WEEK3 & 4

September 25, 2024

import numpy as np a = [1,3,3,5] sum = np.sum(a) print("Sum of array elements is",sum) min = np.min(a) print("Minimum of array elements is",min) max = np.max(a) print("Maximum of array elements is",max) mean = np.mean(a) print("Mean of array elements is",mean) med = np.median(a) print("Median of array elements is",med) cor = np.corrcoef(a) print("Correaltion coefficiant of array elements is",cor) std = np.std(a) print("standard deviation of array elements is",std)

[6]:
```
mat = [[1,2,3,4],[3,5,6,8],[8,9,3,4]]
sum = np.sum(mat)
print("Sum of matrix elements ",sum)
min = np.min(mat)
print("Minimum of the matrix:",min)
max = np.max(mat)
print("Maximum of the matrix:",max)
mean = np.mean(mat)
print("Mean of the matrix:",mean)
med = np.median(a)
print("Median of the matrix:",med)
cor = np.corrcoef(mat)
print("Correaltion coefficiants of the matrix elements:\n",cor)
std = np.std(mat)
print("standard deviation of array elements is",std)
```

```
Swru 56
Minimum of the matrix: 1
Maximum of the matrix: 9
Mean of the matrix: 4.666666666666667
Median of the matrix: 3.0
Correaltion coefficiants of the matrix elements:
 [[ 1.          0.99227788 -0.78935222]
 [ 0.99227788  1.         -0.70710678]
 [-0.78935222 -0.70710678  1.        ]]
standard deviation of array elements is 2.4608038433722337
```

[7]:
```
mat = np.array([[1,7,3,4],[3,5,6,8],[10,9,3,4]])
print(mat)
sum = np.sum(mat, axis=1)
print("Sum of array elements row-wise",sum)
```

```
sum = np.sum(mat, axis=0)
print("Sum of array elements column-wise",sum)
min = np.min(mat, axis=1)
print("Row-wise minimum of the matrix:",min)
max = np.max(mat,axis=0)
print("Column-wise maximum of the matrix:",max)
```

```
[[ 1  7  3  4]
 [ 3  5  6  8]
 [10  9  3  4]]
Sum of array elements row-wise [15 22 26]
Sum of array elements column-wise [14 21 12 16]
Row-wise minimum of the matrix: [1 3 3]
Column-wise maximum of the matrix: [10  9  6  8]
```

[16]:
```
def sum(a,b,c):
 return a+b+c

sum(4,5,9)
```

[16]: 18

[12]:
```
def cube(x):
 return x*x*x*x
cube(7)
```

[12]: 2401

[15]:
```
lambda_cube = lambda y: y*y
lambda_cube(2)
```

[15]: 4

[17]:
```
sum = lambda a,b:a+b
sum(4,5)
```

[17]: 9

[18]:
```
add = lambda num: num + 4
print( add(6))
```

```
10
```

[24]:
```
def greater(a, b):
    if a > b:
        return a
    else:
        return b
```

```
# Example usage
print(greater(488, 5))
```

```
488
```

[25]:
```
Max = lambda a, b : a if(a > b) else b
Max(4,5)
```

[25]:
```
5
```

[26]:
```
my_list= [5,7,2,8,6]
my_list_squared = []
for i in my_list:
 i_squared = i**2
 my_list_squared.append(i_squared)
my_list_squared
```

[26]:
```
[25, 49, 4, 64, 36]
```

[27]:
```
my_list_squared = [i**2 for i in my_list]
my_list_squared
```

[27]:
```
[25, 49, 4, 64, 36]
```

[28]:
```
my_list_squared = list(map(lambda i: i**2, my_list))
my_list_squared
```

[28]:
```
[25, 49, 4, 64, 36]
```

[29]:
```
def add4(x):
 return x+4
list1 = [4,6,7,8,9]
list2 = list(map(add4,list1))
list2
```

[29]:
```
[8, 10, 11, 12, 13]
```

[32]:
```
x = np.zeros(10)
print(x)
print("Update sixth value to 11")
x[8] = 1
print(x)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Update sixth value to 11
[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
```

[33]:
```
item_list = ['Bread', 'Milk', 'Eggs', 'Butter', 'Cocoa']
student_marks = [78, 47, 96, 55, 34]
```

```
hetero_list = [ 1,2,3.0, 'text', True, 3+2j ]
```

[34]:
```
student_marks = [78, 47, 96, 55, 34]
for i in range(len(student_marks)):
 student_marks[i]+=5
print(student_marks)
```

```
[83, 52, 101, 60, 39]
```

[35]:
```
%%time
#Used to calculate total operation time
list1 = list(range(1,1000000))
list2 = list(range(2,1000001))
list3 = []
for i in range(len(list1)):
 list3.append(list1[i]+list2[i])
```

```
CPU times: total: 391 ms
Wall time: 386 ms
```

[36]:
```
import numpy as np
student_marks_arr = np.array([78, 92, 36, 64, 89])
student_marks_arr
```

[36]:
```
array([78, 92, 36, 64, 89])
```

[41]:
```
import numpy as np
car_attributes = [[18, 15, 18, 16, 17],[130, 165, 150, 150, 140],[307, 350,␣
 ↪318, 30]]
#creating a numpy array from car_attributes list
car_attributes_arr = np.array(car_attributes,dtype=object)
print(car_attributes_arr)
```

```
[list([18, 15, 18, 16, 17]) list([130, 165, 150, 150, 140])
 list([307, 350, 318, 30])]
```

[43]:
```
import numpy as np

car_attributes = [[18, 15, 18, 16, 17], [130, 165, 150, 150, 140], [307, 350,␣
 ↪318, 30]]
car_attributes_arr = np.array(car_attributes, dtype=object)

print(car_attributes_arr)
```

```
[list([18, 15, 18, 16, 17]) list([130, 165, 150, 150, 140])
 list([307, 350, 318, 30])]
```

[2]:
```
import numpy as np
```

```
car_attributes = [[18, 15, 18, 16, 17],
                  [130, 165, 150, 150, 140],
                  [307, 350, 318, 290, 320]]


car_attributes_arr = np.array(car_attributes, dtype='float')

print(car_attributes_arr)
print(car_attributes_arr.dtype)
```

```
[[ 18.  15.  18.  16.  17.]
 [130. 165. 150. 150. 140.]
 [307. 350. 318. 290. 320.]]
float64
```

[46]:
```
import numpy as np

car_attributes = [
    [18, 15, 18, 16, 17],
    [130, 165, 150, 150, 140],
    [307, 350, 318, 30]
]

car_attributes_arr = np.array(car_attributes, dtype=object)
print(car_attributes_arr)
print(car_attributes_arr.dtype)
```

```
[list([18, 15, 18, 16, 17]) list([130, 165, 150, 150, 140])
 list([307, 350, 318, 30])]
object
```

[47]:
```
car_names = ['chevrolet', 'buick', 'ply', 'amc', 'ford']
horsepower = [130, 165, 150, 150, 140]
car_hp_arr = np.array([car_names, horsepower])
car_hp_arr
```

[47]:
```
array([['chevrolet', 'buick', 'ply', 'amc', 'ford'],
       ['130', '165', '150', '150', '140']], dtype='<U11')
```

[60]:
```
horsepower = [130, 165, 150, 150, 140]
horsepower_arr = np.array(horsepower)
print("Index of Minimum horsepower: ", np.argmin(horsepower_arr))
print("Index of Maximum horsepower: ", np.argmax(horsepower_arr))
```

```
Index of Minimum horsepower:  0
Index of Maximum horsepower:  1
```

[61]:
```
horsepower = [130, 165, 150, 150, 140]
#creating a numpy array from horsepower list
```

```
horsepower_arr = np.array(horsepower)
x = np.where(horsepower_arr >= 150)
print(x) # gives the indices
# With the indices , we can find those values
horsepower_arr[x]
```

```
(array([1, 2, 3], dtype=int64),)
```

[61]:
```
array([165, 150, 150])
```

[62]:
```
horsepower_arr[3]
```

[62]:
```
150
```

[63]:
```
horsepower = [130, 165, 150, 150, 140]
#creating a numpy array from horsepower list
horsepower_arr = np.array(horsepower)
#creating filter array
x = horsepower_arr > 135
print(x.dtype)
newarr = horsepower_arr[x]
print(x)
print(newarr)
```

```
bool
[False  True  True  True  True]
[165 150 150 140]
```

[64]:
```
#creating a list of 5 horsepower values
horsepower = [130, 165, 150, 150, 140]
#creating a numpy array from horsepower list
horsepower_arr = np.array(horsepower)
#using sort(array)
print('original array: ', horsepower_arr)
print('Sorted array: ', np.sort(horsepower_arr))
print('original array after sorting: ', horsepower_arr)
sortedarray = np.sort(horsepower_arr)
print(sortedarray)
```

```
original array:  [130 165 150 150 140]
Sorted array:  [130 140 150 150 165]
original array after sorting:  [130 165 150 150 140]
[130 140 150 150 165]
```

[65]:
```
horsepower = [130, 165, 150, 150, 140]
#creating a numpy array from horsepower list
horsepower_arr = np.array(horsepower)
#using sort(array)
print('original array: ', horsepower_arr)
```

```
horsepower_arr.sort()
print('original array after sorting: ', horsepower_arr)
```

```
original array:  [130 165 150 150 140]
original array after sorting:  [130 140 150 150 165]
```

[66]:
```
import numpy as np
#Creating a 2D array
Day_number = np.arange(1,11)
Steps_walked = [6012,7079,6886,7230,4598,5564,6971,7763,8032,9569]
arr = np.array([Day_number, Steps_walked])
arr = arr.T
arr
```

[66]:
```
array([[   1, 6012],
       [   2, 7079],
       [   3, 6886],
       [   4, 7230],
       [   5, 4598],
       [   6, 5564],
       [   7, 6971],
       [   8, 7763],
       [   9, 8032],
       [  10, 9569]])
```

[5]:
```
import numpy as np
import time
v1 = np.random.rand(1000000, 1)

v1_scaled = np.zeros((1000000, 1))

# Measure the start time
start = time.time()
v1_scaled = 2 * v1

end = time.time()

print("Scaling vector Answer = " + str(v1_scaled[:10]))  # Print only the first␣
 ↪10 elements for brevity
print("Time taken = " + str(1000 * (end - start)) + " ms")
```

```
Scaling vector Answer = [[1.01192485]
 [1.85451915]
 [1.21993857]
 [1.14026674]
 [1.80903808]
 [0.92068774]
 [1.70251671]
```

```
 [1.74762717]
 [0.89428174]
 [0.58185255]]
Time taken = 0.0 ms
```

[2]:
```
a = 1
b = 2

result = a + b

print("1 + 2 =", result)
```

```
1 + 2 = 3
```

[1]:
```
import numpy as np
import time

v1 = np.random.rand(1000000, 1)
v1_scaled = np.zeros((1000000, 1))

start = time.time()
v1_scaled = 2 * v1
end = time.time()

print("Scaled vector result = " + str(v1_scaled[:10]))
print("Time taken = " + str(1000 * (end - start)) + " milliseconds")
```

```
Scaled vector result = [[0.87752247]
 [0.58359226]
 [0.13575574]
 [0.97285493]
 [0.65908095]
 [0.53213784]
 [0.35490197]
 [1.01087209]
 [1.43544008]
 [1.88570756]]
Time taken = 2.8624534606933594 milliseconds
```

[17]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import time
import matplotlib.pyplot as plt
```

[18]:
```
v1 = np.random.rand(1000000, 1)
v2 = np.random.rand(1000000, 1)
```

[19]:
```
start = time.process_time()
v1_scaled = np.zeros((1000000, 1))
for i in range(len(v1)):
 v1_scaled[i] = 2 * v1[i]
end = time.process_time()

print("Scaling vector Answer = " + str(v1_scaled))
print("Time taken = " + str(1000*(end - start)) + " ms")
```

```
Scaling vector Answer = [[1.52363134]
 [1.61330062]
 [0.60583127]
 …
 [0.19475037]
 [1.23281665]
 [1.6782906 ]]
Time taken = 4500.0 ms
```

[21]:
```
import pandas as pd
import numpy as np
marks = {'Chemistry': [67,90,66,32],
 'Physics': [45,92,72,40],
 'Mathematics': [50,87,81,12],
 'English': [19,90,72,68]}
marks_df = pd.DataFrame(marks, index = ['Swaroop', 'uday', 'darshan', 'skandha␣
 ↪gagan m, Shankarpura'])
marks_df
```

[21]:

	Chemistry	Physics	Mathematics	English
Swaroop	67	45	50	19
uday	90	92	87	90
darshan	66	72	81	72
skandha gagan m, Shankarpura	32	40	12	68

[22]:
```
import pandas as pd
import numpy as np
marks = {'Chemistry': [67,90,66,32],
 'Physics': [45,92,72,40],
 'Mathematics': [50,87,81,12],
 'English': [19,90,72,68]}
marks_df = pd.DataFrame(marks, index = ['Swaroop', 'uday', 'darshan', 'skandha␣
 ↪gagan m, Shankarpura'])
marks_df
marks_df['Total'] = marks_df['Chemistry'] + marks_df['Physics'] +␣
 ↪marks_df['Mathematics']
marks_df
```

[22]:

	Chemistry	Physics	Mathematics	English	Total
Swaroop	67	45	50	19	162
uday	90	92	87	90	269
darshan	66	72	81	72	219
skandha gagan m, Shankarpura	32	40	12	68	84

[11]:
```
import pandas as pd
import numpy as np

df = pd.read_excel("C:/Users/HP/Desktop/rainfall.xlsx")


df
```

[11]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7
…	…	…	…	…	…	…	…	…
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6

	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May
0	365.1	481.1	332.6	388.5	558.2	33.6	3373.2	136.3	560.3
1	228.9	753.7	666.2	197.2	359.0	160.5	3520.7	159.8	458.3
2	728.4	326.7	339.0	181.2	284.4	225.0	2957.4	156.7	236.1
3	502.0	160.1	820.4	222.2	308.7	40.1	3079.6	24.1	506.9
4	368.7	330.5	297.0	260.7	25.4	344.7	2566.7	1.3	309.7
…	…	…	…	…	…	…	…	…	
4111	350.2	254.0	255.2	117.4	184.3	14.9	1533.7	7.9	196.2
4112	231.5	381.2	179.8	145.9	12.4	8.8	1405.5	19.3	99.6
4113	296.4	154.4	180.0	72.8	78.1	26.7	1426.3	60.6	131.1
4114	116.1	466.1	132.2	169.2	59.0	62.3	1395.0	69.3	76.7
4115	257.5	146.4	160.4	165.4	231.0	159.0	1642.9	2.7	223.9

	Jun-Sep	Oct-Dec
0	1696.3	980.3
1	2185.9	716.7
2	1874.0	690.6
3	1977.6	571.0
4	1624.9	630.8
…	…	…

```
4111   1013.0    316.6
4112   1119.5    167.1
4113   1057.0    177.6
4114    958.5    290.5
4115    860.9    555.4

[4116 rows x 19 columns]
```

[12]:
```
sorted_df = df.sort_values(by = 'ANNUAL', ascending=False)
highest = sorted_df.iloc[0,1]
print("District that gets the highest annual rainfall:",highest)
```

```
District that gets the highest annual rainfall: 1948
```

[13]:
```
sorted_df.head(5)
```

[13]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
142	ARUNACHAL PRADESH	1948	35.6	119.5	136.1	441.1	1168.6	889.5	2362.8	603.3	350.4	150.1	52.7	21.4	6331.1	155.1	1745.8	4206.0	224.2
132	ARUNACHAL PRADESH	1938	144.8	121.6	340.5	395.3	306.5	1511.3	1355.1	790.8	877.3	220.6	59.7	5.6	6129.0	266.4	1042.3	4534.5	285.8
115	ARUNACHAL PRADESH	1921	78.9	54.3	180.3	358.0	598.0	1233.2	1433.0	885.9	603.4	246.3	4.6	15.5	5691.4	133.2	1136.3	4155.5	266.4
3602	COASTAL KARNATAKA	1961	0.0	0.0	0.9	47.7	635.0	1013.0	1884.9	936.3	702.8	309.6	21.0	2.7	5553.9	0.0	683.6	4536.9	333.4
112	ARUNACHAL PRADESH	1918	10.4	11.0	191.2	144.6	861.1	1609.9	1303.0	692.6	515.8	125.2	7.8	13.7	5486.3	21.4	1196.9	4121.3	146.7

[14]:
```
new_df = df.drop(['JAN','FEB', 'MAR','JUN', 'JUL','SEP', 'OCT','DEC'],axis=1)
new_df
```

[14]:

	SUBDIVISION	YEAR	APR	MAY	AUG	NOV	ANNUAL
0	ANDAMAN & NICOBAR ISLANDS	1901	2.3	528.8	481.1	558.2	3373.2
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	446.1	753.7	359.0	3520.7
2	ANDAMAN & NICOBAR ISLANDS	1903	1.0	235.1	326.7	284.4	2957.4
3	ANDAMAN & NICOBAR ISLANDS	1904	202.4	304.5	160.1	308.7	3079.6
4	ANDAMAN & NICOBAR ISLANDS	1905	26.9	279.5	330.5	25.4	2566.7
…	…	…	…	…	…	…	…
4111	LAKSHADWEEP	2011	85.9	107.2	254.0	184.3	1533.7

```
4112                LAKSHADWEEP  2012   76.8   21.2  381.2   12.4  1405.5
4113                LAKSHADWEEP  2013    5.3   88.3  154.4   78.1  1426.3
4114                LAKSHADWEEP  2014   14.9   57.4  466.1   59.0  1395.0
4115                LAKSHADWEEP  2015   87.1  133.1  146.4  231.0  1642.9

      Jan-Feb  Mar-May  Jun-Sep  Oct-Dec
0       136.3    560.3   1696.3    980.3
1       159.8    458.3   2185.9    716.7
2       156.7    236.1   1874.0    690.6
3        24.1    506.9   1977.6    571.0
4         1.3    309.7   1624.9    630.8
…         …        …        …        …
4111      7.9    196.2   1013.0    316.6
4112     19.3     99.6   1119.5    167.1
4113     60.6    131.1   1057.0    177.6
4114     69.3     76.7    958.5    290.5
4115      2.7    223.9    860.9    555.4

[4116 rows x 11 columns]
```

[27]:
```
# Drop the 'ANNUAL' column
new_df = df.drop(['ANNUAL'], axis=1)

# Create the pivot table based on the 'SUBDIVISION' column
table = pd.pivot_table(new_df, index=['SUBDIVISION'])

# Display the pivot table
table
```

[27]:
```
                                       APR         AUG         DEC  \
SUBDIVISION
ANDAMAN & NICOBAR ISLANDS           72.223148  400.047222  153.144860
ARUNACHAL PRADESH                  263.836082  495.229897   24.502105
ASSAM & MEGHALAYA                  203.115652  404.593043    8.951304
BIHAR                               16.918261  299.643478    3.694783
CHHATTISGARH                        16.773043  389.873043    5.248696
COASTAL ANDHRA PRADESH              26.740870  175.923478   11.420000
COASTAL KARNATAKA                   30.916522  713.618261   12.613913
EAST MADHYA PRADESH                  7.188696  369.368696    8.404348
EAST RAJASTHAN                       3.144348  218.277391    3.651304
EAST UTTAR PRADESH                   6.430435  275.613913    5.776522
GANGETIC WEST BENGAL                44.885217  311.382609    5.690435
GUJARAT REGION                       1.116522  259.193043    1.339130
HARYANA DELHI & CHANDIGARH           7.633913  150.840870    7.186087
HIMACHAL PRADESH                    62.428696  273.933043   39.893043
JAMMU & KASHMIR                     93.702609  180.973043   55.425439
JHARKHAND                           19.366957  325.524348    4.939130
```

KERALA	110.573913	421.977391	39.950435
KONKAN & GOA	4.266087	682.756522	4.516522
LAKSHADWEEP	45.163393	207.993750	60.810909
MADHYA MAHARASHTRA	9.146957	184.397391	5.848696
MATATHWADA	7.594783	166.484348	7.302609
NAGA MANI MIZO TRIPURA	170.733043	411.281739	12.399130
NORTH INTERIOR KARNATAKA	24.300870	119.459130	6.327826
ORISSA	34.160000	355.382609	5.567826
PUNJAB	12.660000	158.167826	12.694783
RAYALSEEMA	19.808696	107.511304	34.260000
SAURASHTRA & KUTCH	1.183478	118.770435	1.108696
SOUTH INTERIOR KARNATAKA	42.280870	174.239130	11.517391
SUB HIMALAYAN WEST BENGAL & SIKKIM	110.681739	520.763478	6.060000
TAMIL NADU	44.995652	95.887826	81.137391
TELANGANA	18.185217	215.059130	5.141739
UTTARAKHAND	35.166087	382.023478	22.035652
VIDARBHA	9.435652	285.949565	7.927826
WEST MADHYA PRADESH	2.375652	288.108696	6.296522
WEST RAJASTHAN	3.571304	94.555652	1.902609
WEST UTTAR PRADESH	6.253043	251.299130	7.114783

SUBDIVISION	FEB	JAN	JUL \
ANDAMAN & NICOBAR ISLANDS	27.994545	52.637273	400.042593
ARUNACHAL PRADESH	91.116667	47.297917	694.544792
ASSAM & MEGHALAYA	31.441739	16.974783	495.102609
BIHAR	14.393913	13.386087	324.441739
CHHATTISGARH	19.259130	14.206957	398.577391
COASTAL ANDHRA PRADESH	12.923478	7.483478	173.824348
COASTAL KARNATAKA	1.518261	1.937719	1127.028696
EAST MADHYA PRADESH	18.693913	19.401739	371.378261
EAST RAJASTHAN	5.417391	6.422609	223.347826
EAST UTTAR PRADESH	15.873913	16.012174	290.568696
GANGETIC WEST BENGAL	22.452174	12.595652	326.377391
GUJARAT REGION	1.191304	1.786087	348.920870
HARYANA DELHI & CHANDIGARH	17.433913	16.889565	150.015652
HIMACHAL PRADESH	90.894783	84.189565	280.284348
JAMMU & KASHMIR	115.450435	102.030435	179.837719
JHARKHAND	24.186087	17.621739	336.975652
KERALA	15.496522	12.246957	700.953043
KONKAN & GOA	0.546957	1.262609	1073.030435
LAKSHADWEEP	15.834513	27.494643	281.928829
MADHYA MAHARASHTRA	1.467826	3.054783	248.980000
MATATHWADA	4.443478	5.000870	180.648696
NAGA MANI MIZO TRIPURA	36.652174	14.025217	438.684348
NORTH INTERIOR KARNATAKA	3.172174	3.013043	138.531304
ORISSA	19.719130	12.329565	351.173043

PUNJAB	26.786957	25.246087	168.963478
RAYALSEEMA	5.680000	9.867826	96.081739
SAURASHTRA & KUTCH	1.615652	1.139130	194.970435
SOUTH INTERIOR KARNATAKA	4.163478	2.928696	231.359130
SUB HIMALAYAN WEST BENGAL & SIKKIM	22.974783	14.083478	646.402609
TAMIL NADU	13.422609	23.819130	71.314783
TELANGANA	9.688696	7.702609	247.499130
UTTARAKHAND	63.452174	53.797391	390.698261
VIDARBHA	11.982609	10.563478	329.428696
WEST MADHYA PRADESH	6.307895	9.241739	302.982609
WEST RAJASTHAN	4.930435	3.327826	95.171304
WEST UTTAR PRADESH	17.893913	17.666087	246.520000

SUBDIVISION	JUN	Jan-Feb	Jun-Sep	\
ANDAMAN & NICOBAR ISLANDS	471.580556	80.632727	1706.687850	
ARUNACHAL PRADESH	647.373958	138.416667	2271.422105	
ASSAM & MEGHALAYA	510.161739	48.413043	1720.590435	
BIHAR	174.315652	27.776522	1015.786087	
CHHATTISGARH	198.266087	33.462609	1204.500870	
COASTAL ANDHRA PRADESH	123.693913	20.404348	655.141739	
COASTAL KARNATAKA	841.326087	3.371053	2981.618261	
EAST MADHYA PRADESH	141.029565	38.094783	1076.018261	
EAST RAJASTHAN	63.399130	11.837391	602.998261	
EAST UTTAR PRADESH	110.712174	31.887826	861.486087	
GANGETIC WEST BENGAL	247.196522	35.042609	1130.657391	
GUJARAT REGION	121.284348	2.977391	878.251304	
HARYANA DELHI & CHANDIGARH	48.626087	34.331304	437.786957	
HIMACHAL PRADESH	91.220870	175.082609	775.664348	
JAMMU & KASHMIR	64.234783	217.482609	515.428070	
JHARKHAND	194.588696	41.809565	1084.510435	
KERALA	654.302609	27.739130	2022.840870	
KONKAN & GOA	688.569565	1.813043	2794.130435	
LAKSHADWEEP	327.627679	42.500000	983.554545	
MADHYA MAHARASHTRA	147.426087	4.526087	738.025217	
MATATHWADA	136.957391	9.448696	662.567826	
NAGA MANI MIZO TRIPURA	445.633913	50.669565	1609.941739	
NORTH INTERIOR KARNATAKA	100.993043	6.184348	501.927826	
ORISSA	210.860870	32.047826	1158.817391	
PUNJAB	46.466957	52.030435	460.392174	
RAYALSEEMA	64.742609	15.545217	400.058261	
SAURASHTRA & KUTCH	74.371304	2.752174	463.536522	
SOUTH INTERIOR KARNATAKA	141.417391	7.087826	684.338261	
SUB HIMALAYAN WEST BENGAL & SIKKIM	537.881739	37.060870	2126.391304	
TAMIL NADU	52.056522	37.239130	330.847826	
TELANGANA	142.126087	17.396522	780.186957	
UTTARAKHAND	162.551304	117.251304	1131.367826	

VIDARBHA	173.578261	22.544348	964.403478
WEST MADHYA PRADESH	111.781739	15.638596	864.043478
WEST RAJASTHAN	28.637391	8.255652	258.707826
WEST UTTAR PRADESH	77.597391	35.554783	721.676522

SUBDIVISION	MAR	MAY	Mar-May \
ANDAMAN & NICOBAR ISLANDS	31.824074	357.056881	462.249533
ARUNACHAL PRADESH	153.527368	358.522680	777.686316
ASSAM & MEGHALAYA	79.026957	341.539130	623.687826
BIHAR	10.124348	53.081739	80.126957
CHHATTISGARH	15.266957	21.048696	53.092174
COASTAL ANDHRA PRADESH	13.221739	62.549565	102.515652
COASTAL KARNATAKA	6.357391	122.787826	160.051304
EAST MADHYA PRADESH	13.637391	9.273043	30.096522
EAST RAJASTHAN	4.516522	9.820000	17.487826
EAST UTTAR PRADESH	8.907826	17.211304	32.553913
GANGETIC WEST BENGAL	29.090435	107.787826	181.766087
GUJARAT REGION	1.220870	5.809565	8.161739
HARYANA DELHI & CHANDIGARH	12.935652	14.533913	35.112174
HIMACHAL PRADESH	101.146087	58.156522	221.726957
JAMMU & KASHMIR	131.378261	67.476522	292.552174
JHARKHAND	18.423478	48.317391	86.095652
KERALA	36.814783	229.881739	377.253913
KONKAN & GOA	1.374783	33.515652	39.161739
LAKSHADWEEP	14.350893	163.893750	223.822727
MADHYA MAHARASHTRA	3.596522	22.943478	35.692174
MATATHWADA	7.105217	15.646957	30.352174
NAGA MANI MIZO TRIPURA	77.199130	290.839130	538.769565
NORTH INTERIOR KARNATAKA	7.123478	47.035652	78.460870
ORISSA	21.134783	64.886087	120.184348
PUNJAB	23.651304	14.136522	50.440000
RAYALSEEMA	8.076522	50.475652	78.358261
SAURASHTRA & KUTCH	1.296522	4.662609	7.140870
SOUTH INTERIOR KARNATAKA	9.485217	92.100000	143.861739
SUB HIMALAYAN WEST BENGAL & SIKKIM	43.135652	269.143478	422.968696
TAMIL NADU	19.475652	69.920870	134.386087
TELANGANA	12.614783	25.373913	56.173043
UTTARAKHAND	57.272174	55.338261	147.773913
VIDARBHA	11.872174	11.551304	32.851304
WEST MADHYA PRADESH	5.173043	7.657391	15.217391
WEST RAJASTHAN	3.986087	9.443478	17.006087
WEST UTTAR PRADESH	11.461739	12.306087	30.026087

SUBDIVISION	NOV	OCT	Oct-Dec \
ANDAMAN & NICOBAR ISLANDS	233.744444	290.264815	675.416822

ARUNACHAL PRADESH	35.696842	194.686316	254.513830
ASSAM & MEGHALAYA	26.938261	152.118261	188.015652
BIHAR	7.178261	63.074783	73.953913
CHHATTISGARH	11.772174	63.660000	80.674783
COASTAL ANDHRA PRADESH	77.903478	185.511304	274.835652
COASTAL KARNATAKA	63.607826	184.552174	260.775652
EAST MADHYA PRADESH	12.705217	39.686087	60.800000
EAST RAJASTHAN	4.873913	14.360870	22.887826
EAST UTTAR PRADESH	4.590435	42.920870	53.293913
GANGETIC WEST BENGAL	21.579130	115.746087	143.018261
GUJARAT REGION	6.928696	20.565217	28.834783
HARYANA DELHI & CHANDIGARH	3.264348	12.823478	23.270435
HIMACHAL PRADESH	16.695652	31.278261	87.871304
JAMMU & KASHMIR	24.133333	34.166957	113.959649
JHARKHAND	11.923478	80.015652	96.880870
KERALA	163.560000	294.122609	497.636522
KONKAN & GOA	24.671304	113.386957	142.579130
LAKSHADWEEP	124.840741	166.727928	355.387037
MADHYA MAHARASHTRA	25.945217	70.194783	101.986087
MATATHWADA	22.436522	58.580000	88.319130
NAGA MANI MIZO TRIPURA	46.833913	175.006087	234.240000
NORTH INTERIOR KARNATAKA	29.207826	95.688696	131.223478
ORISSA	27.961739	113.592174	147.122609
PUNJAB	4.140000	13.836522	30.669565
RAYALSEEMA	102.653913	135.327826	272.241739
SAURASHTRA & KUTCH	6.096522	14.510435	21.718261
SOUTH INTERIOR KARNATAKA	54.431304	139.143478	205.093913
SUB HIMALAYAN WEST BENGAL & SIKKIM	16.088696	143.646087	165.801739
TAMIL NADU	176.903478	183.196522	441.234783
TELANGANA	20.250435	74.226957	99.620870
UTTARAKHAND	8.187826	39.073913	69.306087
VIDARBHA	15.574783	52.148696	75.654783
WEST MADHYA PRADESH	12.340870	28.086957	46.724348
WEST RAJASTHAN	1.666957	5.127826	8.700870
WEST UTTAR PRADESH	3.966087	28.777391	39.858261

SUBDIVISION	SEP	YEAR
ANDAMAN & NICOBAR ISLANDS	439.482243	1958.918182
ARUNACHAL PRADESH	432.134021	1965.824742
ASSAM & MEGHALAYA	310.734783	1958.000000
BIHAR	217.384348	1958.000000
CHHATTISGARH	217.780000	1958.000000
COASTAL ANDHRA PRADESH	181.707826	1958.000000
COASTAL KARNATAKA	299.652174	1958.000000
EAST MADHYA PRADESH	194.236522	1958.000000
EAST RAJASTHAN	97.978261	1958.000000

```
EAST UTTAR PRADESH                  184.591304  1958.000000
GANGETIC WEST BENGAL                245.710435  1958.000000
GUJARAT REGION                      148.841739  1958.000000
HARYANA DELHI & CHANDIGARH           88.306957  1958.000000
HIMACHAL PRADESH                    130.219130  1958.000000
JAMMU & KASHMIR                      89.289565  1958.000000
JHARKHAND                           227.421739  1958.000000
KERALA                              245.619130  1958.000000
KONKAN & GOA                        349.780000  1958.000000
LAKSHADWEEP                         163.170270  1958.350877
MADHYA MAHARASHTRA                  157.221739  1958.000000
MATATHWADA                          178.476522  1958.000000
NAGA MANI MIZO TRIPURA              314.350435  1958.000000
NORTH INTERIOR KARNATAKA            142.940870  1958.000000
ORISSA                              241.403478  1958.000000
PUNJAB                               86.789565  1958.000000
RAYALSEEMA                          131.720000  1958.000000
SAURASHTRA & KUTCH                   75.418261  1958.000000
SOUTH INTERIOR KARNATAKA            137.313913  1958.000000
SUB HIMALAYAN WEST BENGAL & SIKKIM  421.341739  1958.000000
TAMIL NADU                          111.597391  1958.000000
TELANGANA                           175.503478  1958.000000
UTTARAKHAND                         196.096522  1958.000000
VIDARBHA                            175.449565  1958.000000
WEST MADHYA PRADESH                 161.168696  1958.000000
WEST RAJASTHAN                       40.342609  1958.000000
WEST UTTAR PRADESH                  146.254783  1958.000000
```

[34]:
```
df.groupby(['SUBDIVISION']).count()['YEAR']
```

[34]:
```
SUBDIVISION
ANDAMAN & NICOBAR ISLANDS    110
ARUNACHAL PRADESH             97
ASSAM & MEGHALAYA            115
BIHAR                        115
CHHATTISGARH                 115
COASTAL ANDHRA PRADESH       115
COASTAL KARNATAKA            115
EAST MADHYA PRADESH          115
EAST RAJASTHAN               115
EAST UTTAR PRADESH           115
GANGETIC WEST BENGAL         115
GUJARAT REGION               115
HARYANA DELHI & CHANDIGARH   115
HIMACHAL PRADESH             115
JAMMU & KASHMIR              115
JHARKHAND                    115
```

```
KERALA                                115
KONKAN & GOA                          115
LAKSHADWEEP                           114
MADHYA MAHARASHTRA                    115
MATATHWADA                            115
NAGA MANI MIZO TRIPURA                115
NORTH INTERIOR KARNATAKA              115
ORISSA                                115
PUNJAB                                115
RAYALSEEMA                            115
SAURASHTRA & KUTCH                    115
SOUTH INTERIOR KARNATAKA              115
SUB HIMALAYAN WEST BENGAL & SIKKIM    115
TAMIL NADU                            115
TELANGANA                             115
UTTARAKHAND                           115
VIDARBHA                              115
WEST MADHYA PRADESH                   115
WEST RAJASTHAN                        115
WEST UTTAR PRADESH                    115
Name: YEAR, dtype: int64
```

1 MELT FUNCTION

[38]:
```
import pandas as pd
df = pd.DataFrame(data = {
 'Day' : ['MON', 'TUE', 'WED', 'THU', 'FRI'],
 'Google' : [1129,1132,1134,1152,1152],
 'Apple' : [191,192,190,190,188],
 'Samsung' : [191,192,190,190,188]
})
df
```

[38]:

	Day	Google	Apple	Samsung
0	MON	1129	191	191
1	TUE	1132	192	192
2	WED	1134	190	190
3	THU	1152	190	190
4	FRI	1152	188	188

[39]:
```
reshaped_df = df.melt(id_vars=['Day'])
reshaped_df
```

[39]:

	Day	variable	value
0	MON	Google	1129
1	TUE	Google	1132
2	WED	Google	1134

```
3   THU   Google   1152
4   FRI   Google   1152
5   MON    Apple    191
6   TUE    Apple    192
7   WED    Apple    190
8   THU    Apple    190
9   FRI    Apple    188
10  MON  Samsung    191
11  TUE  Samsung    192
12  WED  Samsung    190
13  THU  Samsung    190
14  FRI  Samsung    188
```

[23]:
```
reshaped_df.columns
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[23], line 1
----> 1 reshaped_df.columns

NameError: name 'reshaped_df' is not defined
```

[41]:
```
reshaped_df.columns = [['Day', 'Company', 'Closing Price']]
reshaped_df
```

[41]:
```
    Day  Company Closing Price
0   MON   Google          1129
1   TUE   Google          1132
2   WED   Google          1134
3   THU   Google          1152
4   FRI   Google          1152
5   MON    Apple           191
6   TUE    Apple           192
7   WED    Apple           190
8   THU    Apple           190
9   FRI    Apple           188
10  MON  Samsung           191
11  TUE  Samsung           192
12  WED  Samsung           190
13  THU  Samsung           190
14  FRI  Samsung           188
```

[42]:
```
reshaped_df = df.melt(id_vars=['Day'], var_name='Company', value_name='Closing␣
↪Price')
reshaped_df
```

[42]:

	Day	Company	Closing Price
0	MON	Google	1129
1	TUE	Google	1132
2	WED	Google	1134
3	THU	Google	1152
4	FRI	Google	1152
5	MON	Apple	191
6	TUE	Apple	192
7	WED	Apple	190
8	THU	Apple	190
9	FRI	Apple	188
10	MON	Samsung	191
11	TUE	Samsung	192
12	WED	Samsung	190
13	THU	Samsung	190
14	FRI	Samsung	188

[43]:
```
reshaped_df.pivot(index='Day', columns='Company')
```

[43]:

	Closing Price		
Company	Apple	Google	Samsung
Day			
FRI	188	1152	188
MON	191	1129	191
THU	190	1152	190
TUE	192	1132	192
WED	190	1134	190

[45]:
```
original_df = reshaped_df.pivot(index='Day', columns='Company', values='Closing␣
↪Price').reset_index()
original_df.columns.name = None
original_df
```

[45]:

	Day	Apple	Google	Samsung
0	FRI	188	1152	188
1	MON	191	1129	191
2	THU	190	1152	190
3	TUE	192	1132	192
4	WED	190	1134	190

[46]:
```
import pandas as pd
import numpy as np
technologies= {
 'Duration':['30days','50days','30days','35days','40days'],
 'Fee' :[22000,25000,23000,np.NaN,26000]
 }
df = pd.DataFrame(technologies)
```

```
print(df)
```

```
  Duration      Fee
0   30days  22000.0
1   50days  25000.0
2   30days  23000.0
3   35days      NaN
4   40days  26000.0
```

[47]:
```
df['Fee'] = df['Fee'].map(lambda x: x - (x*10/100))
print(df)
```

```
  Duration      Fee
0   30days  19800.0
1   50days  22500.0
2   30days  20700.0
3   35days      NaN
4   40days  23400.0
```

[48]:
```
def fun1(x):
 return x/100
df['Discount'] = df['Fee'].map(lambda x:fun1(x))
print(df)
```

```
  Duration      Fee  Discount
0   30days  19800.0     198.0
1   50days  22500.0     225.0
2   30days  20700.0     207.0
3   35days      NaN       NaN
4   40days  23400.0     234.0
```

[49]:
```
df['Service'] = df['Fee'].map(lambda x: x - (x*10/100))
df
```

[49]:
```
   Duration      Fee  Discount  Service
0   30days  19800.0     198.0  17820.0
1   50days  22500.0     225.0  20250.0
2   30days  20700.0     207.0  18630.0
3   35days      NaN       NaN      NaN
4   40days  23400.0     234.0  21060.0
```

[50]:
```
df['Fee'] = df['Fee'].map('{} RS'.format)
df['Discount'] = df['Discount'].map('{} RS'.format)
print(df)
```

```
  Duration         Fee  Discount  Service
0   30days  19800.0 RS  198.0 RS  17820.0
1   50days  22500.0 RS  225.0 RS  20250.0
2   30days  20700.0 RS  207.0 RS  18630.0
```

```
3   35days      nan RS    nan RS      NaN
4   40days  23400.0 RS  234.0 RS  21060.0
```

[51]:
```
df['Service'] = df['Service'].map('{} RS'.format, na_action='ignore')
print(df)
```

```
  Duration         Fee  Discount     Service
0   30days  19800.0 RS  198.0 RS  17820.0 RS
1   50days  22500.0 RS  225.0 RS  20250.0 RS
2   30days  20700.0 RS  207.0 RS  18630.0 RS
3   35days      nan RS    nan RS         NaN
4   40days  23400.0 RS  234.0 RS  21060.0 RS
```

[55]:
```
import numpy as np
import pandas as pd

marks = {
    'Chemistry': [67, 90, 66, 32, 72, 45, 60, 98],
    'Physics': [45, 92, 72, 92, 72, 34, 72, 45]
}

# Complete the list of indices
indices = ['Subodh', 'Ram', 'Abdul', 'John', 'Nandini', 'Zoya', 'Amit', 'Ravi']

# Create the DataFrame
marks_df = pd.DataFrame(marks, index=indices)

# Display the DataFrame
print(marks_df)
```

```
         Chemistry  Physics
Subodh          67       45
Ram             90       92
Abdul           66       72
John            32       92
Nandini         72       72
Zoya            45       34
Amit            60       72
Ravi            98       45
```

[56]:
```
marks_df.Chemistry.sum()
```

[56]: 530

[57]:
```
marks_df['Chemistry'].sum()
```

[57]: 530

[58]:
```
marks_df['Physics'].mean()
```

[58]:
```
65.5
```

[59]:
```
marks_df.mean()
```

[59]:
```
Chemistry    66.25
Physics      65.50
dtype: float64
```

[60]:
```
marks_df.sum()
```

[60]:
```
Chemistry    530
Physics      524
dtype: int64
```

[61]:
```
marks_df.count()
```

[61]:
```
Chemistry    8
Physics      8
dtype: int64
```

[62]:
```
marks_df.agg(['min', 'max','sum','mean','median'])
```

[62]:

	Chemistry	Physics
min	32.00	34.0
max	98.00	92.0
sum	530.00	524.0
mean	66.25	65.5
median	66.50	72.0

[63]:
```
print(marks_df)
marks_df.groupby("Physics").max()
```

	Chemistry	Physics
Subodh	67	45
Ram	90	92
Abdul	66	72
John	32	92
Nandini	72	72
Zoya	45	34
Amit	60	72
Ravi	98	45

[63]:

Physics	Chemistry
34	45
45	98
72	72
92	90

[70]:
```
import pandas as pd
from datetime import datetime
import numpy as np
```

[71]:
```
range_date1 = pd.date_range(start ='1/1/2019', end ='1/08/2019',freq='D') #days
print(range_date1)
```

```
DatetimeIndex(['2019-01-01', '2019-01-02', '2019-01-03', '2019-01-04',
               '2019-01-05', '2019-01-06', '2019-01-07', '2019-01-08'],
              dtype='datetime64[ns]', freq='D')
```

[ ]:
```
range_date3 = pd.date_range(start ='1/1/2019', end ='1/08/2020',freq='M')␣
↪#months
print(range_date3)
```

[72]:
```
range_date2 = pd.date_range(start ='1/1/2019', end ='1/02/2019',freq='H') #hours
print(range_date2)
```

```
DatetimeIndex(['2019-01-01 00:00:00', '2019-01-01 01:00:00',
               '2019-01-01 02:00:00', '2019-01-01 03:00:00',
               '2019-01-01 04:00:00', '2019-01-01 05:00:00',
               '2019-01-01 06:00:00', '2019-01-01 07:00:00',
               '2019-01-01 08:00:00', '2019-01-01 09:00:00',
               '2019-01-01 10:00:00', '2019-01-01 11:00:00',
               '2019-01-01 12:00:00', '2019-01-01 13:00:00',
               '2019-01-01 14:00:00', '2019-01-01 15:00:00',
               '2019-01-01 16:00:00', '2019-01-01 17:00:00',
               '2019-01-01 18:00:00', '2019-01-01 19:00:00',
               '2019-01-01 20:00:00', '2019-01-01 21:00:00',
               '2019-01-01 22:00:00', '2019-01-01 23:00:00',
               '2019-01-02 00:00:00'],
              dtype='datetime64[ns]', freq='H')
```

[73]:
```
range_date4= pd.date_range(start ='1/1/2019', end ='1/08/2020',freq='3M')␣
↪#3months
print(range_date4)
```

```
DatetimeIndex(['2019-01-31', '2019-04-30', '2019-07-31', '2019-10-31'],
dtype='datetime64[ns]', freq='3M')
```

[74]:
```
range_date5 = pd.date_range(start ='1/1/2019', end ='1/08/2020',freq=None)␣
↪#days by default
print(range_date5)
```

```
DatetimeIndex(['2019-01-01', '2019-01-02', '2019-01-03', '2019-01-04',
               '2019-01-05', '2019-01-06', '2019-01-07', '2019-01-08',
               '2019-01-09', '2019-01-10',
               …
               '2019-12-30', '2019-12-31', '2020-01-01', '2020-01-02',
```

```
               '2020-01-03', '2020-01-04', '2020-01-05', '2020-01-06',
               '2020-01-07', '2020-01-08'],
              dtype='datetime64[ns]', length=373, freq='D')
```

[75]:
```
range_date6= pd.date_range(start ='1/1/2019', end ='1/2/2019',freq='min')␣
  ↪#minutes
print(range_date6)
```

```
DatetimeIndex(['2019-01-01 00:00:00', '2019-01-01 00:01:00',
               '2019-01-01 00:02:00', '2019-01-01 00:03:00',
               '2019-01-01 00:04:00', '2019-01-01 00:05:00',
               '2019-01-01 00:06:00', '2019-01-01 00:07:00',
               '2019-01-01 00:08:00', '2019-01-01 00:09:00',
               …
               '2019-01-01 23:51:00', '2019-01-01 23:52:00',
               '2019-01-01 23:53:00', '2019-01-01 23:54:00',
               '2019-01-01 23:55:00', '2019-01-01 23:56:00',
               '2019-01-01 23:57:00', '2019-01-01 23:58:00',
               '2019-01-01 23:59:00', '2019-01-02 00:00:00'],
              dtype='datetime64[ns]', length=1441, freq='T')
```

[76]:
```
range_date7 = pd.date_range(start ='1/1/2018', periods = 13)
print(range_date7)
```

```
DatetimeIndex(['2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04',
               '2018-01-05', '2018-01-06', '2018-01-07', '2018-01-08',
               '2018-01-09', '2018-01-10', '2018-01-11', '2018-01-12',
               '2018-01-13'],
              dtype='datetime64[ns]', freq='D')
```

[77]:
```
range_date7 = pd.date_range(end ='1/13/2018', periods = 13)
print(range_date7)
```

```
DatetimeIndex(['2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04',
               '2018-01-05', '2018-01-06', '2018-01-07', '2018-01-08',
               '2018-01-09', '2018-01-10', '2018-01-11', '2018-01-12',
               '2018-01-13'],
              dtype='datetime64[ns]', freq='D')
```

[78]:
```
print(type(range_date1[1]))
```

```
<class 'pandas._libs.tslibs.timestamps.Timestamp'>
```

[81]:
```
range_data = pd.date_range(start ='1/1/2019', end ='1/08/2019',
 freq ='D')
df = pd.DataFrame(range_data, columns =['date'])
df['values'] = np.random.randint(0, 100, size =(len(range_data)))
print(df.head(10))
```

```
        date  values
```

```
0 2019-01-01      33
1 2019-01-02      62
2 2019-01-03      63
3 2019-01-04      73
4 2019-01-05      90
5 2019-01-06      53
6 2019-01-07      72
7 2019-01-08      13
```

[82]:
```
df['datetime'] = pd.to_datetime(df['date'])
df = df.set_index('datetime')
df.drop(['date'], axis = 1, inplace = True)
df
```

[82]:
```
            values
datetime
2019-01-01      33
2019-01-02      62
2019-01-03      63
2019-01-04      73
2019-01-05      90
2019-01-06      53
2019-01-07      72
2019-01-08      13
```

[83]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm
```

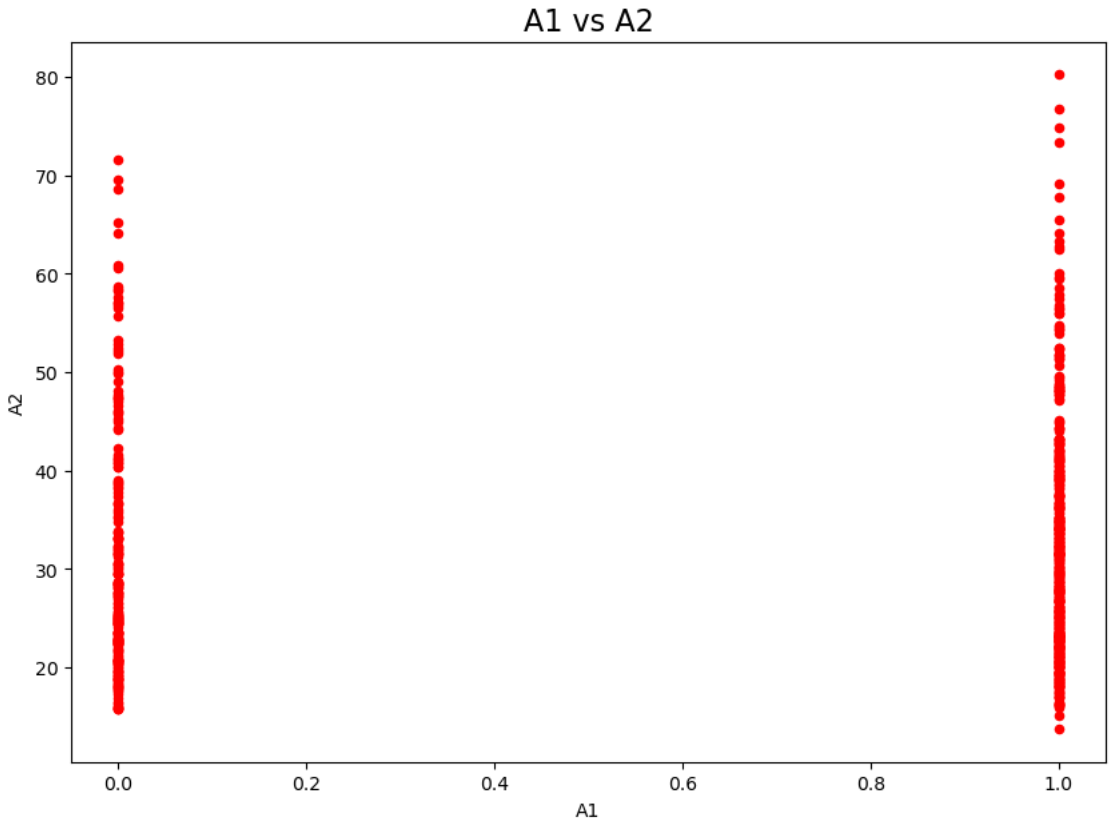
[84]:
```
credit_df = pd.read_csv("C:/Users/HP/Downloads/Credit Card.csv")
credit_df
```

[84]:
```
     CustomerID  A1     A2      A3  A4  A5  A6     A7  A8  A9  A10  A11  A12  \
0      15776156   1  22.08  11.460   2   4   4  1.585   0   0    0    1    2
1      15739548   0  22.67   7.000   2   8   4  0.165   0   0    0    0    2
2      15662854   0  29.58   1.750   1   4   4  1.250   0   0    0    1    2
3      15687688   0  21.67  11.500   1   5   3  0.000   1   1   11    1    2
4      15715750   1  20.17   8.170   2   6   4  1.960   1   1   14    0    2
..          ...  ..    ...     ...  ..  ..  ..    ...  ..  ..  ...  ...  ...
685    15808223   1  31.57  10.500   2  14   4  6.500   1   0    0    0    2
686    15769980   1  20.67   0.415   2   8   4  0.125   0   0    0    0    2
687    15675450   0  18.83   9.540   2   6   4  0.085   1   0    0    0    2
688    15776494   0  27.42  14.500   2  14   8  3.085   1   1    1    0    2
689    15592412   1  41.00   0.040   2  10   4  0.040   0   1    1    0    1

     A13   A14  Class
0    100  1213      0
```

```
1     160     1       0
2     280     1       0
3       0     1       1
4      60   159       1
..    ...   ...     ...
685     0     1       1
686     0    45       0
687   100     1       1
688   120    12       1
689   560     1       1

[690 rows x 16 columns]
```

[87]:
```
ax = credit_df.plot("A1","A2",kind="scatter", color = "red",marker =␣
 ↪"o",figsize=(10,7))
#To add labels and title to the output
ax.set_xlabel("A1") #sets label for x-axis
ax.set_ylabel("A2") #sets label for y-axis
ax.set_title("A1 vs A2",fontsize=16)
```

[87]:
```
Text(0.5, 1.0, 'A1 vs A2')
```

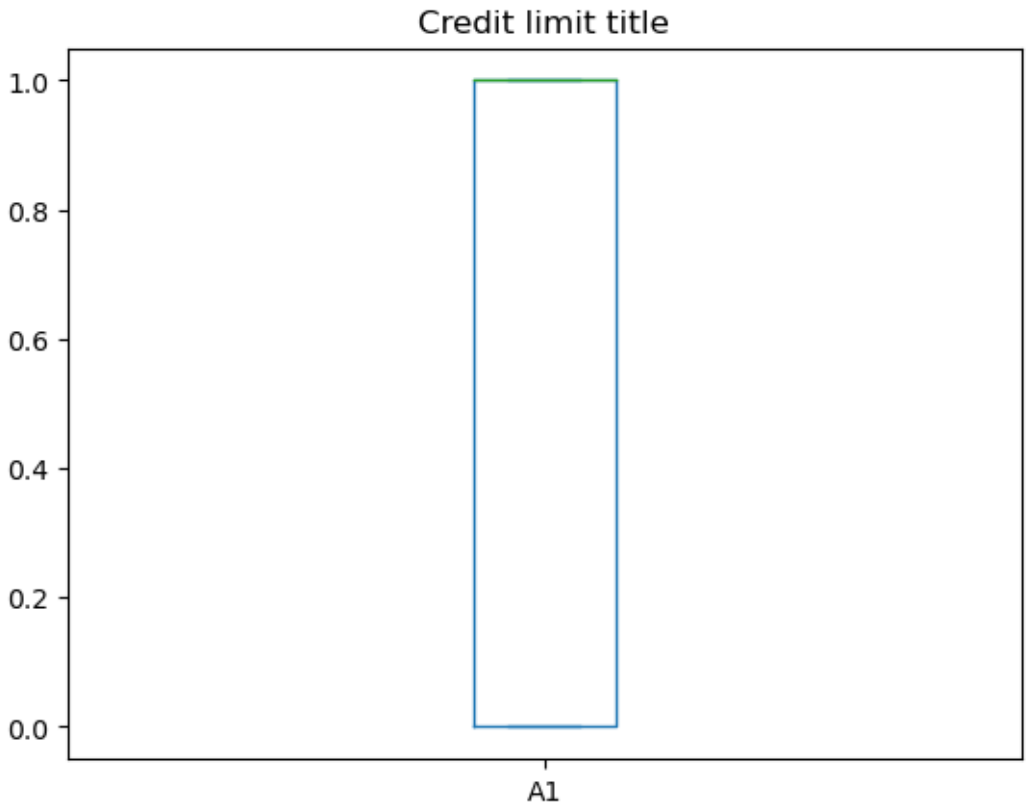


[89]:
```
credit_df["A1"].describe()
```

[89]:
```
count    690.000000
mean       0.678261
std        0.467482
min        0.000000
25%        0.000000
50%        1.000000
75%        1.000000
max        1.000000
Name: A1, dtype: float64
```

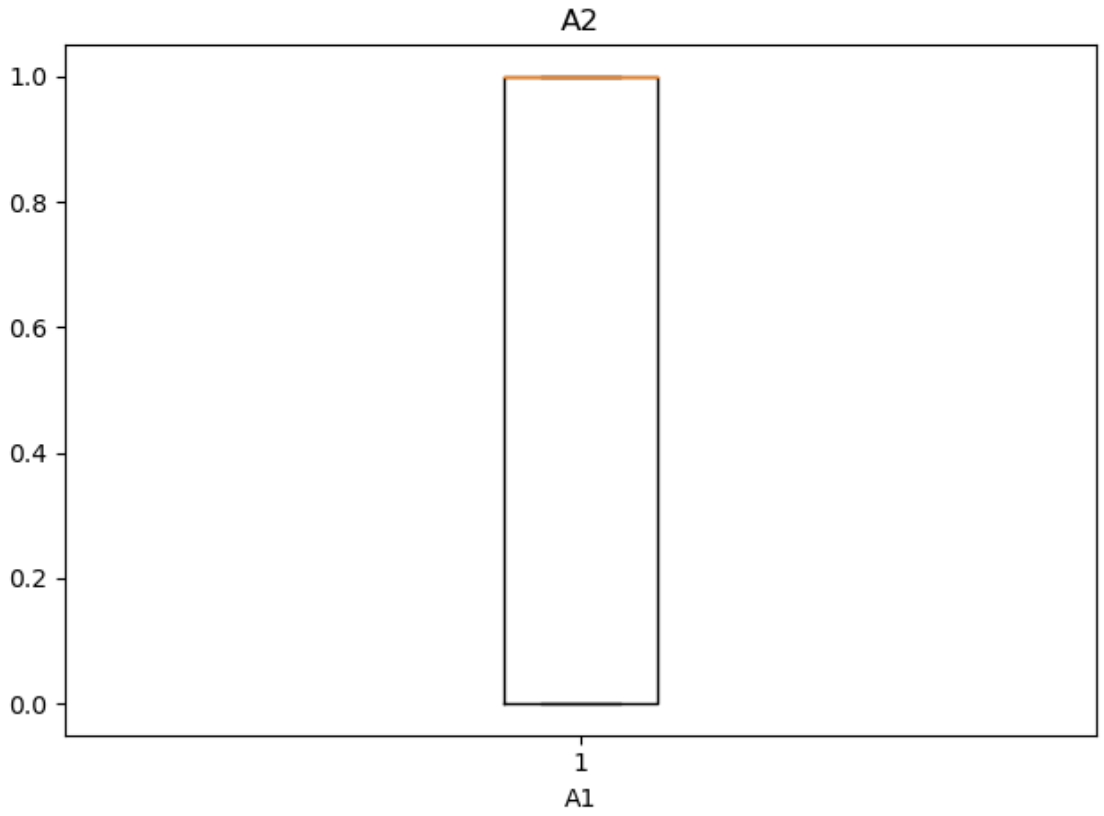
[90]:
```
ax = credit_df["A1"].plot(kind="box")
ax.set_title("Credit limit title")
```

[90]:
```
Text(0.5, 1.0, 'Credit limit title')
```



[92]:
```
fig, ax1 = plt.subplots(1, 1)
#The following lines of code change the alignment from vertical to horizontal
ax1.boxplot(credit_df["A1"])
```

```
#The following lines of code are used to add labels to axes and title to the␣
 ↪graph
ax1.set_title('A2')
ax1.set_xlabel('A1')
#In case of any superimposition of the subplots, the following functions caters␣
 ↪the aesthetics
fig.tight_layout()
```



[ ]:

2 WEEK 4

[6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm
```

[10]:
```
credit_df = pd.read_csv("C:/Users/HP/Downloads/Credit card.csv")
credit_df
```

[10]:

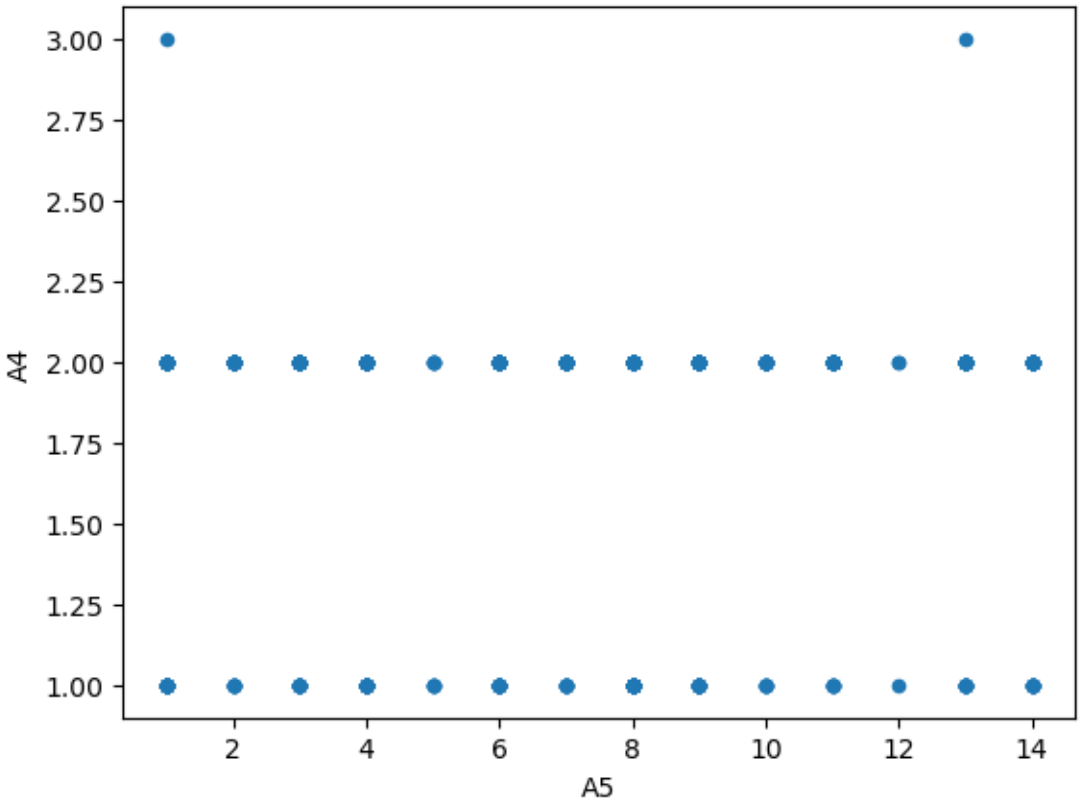
	CustomerID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	\
0	15776156	1	22.08	11.460	2	4	4	1.585	0	0	0	1	2	
1	15739548	0	22.67	7.000	2	8	4	0.165	0	0	0	0	2	
2	15662854	0	29.58	1.750	1	4	4	1.250	0	0	0	1	2	
3	15687688	0	21.67	11.500	1	5	3	0.000	1	1	11	1	2	
4	15715750	1	20.17	8.170	2	6	4	1.960	1	1	14	0	2	
..	…	..	…	…	..	..	..	…	..	..	…	…	…	
685	15808223	1	31.57	10.500	2	14	4	6.500	1	0	0	0	2	
686	15769980	1	20.67	0.415	2	8	4	0.125	0	0	0	0	2	
687	15675450	0	18.83	9.540	2	6	4	0.085	1	0	0	0	2	
688	15776494	0	27.42	14.500	2	14	8	3.085	1	1	1	0	2	
689	15592412	1	41.00	0.040	2	10	4	0.040	0	1	1	0	1	

	A13	A14	Class
0	100	1213	0
1	160	1	0
2	280	1	0
3	0	1	1
4	60	159	1
..	…	…	…
685	0	1	1
686	0	45	0
687	100	1	1
688	120	12	1
689	560	1	1

[690 rows x 16 columns]

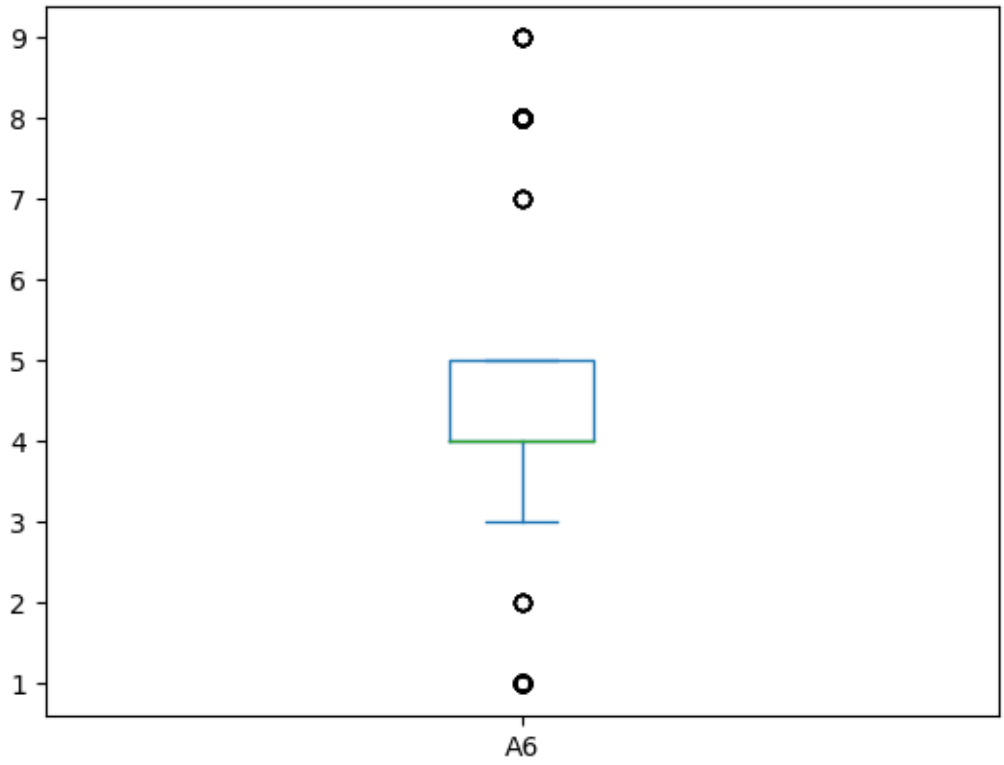
[13]:
```
credit_df.plot('A5','A4',kind='scatter',marker='o')
```

[13]: <Axes: xlabel='A5', ylabel='A4'>



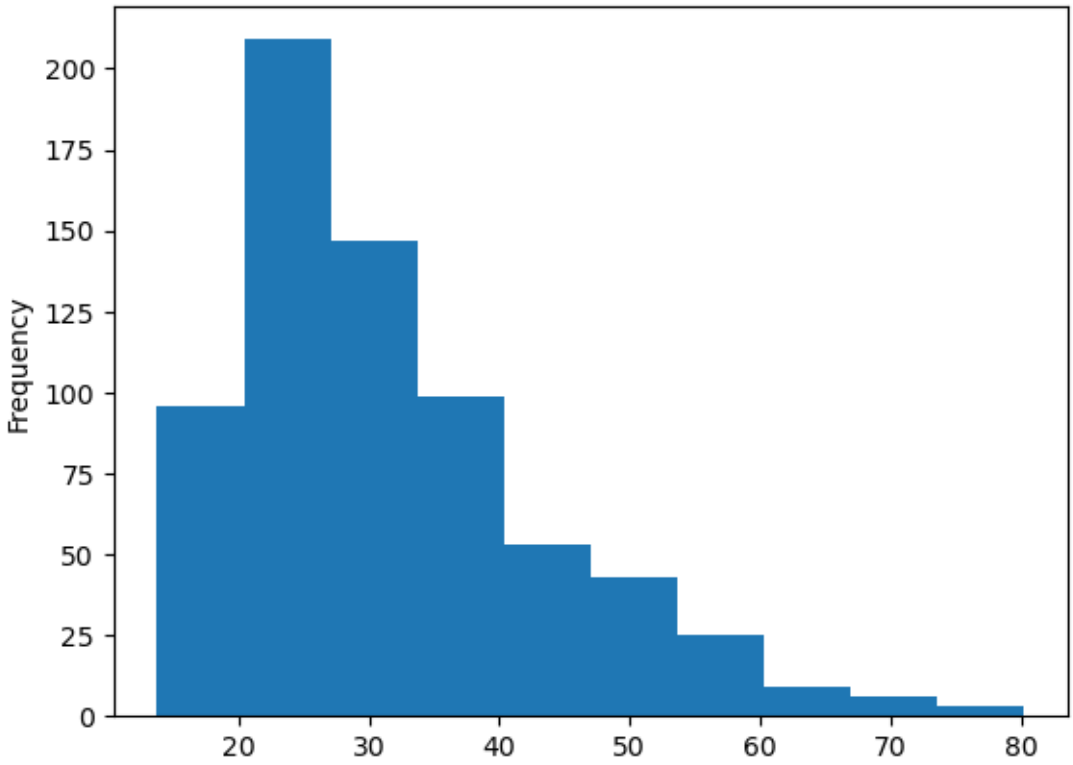
[14]:
```
credit_df['A6'].plot(kind='box')
```

[14]: <Axes: >



[15]:
```
credit_df['A2'].plot(kind='hist')
```

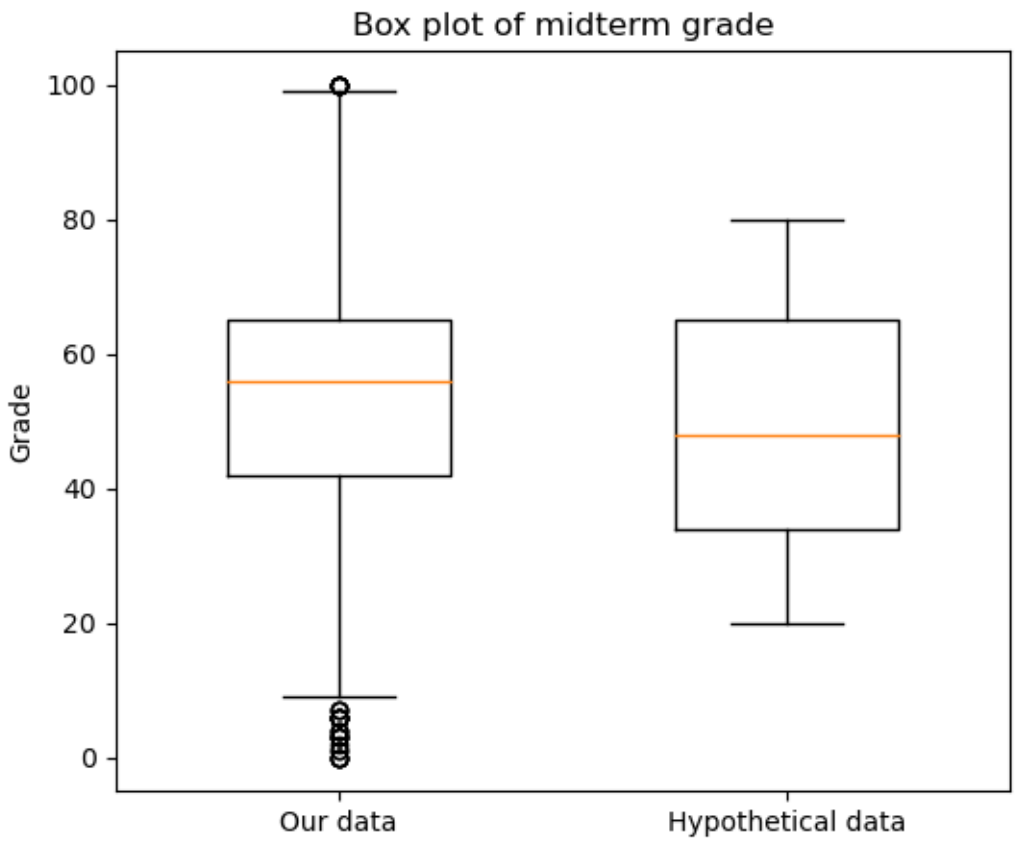
[15]: <Axes: ylabel='Frequency'>



[24]:
```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(102)
grades = np.concatenate([[50,52,53,55,56,60,61,62,65,67]*20,
np.random.randint(0, 101, size=300)])
Q1 = np.percentile(grades , 25)
Q3 = np.percentile(grades , 75)
Q1,Q3 = np.percentile(grades , [25,75])
IQR = Q3 - Q1
ul = Q3+1.5*IQR
ll = Q1-1.5*IQR
outliers = grades[(grades > ul) | (grades < ll)]
print(outliers)
fig = plt.figure(figsize=(6,5))
hypo = np.random.randint(20, 81, size=500)
plt.boxplot([grades, hypo], widths=0.5)
plt.xticks([1,2],['Our data', 'Hypothetical data'])
plt.ylabel('Grade')
plt.title('Box plot of midterm grade')
plt.show()
```

```
[  0   7   4   3   0   4   2   7   6 100   1   3   0   3 100 100 100 100
```

```
 4   0   3   6   6   6 100   7   6 100 100   6   3   6   1   6   0]
```



[27]:
```python
import numpy as np

# Data
data = [1, 2, 2, 2, 3, 1, 1, 15, 2, 2, 2, 3, 1, 1, 2]

# Calculate mean and standard deviation
mean = np.mean(data)
std = np.std(data)

# Print mean and standard deviation
print('Mean of the dataset is', mean)
print('Standard deviation is', std)

# Define threshold for Z-score
threshold = 3

# Identify outliers
outliers = []
```

```
for i in data:
    z = (i - mean) / std
    if z > threshold:
        outliers.append(i)

# Print outliers
print('Outliers in the dataset based on Z-score are', outliers)
```

```
Mean of the dataset is 2.6666666666666665
Standard deviation is 3.3598941782277745
Outliers in the dataset based on Z-score are [15]
```

[32]:
```
import pandas as pd
# Creating the DataFrame
df = pd.DataFrame({'Date':['10/2/2011', '11/2/2011', '12/2/2011', '13/2/2011'],
 'Event':['Music', 'Poetry', 'Theatre', 'Comedy'],
 'Cost':[10000, 5000, 15000, 2000]})
# Print the dataframe
print(df)
```

```
        Date    Event   Cost
0  10/2/2011    Music  10000
1  11/2/2011   Poetry   5000
2  12/2/2011  Theatre  15000
3  13/2/2011   Comedy   2000
```

[33]:
```
df['Discounted_Price'] = df.apply(lambda row: row.Cost * 0.9, axis = 1)
# Print the DataFrame after addition of new column
print(df)
```

```
        Date    Event   Cost  Discounted_Price
0  10/2/2011    Music  10000            9000.0
1  11/2/2011   Poetry   5000            4500.0
2  12/2/2011  Theatre  15000           13500.0
3  13/2/2011   Comedy   2000            1800.0
```

[34]:
```
df = pd.DataFrame({'Name':['John','Ted','Dove','Brad','Rex'],
 'Salary':[44000, 35000, 75000, 20000,6000]})
# Print the dataframe
print(df)
```

```
   Name  Salary
0  John   44000
1   Ted   35000
2  Dove   75000
3  Brad   20000
4   Rex    6000
```

[37]:
```
def salary_stats(value):
 if value < 10000:
   return "very low"
 elif 10000 <= value < 25000:
   return "low"
 elif 25000 <= value < 40000:
   return "average"
 elif 40000 <= value < 50000:
   return "better"
 elif value >= 50000:
   return "very good"

df['salary_stats'] = df['Salary'].map(salary_stats)
df
```

[37]:

	Name	Salary	salary_stats
0	John	44000	better
1	Ted	35000	average
2	Dove	75000	very good
3	Brad	20000	low
4	Rex	6000	very low

[66]:
```
import pandas as pd

# Create the DataFrame
data = pd.DataFrame({
    'Name': ['A', 'B', 'C', 'D', 'E', 'F'],
    'Education': ['High School', 'Masters', 'Doctorate', 'Bachelors',␣
 ↪'Masters', None]})

# Display the DataFrame
data
```

[66]:

	Name	Education
0	A	High School
1	B	Masters
2	C	Doctorate
3	D	Bachelors
4	E	Masters
5	F	None

[42]:
```
education_data = pd.get_dummies(data.Education)
print(education_data)
```

	Bachelors	Doctorate	High School	Masters
0	False	False	True	False
1	False	False	False	True
2	False	True	False	False

```
3       True      False        False    False
4      False      False        False     True
5      False      False        False    False
```

[43]:
```
education_map = {
 'High School' : 1,
 'Bachelors' : 2,
 'Masters': 3,
 'Doctorate': 4
}
education_data = data['Education'].map(education_map)
data['Education'] = education_data
data
```

[43]:
```
  Name  Education
0    A        1.0
1    B        3.0
2    C        4.0
3    D        2.0
4    E        3.0
5    F        NaN
```

[44]:
```
education_map = {
 'High School' : 12,
 'Bachelors' : 16,
 'Masters': 18,
 'Doctorate': 21
}
education_data = data['Education'].map(education_map)
data['Education'] = education_data
data
```

[44]:
```
  Name  Education
0    A        NaN
1    B        NaN
2    C        NaN
3    D        NaN
4    E        NaN
5    F        NaN
```

[45]:
```
df.loc[len(df.index)]=['Hruthvik', 15000, 'low']
df
```

[45]:
```
        Name  Salary salary_stats
0       John   44000       better
1        Ted   35000      average
2       Dove   75000    very good
3       Brad   20000          low
```

```
4       Rex    6000     very low
5  Hruthvik   15000          low
```

[47]:
```
import pandas as pd
d1 = {'Name': ['Pankaj', 'Meghna', 'Lisa'], 'Country': ['India', 'India',␣
 ↪'USA']}
df1 = pd.DataFrame(d1)
print('DataFrame 1:\n', df1,'\n')
df2 = pd.DataFrame({'ID': [1, 2, 3], 'Name': ['Pankaj', 'Anupam', 'Amit']})
print('DataFrame 2:\n', df2,'\n')
df3 = pd.DataFrame({'Name': ['Priya'], 'Country': ['India'], 'Role': ['COO']})
print('DataFrame 3:\n', df3,'\n')
```

```
DataFrame 1:
      Name Country
0  Pankaj   India
1  Meghna   India
2    Lisa     USA

DataFrame 2:
    ID    Name
0   1  Pankaj
1   2  Anupam
2   3    Amit

DataFrame 3:
     Name Country Role
0  Priya   India  COO
```

[48]:
```
same_cols_df = pd.concat([df1,df3],ignore_index=True)
same_cols_df
```

[48]:
```
      Name Country Role
0  Pankaj   India  NaN
1  Meghna   India  NaN
2    Lisa     USA  NaN
3   Priya   India  COO
```

[50]:
```
c_df = pd.concat([df1,df2],ignore_index=True)
c_df
```

[50]:
```
      Name Country   ID
0  Pankaj   India  NaN
1  Meghna   India  NaN
2    Lisa     USA  NaN
3  Pankaj     NaN  1.0
4  Anupam     NaN  2.0
```

```
5    Amit     NaN  3.0
```

[51]:
```
df_merged = df1.merge(df2)
print('Result:\n', df_merged)
```

```
Result:
      Name Country  ID
0  Pankaj   India   1
```

[57]:
```
import pandas as pd
import numpy as np
df = pd.read_csv('C:/Users/HP/Downloads/auto-mpg.csv')
df.head()
```

[57]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model-year
0	18.0	8	307.0	130.0	3504	12.0	70
1	15.0	8	350.0	165.0	3693	11.5	70
2	18.0	8	318.0	150.0	3436	11.0	70
3	16.0	8	304.0	150.0	3433	12.0	70
4	17.0	8	302.0	140.0	3449	10.5	70

[58]:
```
df.loc[(df['mpg'] > 29) & (df['horsepower'] < 93.5) & (df['weight'] < 2500)]
```

[58]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	\
51	30.0	4	79.0	70.0	2074	19.5	
52	30.0	4	88.0	76.0	2065	14.5	
53	31.0	4	71.0	65.0	1773	19.0	
54	35.0	4	72.0	69.0	1613	18.0	
129	31.0	4	79.0	67.0	1950	19.0	
..	…	…	…	…	…	…	
384	32.0	4	91.0	67.0	1965	15.7	
385	38.0	4	91.0	67.0	1995	16.2	
391	36.0	4	135.0	84.0	2370	13.0	
394	44.0	4	97.0	52.0	2130	24.6	
395	32.0	4	135.0	84.0	2295	11.6	

	model-year
51	71
52	71
53	71
54	71
129	74
..	…
384	82
385	82
391	82

```
394          82
395          82

[83 rows x 7 columns]
```

[65]:
```
df.loc[(df['displacement'] > 262) & (df['horsepower'] > 126) &␣
↪(df['weight'] >=2800) & (df['weight']<=3800)]
```

[65]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model-year
0	18.0	8	307.0	130.0	3504	12.0	70
1	15.0	8	350.0	165.0	3693	11.5	70
2	18.0	8	318.0	150.0	3436	11.0	70
3	16.0	8	304.0	150.0	3433	12.0	70
4	17.0	8	302.0	140.0	3449	10.5	70
10	15.0	8	383.0	170.0	3563	10.0	70
11	14.0	8	340.0	160.0	3609	8.0	70
12	15.0	8	400.0	150.0	3761	9.5	70
13	14.0	8	455.0	225.0	3086	10.0	70
66	17.0	8	304.0	150.0	3672	11.5	72
86	14.0	8	304.0	150.0	3672	11.5	73
89	15.0	8	318.0	150.0	3777	12.5	73
121	15.0	8	318.0	150.0	3399	11.0	73
124	11.0	8	350.0	180.0	3664	11.0	73
166	13.0	8	302.0	129.0	3169	12.0	75
215	13.0	8	318.0	150.0	3755	14.0	77
250	19.4	8	318.0	140.0	3735	13.2	78
251	20.2	8	302.0	139.0	3570	12.8	78
262	19.2	8	305.0	145.0	3425	13.2	78
264	18.1	8	302.0	139.0	3205	11.2	78
286	17.6	8	302.0	129.0	3725	13.4	78

215	76
250	78
251	78
262	78
264	78
286	79