	Recky	Six	94	female
12	13	Kty	Seven	88	female
13	14	Bigy	Seven	88	female
14	15	Tade Row	Four	88	male
15	16	Gimmy	Four	88	male
17	18	Honny	Five	75	male
19	20	Jackly	Nine	65	female
20	21	Babby John	Four	69	female
22	23	Herod	Eight	79	male
23	24	Tiddy Now	Seven	78	male
24	25	Giff Tow	Seven	88	male
25	26	Crelea	Seven	79	male
26	27	Big Nose	Three	81	female
27	28	Rojj Base	Seven	86	female
29	30	Reppy Red	Six	79	female
30	31	Marry Toeey	Four	88	male
31	32	Binn Rott	Seven	90	female
32	33	Kenn Rein	Six	96	female
33	34	Gain Toe	Seven	69	male
34	35	Rows Noump	Six	88	female

fill missing values with a specific value or with zero

In [31]:
```
new_cc=cc.fillna(0)
```

filter rows where the "class" column is Three and the "mark" column is More than 50.

In [32]:
```
cc.loc[(cc['mark']>50) & (cc['class'].str.startswith('T'))]
```

Out[32]:

	id	name	class	mark	gender
1	2	Max Ruin	Three	85	male
2	3	Arnold	Three	55	male
26	27	Big Nose	Three	81	female

Display the total number of missing values in each column

In [34]:
```
cc.isna().sum().sum()
```

Out[34]:
```
0
```

Randomly select 7 rows (sample) from a DataFrame

In [35]:
```
cc.sample(7)
```

Out[35]:

	id	name	class	mark	gender
28	29	Tess Played	Seven	55	male
13	14	Bigy	Seven	88	female
2	3	Arnold	Three	55	male
21	22	Reggid	Seven	55	female
8	9	Tes Qry	Six	78	male
31	32	Binn Rott	Seven	90	female
20	21	Babby John	Four	69	female