Lab Report # 06 & 07

Fall 2021

CSE-307L Data Analytics Lab

Submitted by: **Shah Raza**
Registration No. **18PWCSE1658**

Section: **B**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature:

Submitted to:
Engr. Mian Ibad Ali Shah
Friday, February 11, 2022

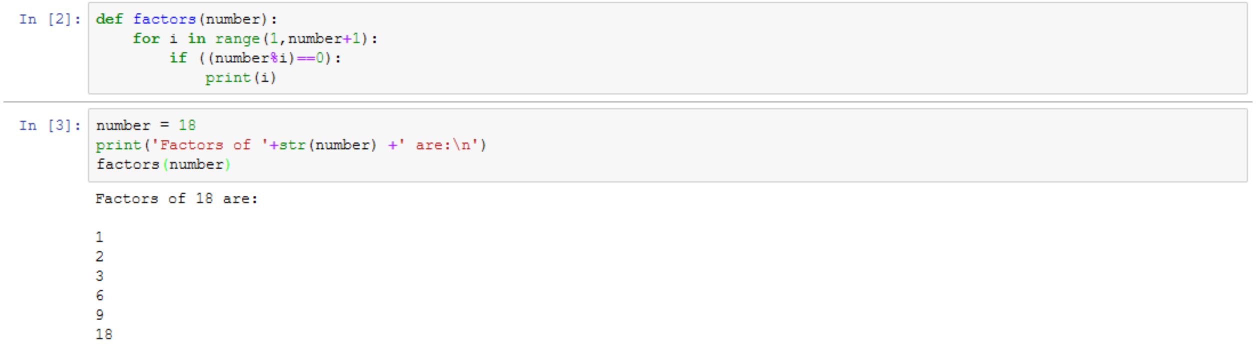
Department of Computer Systems Engineering
University of Engineering and Technology Peshawar

Python Assessment (Basic & Libraries):

Task 01:

1) Using function, find all the multiplicative factors of a number provided as input.

Code:



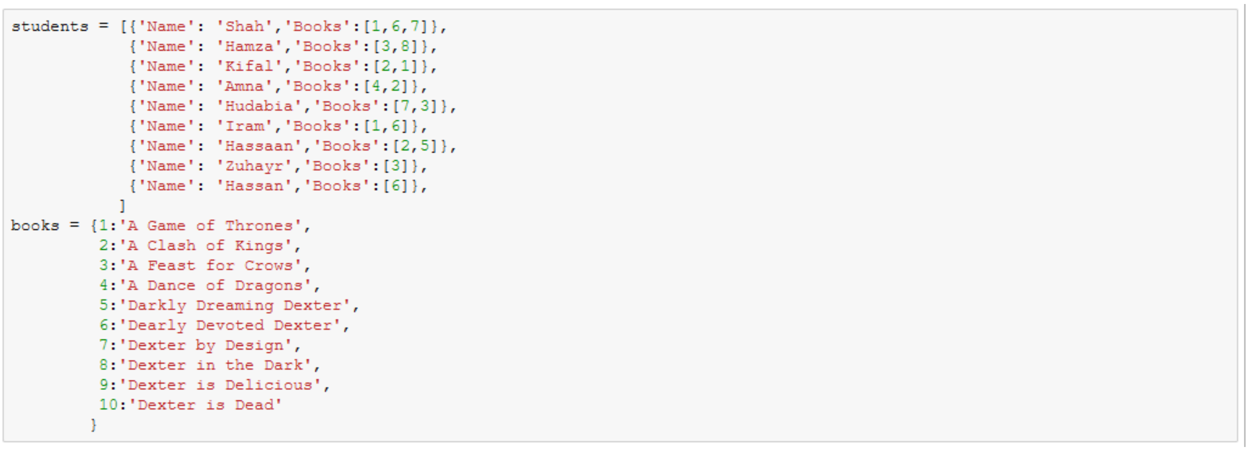
```
In [2]: def factors(number):
            for i in range(1,number+1):
                if ((number%i)==0):
                    print(i)
```

```
In [3]: number = 18
        print('Factors of '+str(number) +' are:\n')
        factors(number)
```

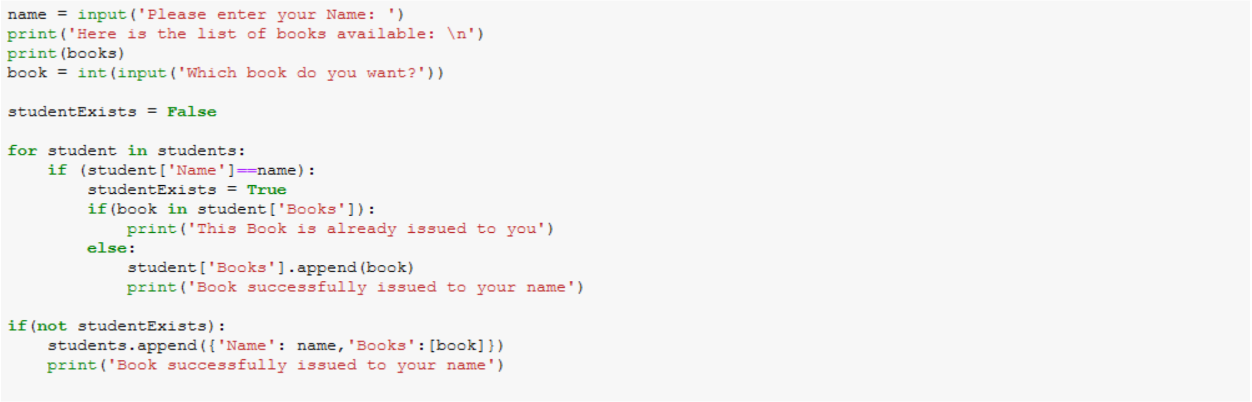
```
Factors of 18 are:

1
2
3
6
9
18
```

2) Create a library system for DCSE using dictionaries (must include Student's information, Books information)

Code:

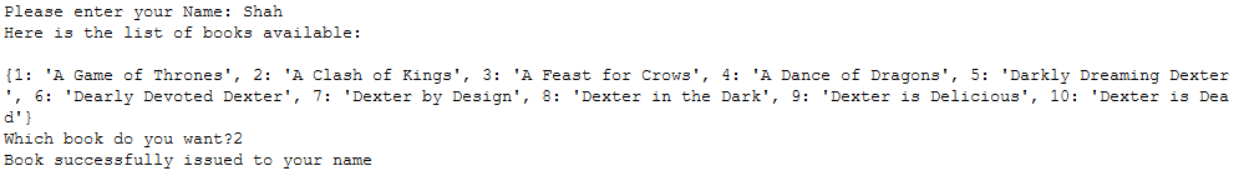


```
students = [{'Name': 'Shah','Books':[1,6,7]},
            {'Name': 'Hamza','Books':[3,8]},
            {'Name': 'Kifal','Books':[2,1]},
            {'Name': 'Amna','Books':[4,2]},
            {'Name': 'Hudabia','Books':[7,3]},
            {'Name': 'Iram','Books':[1,6]},
            {'Name': 'Hassaan','Books':[2,5]},
            {'Name': 'Zuhayr','Books':[3]},
            {'Name': 'Hassan','Books':[6]},
           ]
books = {1:'A Game of Thrones',
         2:'A Clash of Kings',
         3:'A Feast for Crows',
         4:'A Dance of Dragons',
         5:'Darkly Dreaming Dexter',
         6:'Dearly Devoted Dexter',
         7:'Dexter by Design',
         8:'Dexter in the Dark',
         9:'Dexter is Delicious',
         10:'Dexter is Dead'
        }
```



```
name = input('Please enter your Name: ')
print('Here is the list of books available: \n')
print(books)
book = int(input('Which book do you want?'))

studentExists = False

for student in students:
    if (student['Name']==name):
        studentExists = True
        if(book in student['Books']):
            print('This Book is already issued to you')
        else:
            student['Books'].append(book)
            print('Book successfully issued to your name')

if(not studentExists):
    students.append({'Name': name,'Books':[book]})
    print('Book successfully issued to your name')
```

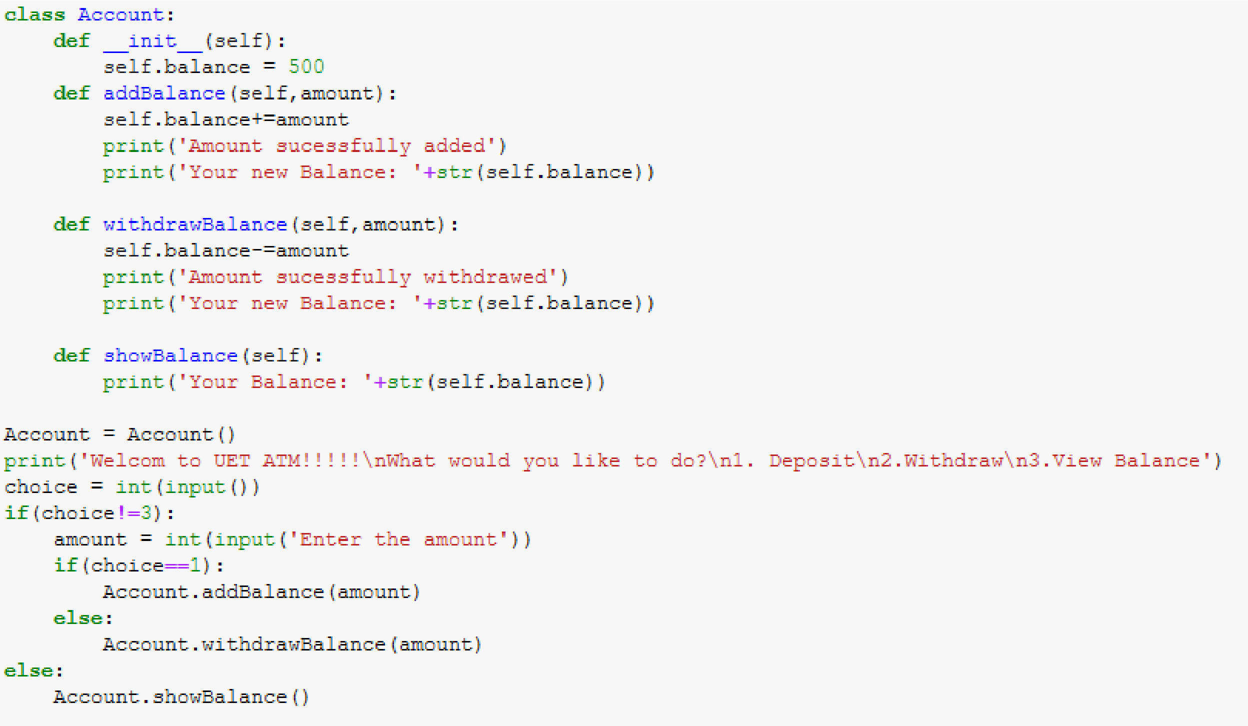
Output:



```
Please enter your Name: Shah
Here is the list of books available:

{1: 'A Game of Thrones', 2: 'A Clash of Kings', 3: 'A Feast for Crows', 4: 'A Dance of Dragons', 5: 'Darkly Dreaming Dexter
', 6: 'Dearly Devoted Dexter', 7: 'Dexter by Design', 8: 'Dexter in the Dark', 9: 'Dexter is Delicious', 10: 'Dexter is Dea
d'}
Which book do you want?2
Book successfully issued to your name
```

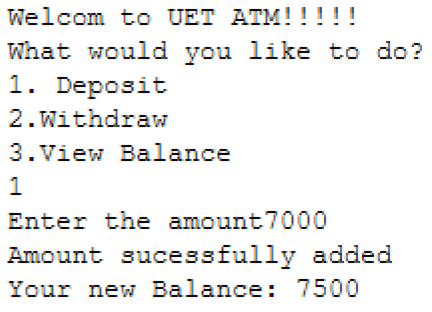
3) Using functions, create an ATM system

Code:



```
class Account:
    def __init__(self):
        self.balance = 500
    def addBalance(self,amount):
        self.balance+=amount
        print('Amount sucessfully added')
        print('Your new Balance: '+str(self.balance))

    def withdrawBalance(self,amount):
        self.balance-=amount
        print('Amount sucessfully withdrawed')
        print('Your new Balance: '+str(self.balance))

    def showBalance(self):
        print('Your Balance: '+str(self.balance))

Account = Account()
print('Welcom to UET ATM!!!!!\nWhat would you like to do?\n1. Deposit\n2.Withdraw\n3.View Balance')
choice = int(input())
if(choice!=3):
    amount = int(input('Enter the amount'))
    if(choice==1):
        Account.addBalance(amount)
    else:
        Account.withdrawBalance(amount)
else:
    Account.showBalance()
```

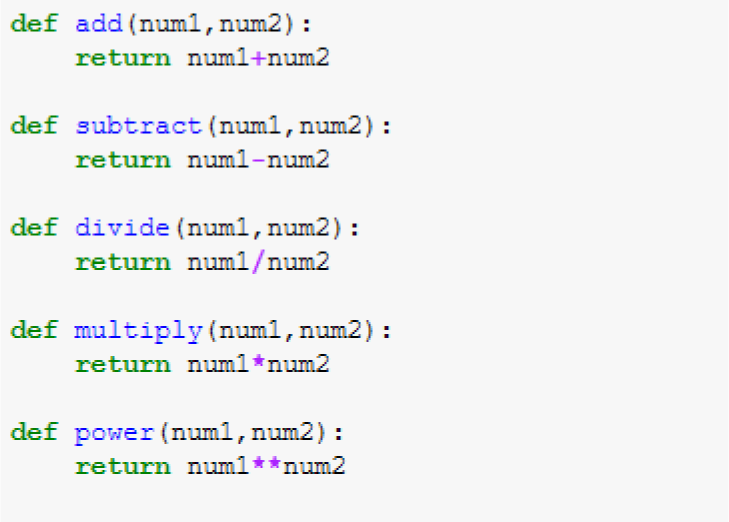
Output:



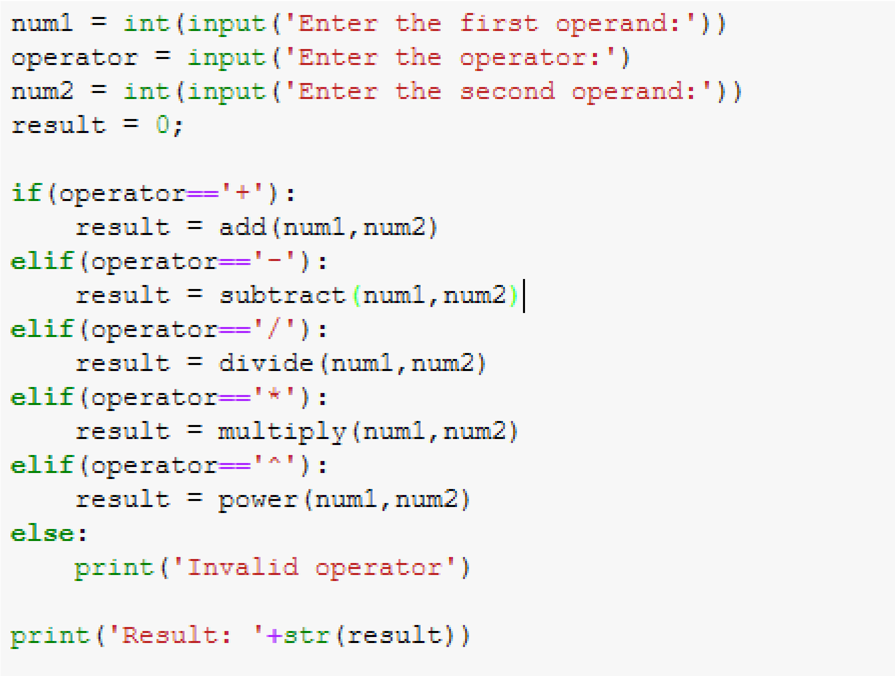
```
Welcom to UET ATM!!!!!
What would you like to do?
1. Deposit
2.Withdraw
3.View Balance
1
Enter the amount7000
Amount sucessfully added
Your new Balance: 7500
```

4) Design a calculator (+,-,*,/,pow) using functions

Code:

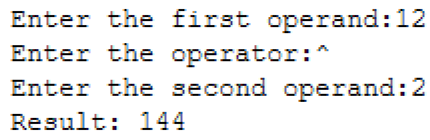


```python
def add(num1,num2):
    return num1+num2

def subtract(num1,num2):
    return num1-num2

def divide(num1,num2):
    return num1/num2

def multiply(num1,num2):
    return num1*num2

def power(num1,num2):
    return num1**num2
```



```python
num1 = int(input('Enter the first operand:'))
operator = input('Enter the operator:')
num2 = int(input('Enter the second operand:'))
result = 0;

if(operator=='+'):
    result = add(num1,num2)
elif(operator=='-'):
    result = subtract(num1,num2)
elif(operator=='/'):
    result = divide(num1,num2)
elif(operator=='*'):
    result = multiply(num1,num2)
elif(operator=='^'):
    result = power(num1,num2)
else:
    print('Invalid operator')

print('Result: '+str(result))
```

Output:



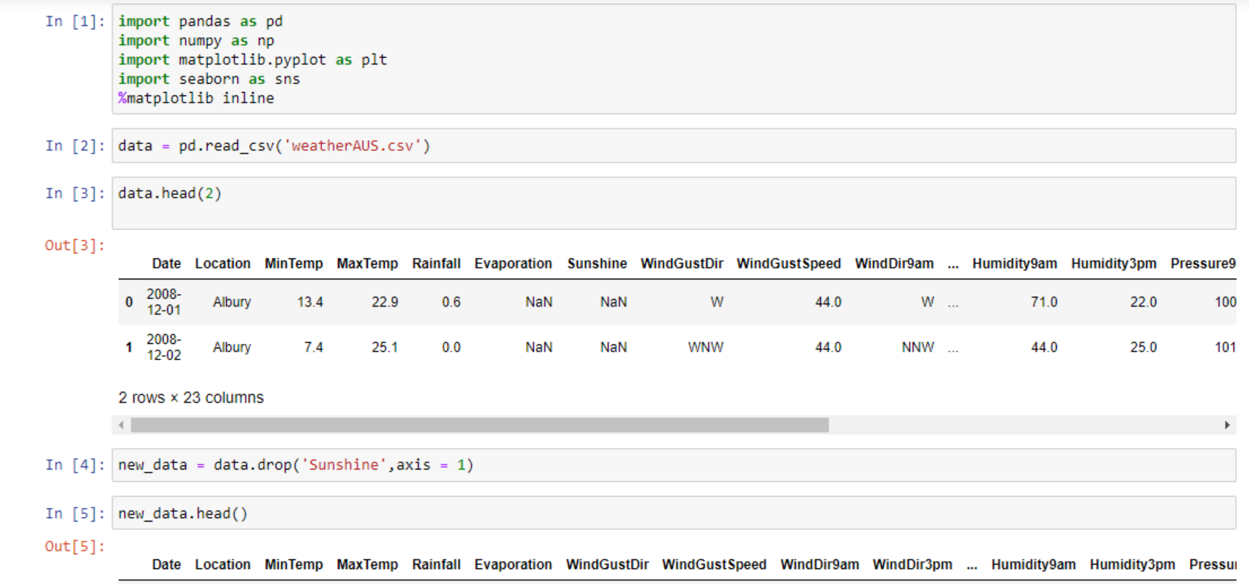
```
Enter the first operand:12
Enter the operator:^
Enter the second operand:2
Result: 144
```

Lab 7 (Pandas and NumPy Tasks):

Pandas:

1) In the 'WeatherAUS' dataset file, Perform using Pandas:

1.1) Drop the Sunshine column.



```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline

In [2]: data = pd.read_csv('weatherAUS.csv')

In [3]: data.head(2)

Out[3]:
```

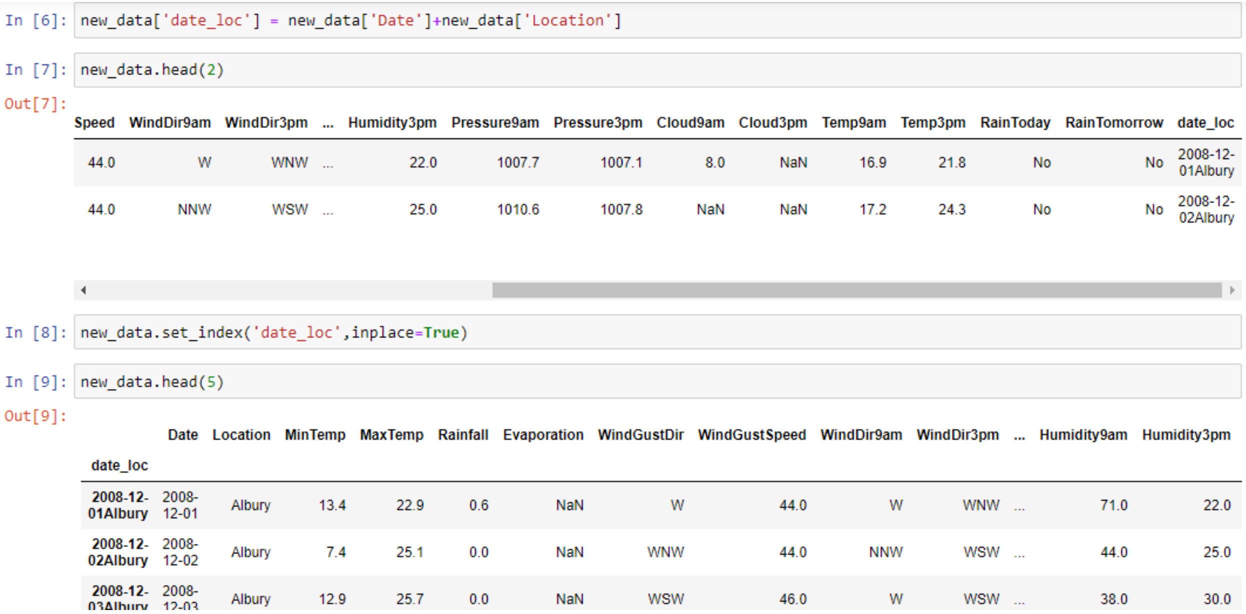
	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	100
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	101

2 rows × 23 columns

```
In [4]: new_data = data.drop('Sunshine',axis = 1)

In [5]: new_data.head()

Out[5]:
```

Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	Humidity9am	Humidity3pm	Pressui

1.2) Change the index of the data frame to (Date and Location) must be concatenated and then set the index.



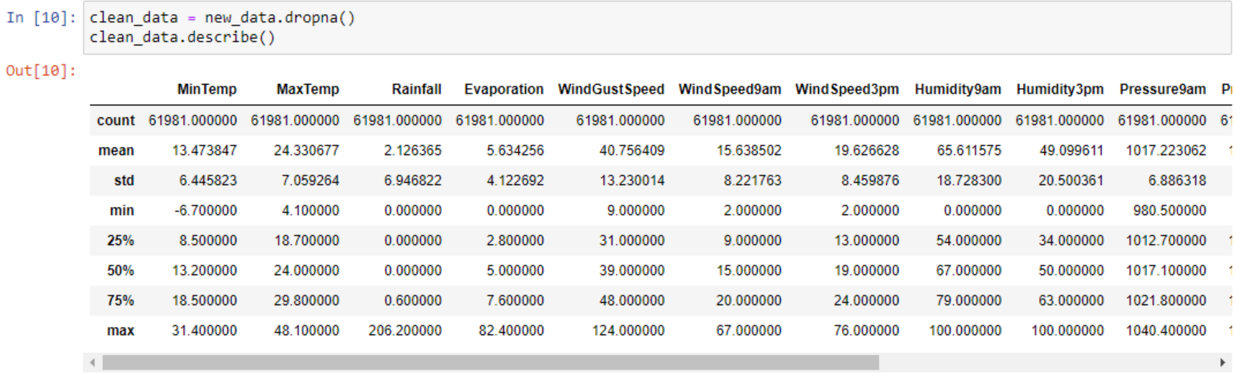
```
In [6]: new_data['date_loc'] = new_data['Date']+new_data['Location']

In [7]: new_data.head(2)

Out[7]:
```

Speed	WindDir9am	WindDir3pm	...	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow	date_loc
44.0	W	WNW	...	22.0	1007.7	1007.1	8.0	NaN	16.9	21.8	No	No	2008-12-01Albury
44.0	NNW	WSW	...	25.0	1010.6	1007.8	NaN	NaN	17.2	24.3	No	No	2008-12-02Albury

```
In [8]: new_data.set_index('date_loc',inplace=True)

In [9]: new_data.head(5)

Out[9]:
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	Humidity9am	Humidity3pm
date_loc													
2008-12-01Albury	2008-12-01	Albury	13.4	22.9	0.6	NaN	W	44.0	W	WNW	...	71.0	22.0
2008-12-02Albury	2008-12-02	Albury	7.4	25.1	0.0	NaN	WNW	44.0	NNW	WSW	...	44.0	25.0
2008-12-03Albury	2008-12-03	Albury	12.9	25.7	0.0	NaN	WSW	46.0	W	WSW	...	38.0	30.0

1.3) Drop rows having NA. Any problems with this step? Explain.



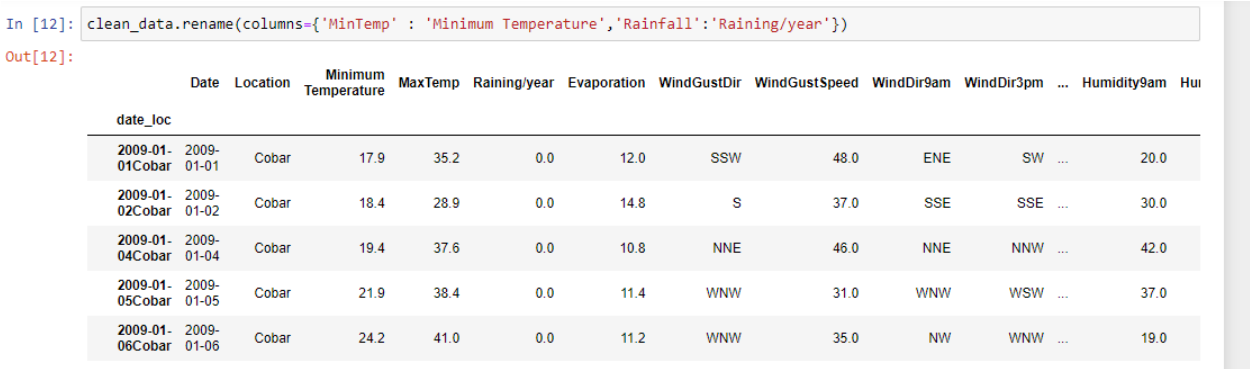
In [10]:
```
clean_data = new_data.dropna()
clean_data.describe()
```

Out[10]:

	MinTemp	MaxTemp	Rainfall	Evaporation	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	P
count	61981.000000	61981.000000	61981.000000	61981.000000	61981.000000	61981.000000	61981.000000	61981.000000	61981.000000	61981.000000	6
mean	13.473847	24.330677	2.126365	5.634256	40.756409	15.638502	19.626628	65.611575	49.099611	1017.223062	1
std	6.445823	7.059264	6.946822	4.122692	13.230014	8.221763	8.459876	18.728300	20.500361	6.886318	
min	-6.700000	4.100000	0.000000	0.000000	9.000000	2.000000	2.000000	0.000000	0.000000	980.500000	
25%	8.500000	18.700000	0.000000	2.800000	31.000000	9.000000	13.000000	54.000000	34.000000	1012.700000	1
50%	13.200000	24.000000	0.000000	5.000000	39.000000	15.000000	19.000000	67.000000	50.000000	1017.100000	1
75%	18.500000	29.800000	0.600000	7.600000	48.000000	20.000000	24.000000	79.000000	63.000000	1021.800000	1
max	31.400000	48.100000	206.200000	82.400000	124.000000	67.000000	76.000000	100.000000	100.000000	1040.400000	1

So by dropping all NaN values, the drop function removes all the rows which contain the NaN so it reduces the dataset as we can see it drops the 50% the data of our dataset.

1.4) Change column names to your choice!

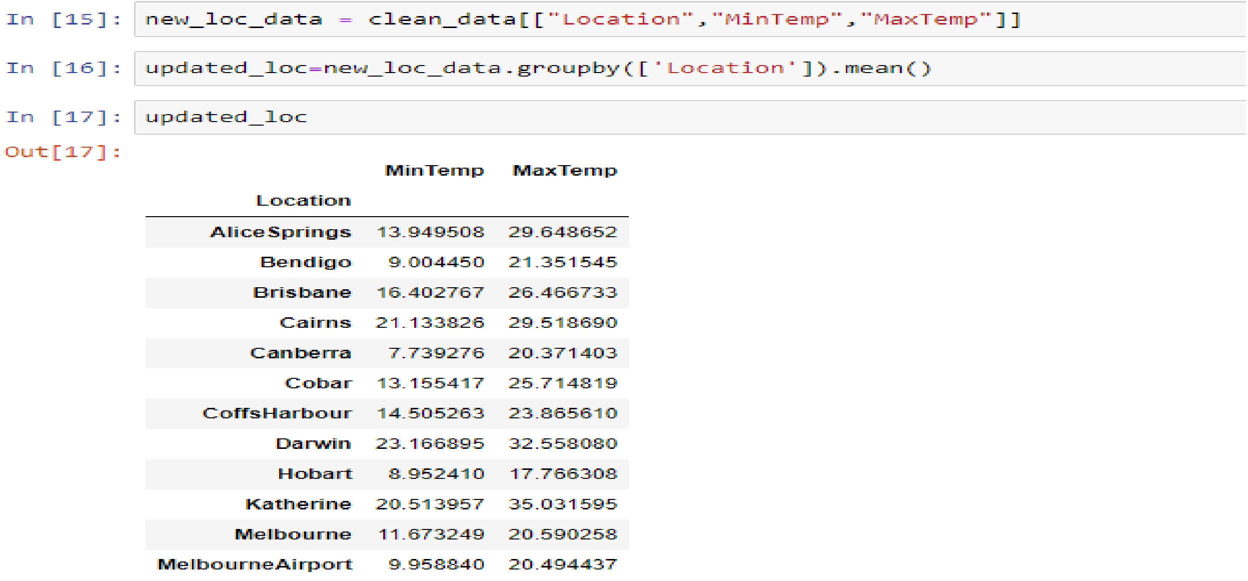


In [12]:
```
clean_data.rename(columns={'MinTemp' : 'Minimum Temperature','Rainfall':'Raining/year'})
```

Out[12]:

	Date	Location	Minimum Temperature	MaxTemp	Raining/year	Evaporation	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	Humidity9am	Hu
date_loc													
2009-01-01Cobar	2009-01-01	Cobar	17.9	35.2	0.0	12.0	SSW	48.0	ENE	SW	...	20.0	
2009-01-02Cobar	2009-01-02	Cobar	18.4	28.9	0.0	14.8	S	37.0	SSE	SSE	...	30.0	
2009-01-04Cobar	2009-01-04	Cobar	19.4	37.6	0.0	10.8	NNE	46.0	NNE	NNW	...	42.0	
2009-01-05Cobar	2009-01-05	Cobar	21.9	38.4	0.0	11.4	WNW	31.0	WNW	WSW	...	37.0	
2009-01-06Cobar	2009-01-06	Cobar	24.2	41.0	0.0	11.2	WNW	35.0	NW	WNW	...	19.0	

1.5) Create a data frame having location and Average Minimum and Average Maximum temperatures grouped by Location.



In [15]:
```
new_loc_data = clean_data[["Location","MinTemp","MaxTemp"]]
```

In [16]:
```
updated_loc=new_loc_data.groupby(['Location']).mean()
```

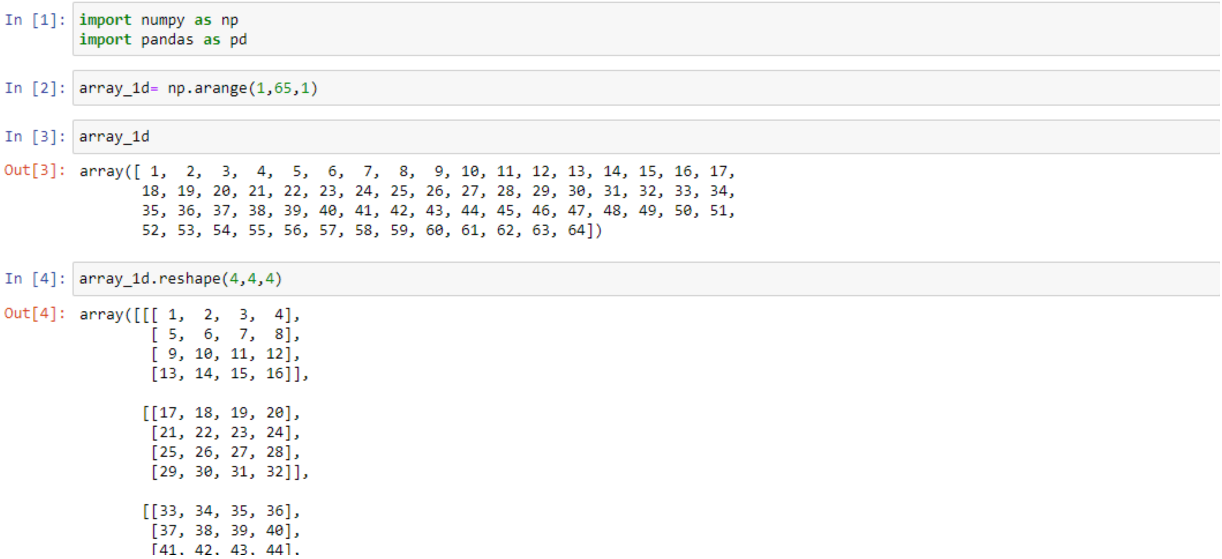
In [17]:
```
updated_loc
```

Out[17]:

	MinTemp	MaxTemp
Location		
AliceSprings	13.949508	29.648652
Bendigo	9.004450	21.351545
Brisbane	16.402767	26.466733
Cairns	21.133826	29.518690
Canberra	7.739276	20.371403
Cobar	13.155417	25.714819
CoffsHarbour	14.505263	23.865610
Darwin	23.166895	32.558080
Hobart	8.952410	17.766308
Katherine	20.513957	35.031595
Melbourne	11.673249	20.590258
MelbourneAirport	9.958840	20.494437

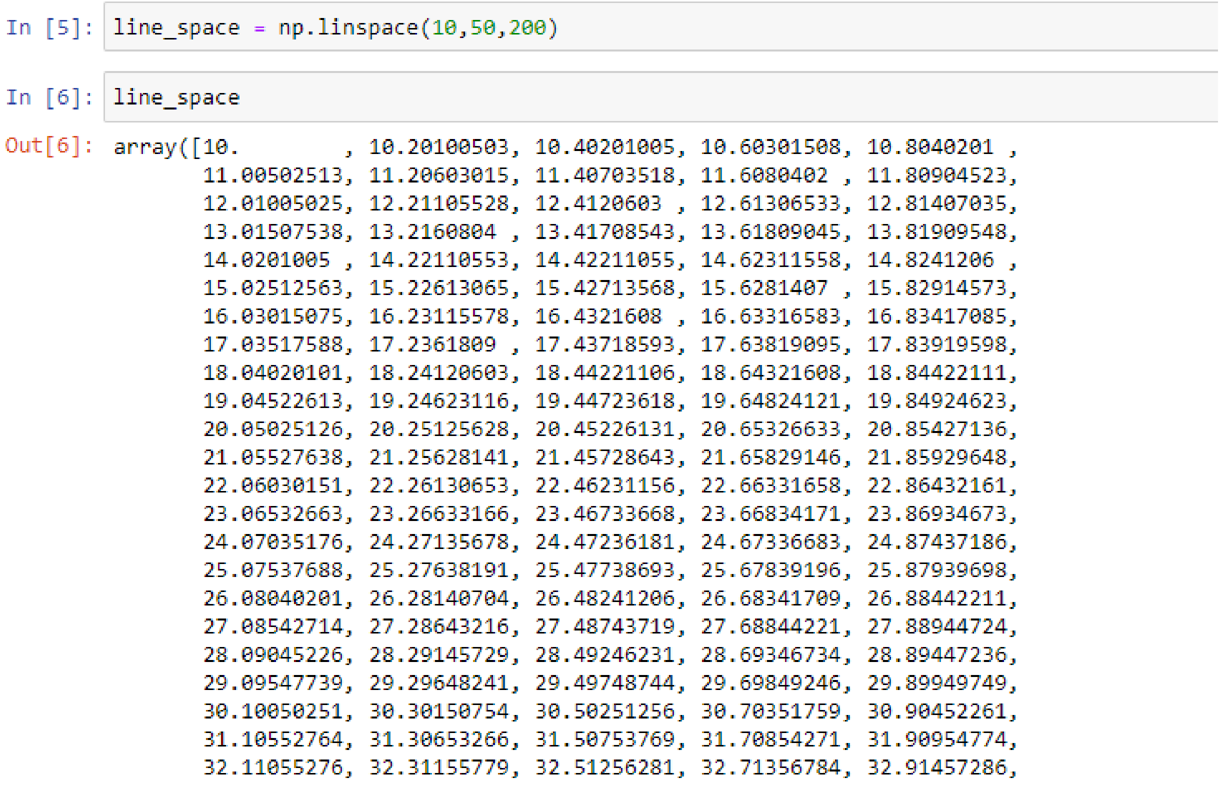
Numpy:

2) Create a simple 1D integer array (64 elements) in Numpy. Reshape the array to (4,4,4). Also change the data type to float.



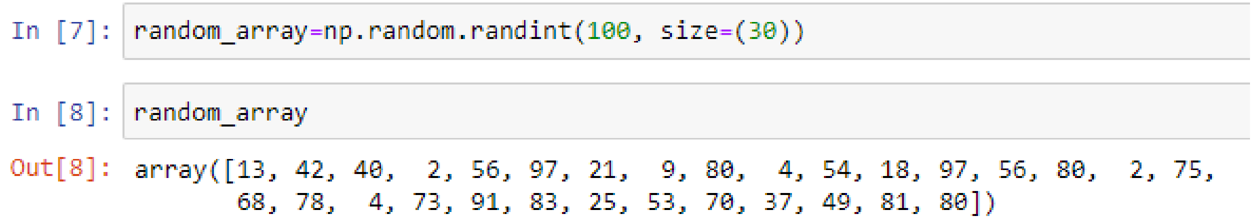
```
In [1]: import numpy as np
        import pandas as pd

In [2]: array_1d= np.arange(1,65,1)

In [3]: array_1d

Out[3]: array([ 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, 27, 28, 29, 30, 31, 32, 33, 34,
               35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
               52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64])

In [4]: array_1d.reshape(4,4,4)

Out[4]: array([[[ 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, 27, 28],
                [29, 30, 31, 32]],

               [[33, 34, 35, 36],
                [37, 38, 39, 40],
                [41, 42, 43, 44],
```

3) Perform linspace function as per your choice (other than practiced in lab)



```
In [5]: line_space = np.linspace(10,50,200)

In [6]: line_space

Out[6]: array([10.        , 10.20100503, 10.40201005, 10.60301508, 10.8040201 ,
               11.00502513, 11.20603015, 11.40703518, 11.6080402 , 11.80904523,
               12.01005025, 12.21105528, 12.4120603 , 12.61306533, 12.81407035,
               13.01507538, 13.2160804 , 13.41708543, 13.61809045, 13.81909548,
               14.0201005 , 14.22110553, 14.42211055, 14.62311558, 14.8241206 ,
               15.02512563, 15.22613065, 15.42713568, 15.6281407 , 15.82914573,
               16.03015075, 16.23115578, 16.4321608 , 16.63316583, 16.83417085,
               17.03517588, 17.2361809 , 17.43718593, 17.63819095, 17.83919598,
               18.04020101, 18.24120603, 18.44221106, 18.64321608, 18.84422111,
               19.04522613, 19.24623116, 19.44723618, 19.64824121, 19.84924623,
               20.05025126, 20.25125628, 20.45226131, 20.65326633, 20.85427136,
               21.05527638, 21.25628141, 21.45728643, 21.65829146, 21.85929648,
               22.06030151, 22.26130653, 22.46231156, 22.66331658, 22.86432161,
               23.06532663, 23.26633166, 23.46733668, 23.66834171, 23.86934673,
               24.07035176, 24.27135678, 24.47236181, 24.67336683, 24.87437186,
               25.07537688, 25.27638191, 25.47738693, 25.67839196, 25.87939698,
               26.08040201, 26.28140704, 26.48241206, 26.68341709, 26.88442211,
               27.08542714, 27.28643216, 27.48743719, 27.68844221, 27.88944724,
               28.09045226, 28.29145729, 28.49246231, 28.69346734, 28.89447236,
               29.09547739, 29.29648241, 29.49748744, 29.69849246, 29.89949749,
               30.10050251, 30.30150754, 30.50251256, 30.70351759, 30.90452261,
               31.10552764, 31.30653266, 31.50753769, 31.70854271, 31.90954774,
               32.11055276, 32.31155779, 32.51256281, 32.71356784, 32.91457286,
```

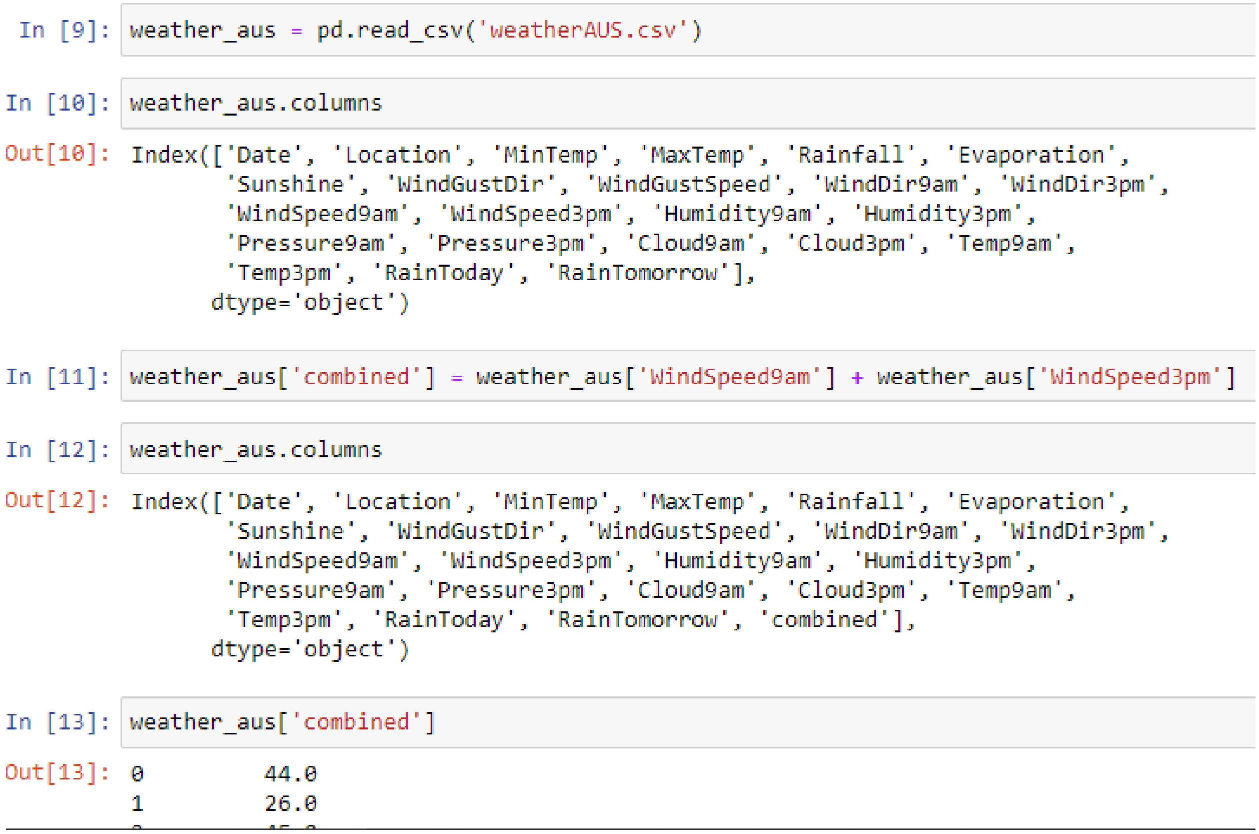
4) Create a random numbered array (random numbers ranging from 1 to 100) using Numpy.random() function.



```
In [7]:  random_array=np.random.randint(100, size=(30))

In [8]:  random_array

Out[8]:  array([13, 42, 40,  2, 56, 97, 21,  9, 80,  4, 54, 18, 97, 56, 80,  2, 75,
                68, 78,  4, 73, 91, 83, 25, 53, 70, 37, 49, 81, 80])
```

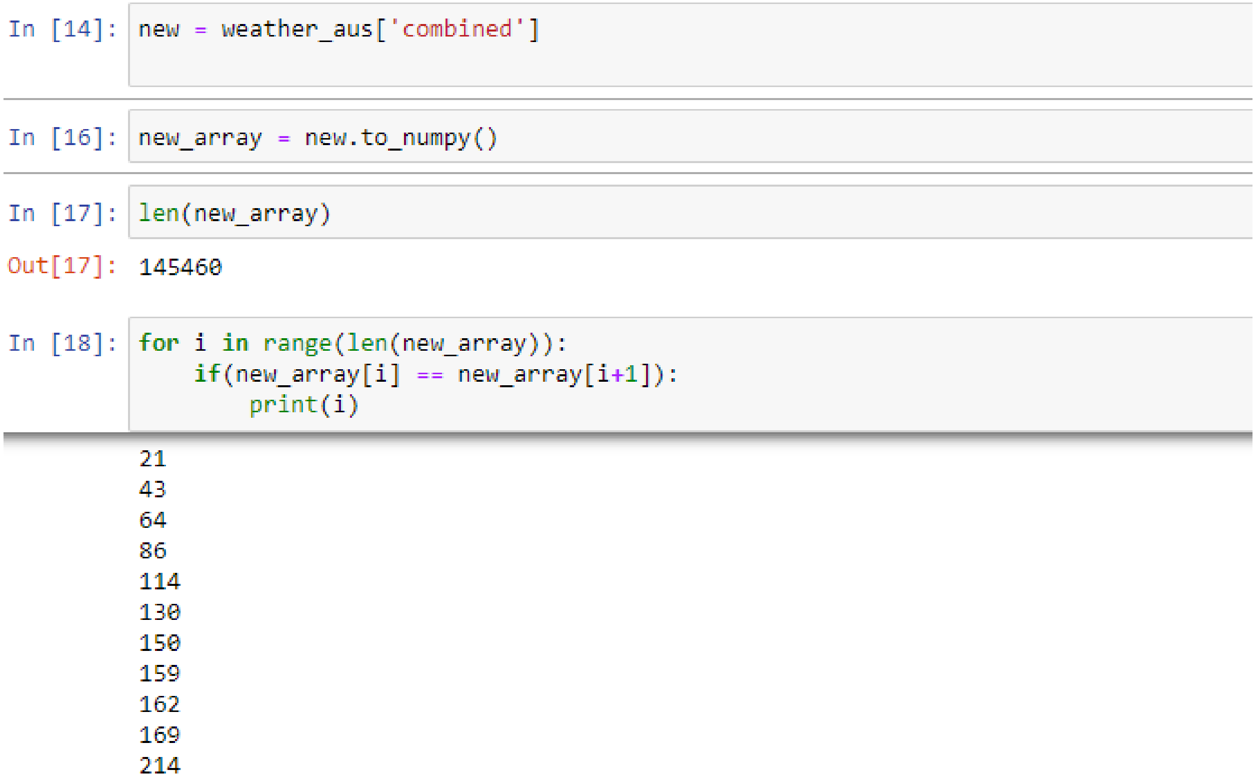
5) In the 'WeatherAUS' dataset file, Perform using Numpy:

5.1) Concatenate WindSpeed9am and WindSpeed3pm in a variable called "Combined".



```
In [9]:  weather_aus = pd.read_csv('weatherAUS.csv')

In [10]: weather_aus.columns

Out[10]: Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation',
                'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
                'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
                'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
                'Temp3pm', 'RainToday', 'RainTomorrow'],
               dtype='object')

In [11]: weather_aus['combined'] = weather_aus['WindSpeed9am'] + weather_aus['WindSpeed3pm']

In [12]: weather_aus.columns

Out[12]: Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation',
                'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
                'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
                'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
                'Temp3pm', 'RainToday', 'RainTomorrow', 'combined'],
               dtype='object')

In [13]: weather_aus['combined']

Out[13]: 0         44.0
         1         26.0
         2         45.0
```

5.2) Get positions of elements where the values of both features are same in "combined".



```
In [14]: new = weather_aus['combined']

In [16]: new_array = new.to_numpy()

In [17]: len(new_array)

Out[17]: 145460

In [18]: for i in range(len(new_array)):
             if(new_array[i] == new_array[i+1]):
                 print(i)

         21
         43
         64
         86
         114
         130
         150
         159
         162
         169
         214
```