In [2]:
```
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 [4]:
```
df=np.array([0,7,14,21,28,35,42,49,56,63,70,77,84,91,98])
print(df)
print("Generate the random number that contains multiples of 7")
d=np.random.choice([0,7,14,21,28,35,42,49,56,63,70,77,84,91,98],size=(3,3))
print(d)
```

```
[ 0  7 14 21 28 35 42 49 56 63 70 77 84 91 98]
Generate the random number that contains multiples of 7
[[63 84 56]
 [14 84 70]
 [ 0 63 21]]
```

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 [5]:
```
import numpy as np
a=np.arange(0,27)
print("one Dimensional array")
print(a)
df=np.random.randint(27,size=(3,3,3))
print("multidimensional array")
print(df)
```

```
one Dimensional 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]
```

```
multidimensional array
[[[20 18  4]
  [20 20  7]
  [17 14 16]]

 [[20  8  6]
  [20 20  7]
  [ 9 16 17]]

 [[14  0 12]
  [23 23 11]
  [ 5 20  5]]]
```

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 [10]:

```
import numpy as np
a=np.array([[1,2,3],[4,5,6]])
b=np.array([[7,8,9],[10,11,12]])
print("given array")
print(a)
print(b)
df=np.concatenate((a,b),axis=1)
print("Resultant array")
print(df)
```

```
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]]
```

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 [7]:

```
import numpy as np
value = np.linspace(0,100,10)
print(value)
```

```
[  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 [25]:

```
import numpy as np
df=np.random.randint(50,size=(5,5))
df[0][3]="999"
print(df)
```

```
[[  8  20  15 999  30]
 [  3  42  39  40  22]
 [ 12  19  45  27  42]
 [ 25  27  33  15   2]
 [ 20  45  26  18   8]]
```

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 [28]:

```
Jan = pd.Series([26,48,65,64,88],index=["Drygrapes","Cashew","Walnut","Fig","Dates"]
Feb = pd.Series([50,38,62,78,93],index=["Drygrapes","Cashew","Walnut","Fig","Dates"]
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 [60]:

```
import pandas as pd
df=pd.Series((Feb-Jan),name="Differnce in sales")
print(df)
```

```
Drygrapes    24
Cashew      -10
Walnut       -3
Fig          14
Dates         5
Name: Differnce 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 [16]:

```
import pandas as pd
Jan = pd.Series([26,48,65,64,88],index=["Drygrapes","Cashew","Walnut","Fig","Dates"]
Feb = pd.Series([50,38,62,78,93],index=["Drygrapes","Cashew","Walnut","Fig","Dates"]
sales = Feb - Jan
fruits = sales[sales > 0]
print(fruits)
```

```
Drygrapes    24
Fig          14
Dates         5
Name: differnce in sales, 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 [36]:

```
import pandas as pd
Jan = pd.Series([26,48,65,64,88],index=["Drygrapes","Cashew","Walnut","Fig","Dates"]
Feb = pd.Series([50,38,62,78,93],index=["Drygrapes","Cashew","Walnut","Fig","Dates"]
print("calculating the percentage of increase in sales")
p=pd.Series((Jan-Feb)/Feb*100)
print(p)
```

```
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 [32]:

```
p = ((Feb - Jan) / Feb) * 100
df = pd.DataFrame({
    "January": Jan,
    "February": Feb,
    "percentage of increase in sale Jan to Feb": p,
})

display(df)
```

	January	February	percentage of increase in sale 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 [40]:
```
percentage_increase = pd.Series([48.000000, float("nan"), float("nan"), 17.948718, 5
df = pd.DataFrame({
    "January": Jan,
    "February": Feb,
    "profit_Percentage":percentage_increase
})

display(df)
```

	January	February	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

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 [41]:
```
percentage_increase = pd.Series([48.000000, float("0"), float("0"), 17.948718, 5.376
df = pd.DataFrame({
    "January": Jan,
    "February": Feb,
    "profit_Percentage":percentage_increase
})

display(df)
```

	January	February	profit_Percentage
Drygrapes	26	50	48.000000
Cashew	48	38	0.000000
Walnut	65	62	0.000000
Fig	64	78	17.948718

	January	February	profit_Percentage
Dates	88	93	5.376344

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 [47]:

```
s = percentage_increase.round(2)
df = pd.DataFrame({
    "January": Jan,
    "February": Feb,
    "percentage of increase in sale Jan to Feb": s,
})

display(df)
```

	January	February	percentage of increase in sale Jan to Feb
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

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 [68]:

```
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']
m = pd.DataFrame(data, index=index)
display(m)
```

	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 [85]:

```
import pandas as pd
p=m.sum(axis=1)
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],
    "Total_Step_count_week":p
}
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	Total_Step_count_week
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 [86]:

```
import pandas as pd
p=m.mean(axis=1)
print(p)
```

```
Jack         1749.285714
Lawrence     2142.857143
Susen         931.285714
Kiran        2937.142857
George       4503.000000
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 [93]:

```
import pandas as pd
p=m.sum(axis=1)
s=m.mean(axis=1)
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],
    "Total_Step_count_week":p,
    "Average_step_count":s.round()
}
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	Total_Step_count_week	Average_step_count
Jack	1020	2400	2800	1056	1089	2800	1080	12245	1749.0
Lawrence	2400	1000	1500	2300	3500	1500	2800	15000	2143.0
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

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 [94]:

```
import pandas as pd
p=m.sum(axis=1)
s=m.mean(axis=1)
d=m.min(axis=1)
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],
    "Total_Step_count_week":p,
    "Average_step_count":s.round(),
    "minimum_step_count":d
}
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	Total_Step_count_week	Average_step_count	m
Jack	1020	2400	2800	1056	1089	2800	1080	12245	1749.0	
Lawrence	2400	1000	1500	2300	3500	1500	2800	15000	2143.0	
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	

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 [97]:

```
import pandas as pd
p=m.sum(axis=1)
s=m.mean(axis=1)
d=m.min(axis=1)
e=m.max(axis=1)
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],
    "Total_Step_count_week":p,
    "Average_step_count":s.round(),
    "minimum_step_count":d,
    "maximum_step_count":e
}
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	Total_Step_count_week	Average_step_count	m
Jack	1020	2400	2800	1056	1089	2800	1080	12245	1749.0	
Lawrence	2400	1000	1500	2300	3500	1500	2800	15000	2143.0	
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	

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

```
'Susen'
```

In [124…

```
def get_row(df, column_name):
    min_row = df[df[column_name] == df[column_name].min()]
    row_name = min_row.index[0]
```

```
    return row_name
get_row(df, "Average_step_count")
```

Out[124…
```
'Susen'
```

Load the student database "student_exercise"

In [ ]:
```
```

Display the columns of the dataframes

In [126…
```
df.columns.to_list()
```

Out[126…
```
['Day 1',
 'Day 2',
 'Day 3',
 'Day 4',
 'Day 5',
 'Day 6',
 'Day 7',
 'Total_Step_count_week',
 'Average_step_count']
```

Display the descriptive statistics of numeric columns in the DataFrame

In [127…
```
df.describe()
```

Out[127…

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	To
count	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.00000	
mean	2374.000000	2138.000000	2329.000000	2440.200000	3217.800000	2329.000000	2341.00000	
std	1790.497138	1152.918037	1683.962589	1430.184673	1651.872937	1683.962589	1223.16393	
min	450.000000	900.000000	500.000000	1056.000000	1089.000000	500.000000	1080.00000	
25%	1020.000000	1000.000000	1500.000000	1089.000000	2000.000000	1500.000000	1080.00000	
50%	2400.000000	2400.000000	1890.000000	2300.000000	3500.000000	1890.000000	2800.00000	
75%	3000.000000	2890.000000	2800.000000	3500.000000	4500.000000	2800.000000	2890.00000	
max	5000.000000	3500.000000	4955.000000	4256.000000	5000.000000	4955.000000	3855.00000	

Display the details of top 5 student based on their marks

In [ ]:

Display the first 5 records of NAME column of the DataFrame

In [ ]:

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

In [ ]:

fill missing values with a specific value or with zero

In [ ]:

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

In [ ]:

Display the total number of missing values in each column

In [ ]:

Randomly select 7 rows (sample) from a DataFrame