Chapter 4 - exercise 1: Hãy thực hiện những yêu cầu liên quan tới series như sau:

1. Tạo nd_array_1 có 5 phần tử số nguyên,nd_array_2 có 5 phần tử số nguyên. Chuyển 2 array này thành 2 series là series1 và series2. Tính cộng, trừ, nhân, chia 2 series này. In các kết quả
2. So sánh xem các phần tử của series1 có >, <. = các phần tử của series2 không?
3. Cho 2 series, tạo series3 chỉ chứa các phần tử có trong series1 mà không có trong series2. in series3
4. Sử dụng lại 2 series là series1 và series2, tạo series6 chứa các phần tử chỉ có trong series1 và chỉ có trong series2
5. Tạo 1 series có 35 phần tử ngẫu nhiên có giá trị trong khoảng 1 đến 9. In shape, head và tail của series này. In series theo dạng array. Thống kê thông tin chung của series. Cho biết tổng của series. Cho biết phần tử có tần suất xuất hiện nhiều nhất
6. Sử dụng series của câu trên. In ra những dòng trong series mà giá trị chia hết cho 2 và cho 3
7. Từ series của câu 5 phía trên, in ra các giá trị unique (array)
8. Từ series của câu 5 phía trên, in các phần tử ở vị trí 0, 5, 10, 15
9. Từ series của câu 5 phía trên hãy tạo một series mới với mỗi phần tử có giá trị = lập phương của phần tử trong series. In head của serires mới
10. Cho list sau: lst = ["abc", "defg", "htlmj", "dfg", "ljsac"]. Tạo series từ lst này.Tạo series mới với mỗi phần tử có giá trị là chiều dài của phần tử trong series trên
11. Cho ser = pd.Series(np.array([1, 2, 4, 5, 8, 7, 6, 9])). In ser. In ra những dòng trong series mà giá trị là số nguyên tố
12. Cho mẫu email như sau pattern ='[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}'. Tạo một series mà mỗi phần tử là một chuỗi. In ra những dòng trong series thỏa điều kiện chuỗi là email

In [95]:
```
%config IPCompleter.greedy = True
import numpy as np
import pandas as pd
```

In [96]:
```
# Câu 1:
# Tạo nd_array_1 có 5 phần tử số nguyên,nd_array_2 có 5 phần tử số nguyên
# Chuyển 2 array này thành 2 series là series1 và series2
# Tính cộng, trừ, nhân, chia 2 series này
nd_array_1 = np.array([2, 4, 6, 8, 10])
nd_array_2 = np.array([1, 3, 5, 7, 11])
series1 = pd.Series(nd_array_1)
print(series1)
series2 = pd.Series(nd_array_2)
print(series2)
```

```
0     2
1     4
2     6
3     8
4    10
dtype: int32
0     1
1     3
2     5
3     7
4    11
dtype: int32
```

In [97]:
```
series1 + series2
```

Out[97]:
```
0     3
1     7
2    11
3    15
4    21
dtype: int32
```

In [98]:
```
series1 - series2
```

Out[98]:
```
0    1
1    1
2    1
3    1
4   -1
dtype: int32
```

In [99]:
```
series1 * series2
```

Out[99]:
```
0      2
1     12
2     30
3     56
4    110
dtype: int32
```

In [100]:
```
series1 / series2
```

Out[100]:
```
0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    0.909091
dtype: float64
```

In [101]:
```
# Câu 2: So sánh xem các phần tử của series1 có >, <. = các phần tử của series2 kh
series1 > series2
```

Out[101]:
```
0     True
1     True
2     True
3     True
4    False
dtype: bool
```

In [102]:
```
series1 < series2
```

Out[102]:
```
0    False
1    False
2    False
3    False
4     True
dtype: bool
```

In [103]:
```
series1 == series2
```

Out[103]:
```
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

In [104]:
```
# Câu 3: Cho 2 series, tạo series3 chỉ chứa các phần tử có trong series1 mà không
ser1 = pd.Series([1, 2, 3, 4, 5])
ser2 = pd.Series([4, 5, 6, 7, 8])
ser3 = ser1[~ser1.isin(ser2)]
ser3
```

Out[104]:
```
0    1
1    2
2    3
dtype: int64
```

In [105]:
```
# Câu 4: Sử dụng lại 2 series là series1 và series2, tạo series6 chứa các phần tử
ser4 = ser1[~ser1.isin(ser2)]
ser5 = ser2[~ser2.isin(ser1)]
ser6 = ser4.append(ser5)
ser6
```

Out[105]:
```
0    1
1    2
2    3
2    6
3    7
4    8
dtype: int64
```

In [106]:
```
# Câu 5: Tạo 1 series có 35 phần tử ngẫu nhiên có giá trị trong khoảng 1 đến 9. In
# In series theo dạng array
# Thống kê thông tin chung của series
# Cho biết tổng của series
# Cho biết phần tử có tần suất xuất hiện nhiều nhất
ser = pd.Series(np.random.randint(1, 10, 35))
print(ser.shape)
print(ser.head())
print(ser.tail())
print(ser.values)
```

```
(35,)
0    5
1    8
2    1
3    9
4    2
dtype: int32
30    3
31    4
32    1
33    3
34    7
dtype: int32
[5 8 1 9 2 4 5 2 2 2 6 7 2 4 6 2 3 5 1 8 2 5 5 9 4 5 5 5 2 5 3 4 1 3 7]
```

In [107]:
```
ser.describe()
```

Out[107]:
```
count    35.000000
mean      4.257143
std       2.279614
min       1.000000
25%       2.000000
50%       4.000000
75%       5.000000
max       9.000000
dtype: float64
```

In [108]:
```
ser.sum()
```

Out[108]:
```
149
```

In [109]:
```
ser.mode()
```

Out[109]:
```
0    5
dtype: int32
```

In [110]:
```
# Câu 6:  Sử dụng series của câu trên. In ra những dòng trong series mà giá trị ch
ser[(ser % 2 == 0) & (ser % 3 == 0)]
```

Out[110]:
```
10    6
14    6
dtype: int32
```

In [111]:
```
# Câu 7: Từ series của câu 5 phía trên, in ra các giá trị unique (array)
ser.unique()
```

Out[111]:
```
array([5, 8, 1, 9, 2, 4, 6, 7, 3], dtype=int64)
```

In [112]:
```
# Câu 8: Từ series của câu trên, in các phần tử ở vị trí 0, 5, 10, 15
pos = [0, 5, 10, 15]
ser[pos]
```

Out[112]:
```
0     5
5     4
10    6
15    2
dtype: int32
```

In [113]:
```
# Câu 9: Từ series của câu 5 trên hãy tạo một series mới với mỗi phần tử có giá tr
# In head của serires mới
import math
print(ser.head())
ser2 = ser.map(lambda x: math.pow(x,3))
print(ser2.head())
```

```
0    5
1    8
2    1
3    9
4    2
dtype: int32
0    125.0
1    512.0
2      1.0
3    729.0
4      8.0
dtype: float64
```

In [114]:
```
# Câu 10: Cho list sau: lst = ["abc", "defg", "htlmj", "dfg", "ljsac"]. Tạo series
# Tạo series mới với mỗi phần tử có giá trị là chiều dài của phần tử trong series
lst = ["abc", "defg", "htlmj", "dfg", "ljsac"]
ser3 = pd.Series(lst)
ser3
```

Out[114]:
```
0      abc
1     defg
2    htlmj
3      dfg
4    ljsac
dtype: object
```

In [115]:
```
ser4 = ser3.map(lambda x: len(x))
ser4
```

Out[115]:
```
0    3
1    4
2    5
3    3
4    5
dtype: int64
```

In [116]:
```
# Câu 11: Cho ser = pd.Series(np.array([1, 2, 4, 5, 8, 7, 6, 9])). In ser.
# In ra những dòng trong series mà giá trị là số nguyên tố
def test_prime(number):
    count = 0
    for i in range(1, number + 1):
        if number % i == 0:
            count += 1
    return count == 2
```

In [117]:
```
ser = pd.Series(np.array([1, 2, 4, 5, 8, 7, 6, 9]))
print(ser)
ser5 = ser.map(lambda x: test_prime(x))
print(ser5)
print(ser[ser5])
```

```
0    1
1    2
2    4
3    5
4    8
5    7
6    6
7    9
dtype: int32
0    False
1     True
2    False
3     True
4    False
5     True
6    False
7    False
dtype: bool
1    2
3    5
5    7
dtype: int32
```

In [118]:
```
# Câu 12: Cho mẫu email như sau pattern ='[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Z
# Tạo một series mà mỗi phần tử là một chuỗi
# In ra những dòng trong series thỏa điều kiện chuỗi là email
import re
pattern ='[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}'
emails = pd.Series(['reading newspaper from tuoitre.vn', 'tubirona@gmail.com', 'ng
is_email = emails.map(lambda x: bool(re.match(pattern, x)))
emails[is_email]
```

Out[118]:
```
1          tubirona@gmail.com
2         nguyen.nn@yahoo.com
3    tran_2014@hotmail.com.vn
dtype: object
```

In [119]:
```
# Câu 13:
'''Cho ser_names = pd.Series(['Manufacturer', 'Model', 'CarType', 'Min_Price', 'Pr
       'MPG_city', 'MPG_highway', 'AirBags', 'DriveTrain', 'Cylinders',
       'EngineSize', 'Horsepower', 'RPM', 'Rev_per_mile', 'Man_trans_avail',
       'Fuel_tank_capacity', 'Passengers', 'Length', 'Wheelbase', 'Width',
       'Turn_circle', 'Rear_seat_room', 'Luggage_room', 'Weight', 'Origin',
       'Make'])
In ra những dòng trong series trên thỏa điều kiện trong chuỗi có 'Price'
'''
ser_names = pd.Series(['Manufacturer', 'Model', 'CarType', 'Min_Price', 'Price',
       'MPG_city', 'MPG_highway', 'AirBags', 'DriveTrain', 'Cylinders',
       'EngineSize', 'Horsepower', 'RPM', 'Rev_per_mile', 'Man_trans_avail',
       'Fuel_tank_capacity', 'Passengers', 'Length', 'Wheelbase', 'Width',
       'Turn_circle', 'Rear_seat_room', 'Luggage_room', 'Weight', 'Origin',
       'Make'])
ser_names
```

Out[119]:
```
0           Manufacturer
1                  Model
2                CarType
3              Min_Price
4                  Price
5              Max_Price
6               MPG_city
7            MPG_highway
8                AirBags
9             DriveTrain
10             Cylinders
11            EngineSize
12            Horsepower
13                   RPM
14         Rev_per_mile
15       Man_trans_avail
16    Fuel_tank_capacity
17            Passengers
18                Length
19             Wheelbase
20                 Width
21           Turn_circle
22        Rear_seat_room
23          Luggage_room
24                Weight
25                Origin
26                  Make
dtype: object
```

In [120]:
```
has_Price = ser_names.map(lambda s: 'Price' in s)
has_Price.head()
```

Out[120]:
```
0    False
1    False
2    False
3     True
4     True
dtype: bool
```

In [121]:
```
names_Price = ser_names[has_Price]
names_Price
```

Out[121]:
```
3    Min_Price
4        Price
5    Max_Price
dtype: object
```

In [ ]:
```

```