📌Q1. 먼저 Pandas에서 지원하는 Series에 대해서 한번 다뤄보겠습니다.아래와 같이 인덱스와 데이터가 주어져있을때 Pandas의 Series 형태로 만들어보세요.
- 주어진 데이터에서 10이상 20이하의 데이터만 출력하는 Series를 재정의 해보세요.

핵심: Pandas 모듈의 **Series**를 활용해 **조건을 만족하는 데이터**를 출력하는 것이 목표입니다.

1. **Series:** 어떤 데이터 타입이든 보유할 수 있는 레이블링된 1차원 배열입니다. 특징은 사용자가 값과 함께 원하는 index를 입력할 수 있다는 것입니다. data 파라미터에는 입력하고자 하는 데이터를 넘겨주고, index 파라미터에는 원하는 index를 넘겨줍니다

series = pd.Series(data=data, index=idx)

```
HDD      19
SSD      11
USB       5
CLOUD    97
dtype: int64
```

2. **Series 조건문:** series 데이터 중에서 특정 조건에 해당하는 값만을 저장합니다.

series = series[series >= 10][series <= 20]

```
HDD    19
SSD    11
dtype: int64
```

소스코드:

```python
import pandas as pd
idx = ["HDD", "SSD", "USB", "CLOUD"]
data = [19, 11, 5, 97]

#series : 어떤 데이터 타입이든 보유할 수 있는 레이블링된 1차원 배열
series = pd.Series(data=data, index=idx)

# 10이상 20 이하 가지는 데이터
series = series[series >= 10][series <= 20]

print(series)
```

📌Q2. 두번째로 Pandas에서 지원하는 Dataframe을 다뤄보도록 하겠습니다. 다음과 같이 과일과 야채 각각 정리된 데이터가 있습니다. 이 두 데이터를 따로 보기엔 효율성이 떨어지니, 각 표에 정리된 데이터를 각각 하나의 데이터 프레임으로 생성한 후 다음 세부 구현을 진행해보세요.

- 두 데이터를 하나의 데이터로 결합해보세요.
- 결합한 데이터의 type을 이용해 데이터를 정렬해보세요.
- 최종적으로 과일과 야채 중 가장 비싼 가격의 합을 출력해보세요.

핵심: **서로 데이터 프레임을 하나의 데이터프레임으로 통합**한 후 **데이터를 정렬**합니다. 그리고 **각 type(과일/야채)에 해당하는 값들 중 가장 높은 두 개의 합**을 출력하는 것이 목표입니다.

1. **데이터프레임 통합하기:** pandas의 concat메소드를 활용해 두 개의 데이터 프레임을 통합한 새로운 데이터프레임을 정의합니다. 첫 번째 파라미터에 결합하고자 하는 두 개의 데이터프레임을 리스트 형식으로 넘겨줍니다. 두 번째 파라미터에는 결합하고자 하는 방향(axis)를 넘겨줍니다.

```
df3 = pd.concat([df1, df2], axis=0)
```

	Name	Type	Price
0	cherry	Fruit	100
1	mango	Fruit	110
2	poatato	Vegetable	60
3	onion	Vegetable	80
0	pepper	Vegetable	50
1	carrot	Vegetable	70
2	banana	Fruit	90
3	kiwi	Fruit	120

2. 데이터 프레임 정렬: 통합된 새로운 데이터프레임으로부터 loc(인덱스 이름)을 이용해 각 type에 해당하는 데이터를 조건문을 이용해 분류하고 새로운 데이터프레임에 저장합니다.

```
df_fruit = df3.loc[df3["Type"] == "Fruit"] # 과일
df_veg = df3.loc[df3["Type"] == "Vegetable"] # 야채
```

분류된 데이터들을 각각 내림차순 정렬합니다. by 파라미터는 정렬하고자 하는 대상 열이며 ascending은 정렬하고자 하는 방법(내림차순/오름차순)입니다.

```
df_fruit = df_fruit.sort_values(by="Price", ascending=False) # 과일
df_veg = df_veg.sort_values(by="Price", ascending=False) # 야채
```

	Name	Type	Price
3	kiwi	Fruit	120
1	mango	Fruit	110
0	cherry	Fruit	100
2	banana	Fruit	90

	Name	Type	Price
3	onion	Vegetable	80
1	carrot	Vegetable	70
2	poatato	Vegetable	60
0	pepper	Vegetable	50

각 데이터로부터 가장 비싼(내림차순한 데이터에서 0,1번째 인덱스) 두 개의 데이터를 추출 및 sum메소드를 이용해 더합니다.

sum(df_fruit[:2]["Price"]

sum(df_veg[:2]["Price"]

```
Sum of Top 2 Fruit Price :  230
Sum of Top 2 Vegetable Price :  150
```

소스코드:

```python
import pandas as pd

df1 = pd.DataFrame([
                    ["cherry", "Fruit", 100],
                    ["mango", "Fruit", 110],
                    ["poatato", "Vegetable", 60],
                    ["onion", "Vegetable", 80]],
                    columns=["Name", "Type","Price"])

df2 = pd.DataFrame([
                    ["pepper", "Vegetable", 50],
                    ["carrot", "Vegetable", 70],
                    ["banana", "Fruit", 90],
                    ["kiwi", "Fruit", 120]],
                    columns=["Name", "Type","Price"])

df3 = pd.concat([df1, df2], axis=0) # 두개의 데이터프레임을 axis 0 방향으로 결합한 새로운 데이터프레임 생성
print()
print(df3)

df_fruit = df3.loc[df3["Type"] == "Fruit"] # 새로운 데이터프레임으로부터 Type이 과일이 데이터만 가져온 과일 데이터프레임 생성
df_fruit = df_fruit.sort_values(by="Price", ascending=False) # 내림차순 정렬
print()
print(df_fruit)

df_veg = df3.loc[df3["Type"] == "Vegetable"] # 새로운 데이터프레임으로부터 Type이 야채이 데이터만 가져오기 야채 데이터프레임 생성
df_veg = df_veg.sort_values(by="Price", ascending=False) # 내림차순 정렬
print()
print(df_veg)

print()
print("Sum of Top 2 Fruit Price : ", sum(df_fruit[:2]["Price"])) # 과일 데이터 프레임으로부터 가장 비싼(0, 1번 인덱스) 두개의 데이터 추출 및 합산
print("Sum of Top 2 Vegetable Price : ", sum(df_veg[:2]["Price"])) # 야채 데이터 프레임으로부터 가장 비싼(0, 1번 인덱스) 두개의 데이터 추출 및 합산
```

📌Q3. 총 5명에서 게임을 진행했습니다. 총 5개 라운드를 진행했고, 각각 참여자당 5개의 점수를 받았습니다. 주어진 데이터를 dataframe의 형태로 만든 후 각 세부 구현을 진행해보세요.

- 참여자의 이름을 index로 해서 각 라운드의 columns를 추가해 데이터를 정리해보세요.
- 6번째 라운드의 점수가 아래와 같을 때, 이를 추가해보세요.
- 각 데이터의 mean, max, min 값을 출력해보세요.

핵심: 주어진 index/column/data를 이용해 새로운 데이터프레임을 생성합니다. 만든 데이터프레임에 **새로운 column(round6) 및 데이터를 추가**하고 각 column의 **mean(평균값), max(최대값) 그리고 min(최소값)**을 출력하는 것이 최종목표입니다.

1. **idx/column/data로 데이터프레임 생성하기:** pandas의 데이터프레임 생성 메소드를 이용해 새로운 데이트프레임을 생성합니다. 파라미터로 주어진 data, index 그리고 column을 넘겨줍니다.

```
df = pd.DataFrame(data, index=idx, columns=col)
```

	round_1	round_2	round_3	round_4	round_5
Sue	55	65	60	66	57
Ryan	64	77	71	79	67
Jay	88	81	79	89	77
Jane	45	35	30	46	47
Anna	91	96	90	97	99

2. 기존 데이터프레임에 column 추가하기: pandas의 series객체로 새로운 column(round_6)를 정의합니다. 이때, 넘겨주게 되는 인덱스 파라미터는 앞서 사용했던 index와 동일합니다. 기존의 데이터 프레임에 새롭게 정의한 column을 추가합니다.

```
col_round_6 = pd.Series([11, 15, 13, 17, 19], index=idx)
df["round_6"] = col_round_6
```

	round_1	round_2	round_3	round_4	round_5	round_6
Sue	55	65	60	66	57	11
Ryan	64	77	71	79	67	15
Jay	88	81	79	89	77	13
Jane	45	35	30	46	47	17
Anna	91	96	90	97	99	19

3. 각 열의 통계값을 출력하기: 각 column의 **mean(평균값), max(최대값) 그리고 min(최소값)**을 출력하기 위해서 pandas 데이터프레임의 describe메소드(Numeric type 데이터의 요약 정보를 보여줍니다)를 활용합니다. describe메소드의 통계자료중에서 mean, max 그리고 min에 대한 데이터만 출력하기 위해 loc메소드를 이용해 넘겨줍니다.

```
print(df.describe().loc[["mean", "max", "min"]])
```

```
      round_1  round_2  round_3  round_4  round_5  round_6
mean     68.6     70.8     66.0     75.4     69.4     15.0
max      91.0     96.0     90.0     97.0     99.0     19.0
min      45.0     35.0     30.0     46.0     47.0     11.0
PS C:\Users\75782\Desktop\오영주\새 폴더\4주차>
```

소스코드:

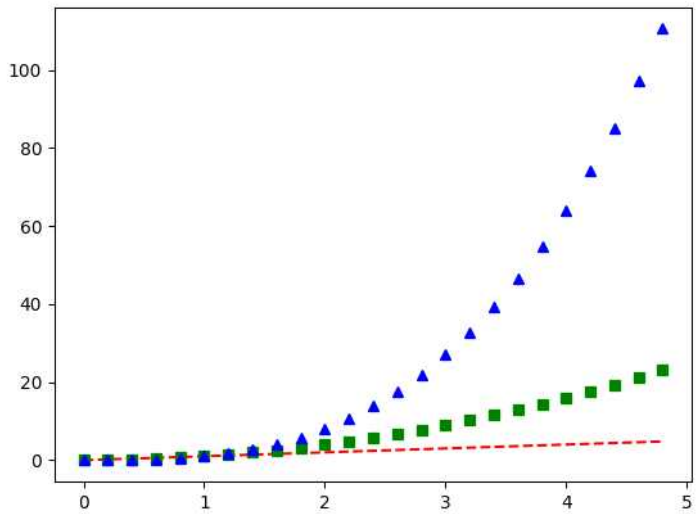
```python
import pandas as pd

idx = ["Sue", "Ryan", "Jay", "Jane", "Anna"]
col = ["round_1", "round_2", "round_3", "round_4", "round_5"]
data = [[55, 65, 60, 66, 57],
        [64, 77, 71, 79, 67],
        [88, 81, 79, 89, 77],
        [45, 35, 30, 46, 47],
        [91, 96, 90, 97, 99]]
print()
df = pd.DataFrame(data, index=idx, columns=col) # 인덱스와 열에 해당하는 리스트를 지정해주고 데이터를 포함한 새로운 데이터프레임을 생성합니다.
col_round_6 = pd.Series([11, 15, 13, 17, 19], index=idx) # 새로운 열 round_6를 Series함수를 이용해 생성합니다.
print(df)
print()
df["round_6"] = col_round_6 # 데이터프레임이 round_6열을 추가합니다.
print(df)
print()
# describe : Numeric type 데이터의 요약 정보를 보여줌
# describe : 다양한 통계량 요약해주는 메소드
#그 중 mean, max, min만
print(df.describe().loc[["mean", "max", "min"]]) # 각 열의 mean, max, min을 출력합니다.
```

📌Q4. 이번 미션부터는 Matplotlib을 이용해 간단히 그래프를 출력해보도록 하겠습니다. 다음과 같은 데이터 t가 주어져있을때, 출력 예시로 제안된 그래프를 출력해보세요.
- Plot 함수 내 각 마커의 색상과 모양을 표현하는 방법을 확인해보세요.

핵심: 주어진 데이터를 이용해 **세가지 색상/모양으로 표현한 그래프를 시각화**하는 것이 목표입니다.

1. **세가지 색상 및 모양으로 그래프를 시각화합니다:** matplotlib모듈의 pyplot을 활용해 원하는 데이터를 원하는 모양으로 시각화합니다. 파라미터 값을 순서대로 y, x, 마커색상, 마커의 스타일을 넘겨줍니다.

plt.plot(t, t, color="r", linestyle="--") **# f(t) = t (red dash)**
plt.plot(t, t**2, 'gs') **# f(t) = t^2 (blue square)**
plt.plot(t, t**3, 'b^') **# f(t) = t^3 (green triangle)**



시각화하기 위해 그래프를 flush합니다.

plt.show()

소스코드:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  # t데이터로 다음과 같은 3가지 plot을 그린다.
5  t = np.arange(0., 5., 0.2)
6
7
8  # t, t, 'r--' >> f(t) = t (red dash)
9  plt.plot(t, t, color="r", linestyle="--")
10 # t, t**2, 'gs' >>  f(t) = t^2 (blue square)
11 plt.plot(t, t**2, 'gs')
12 # t, t**3, 'b^' >>  f(t) = t^3 (green triangle)
13 plt.plot(t, t**3, 'b^')
14
15 #생성된 그래프를 flush한다.
16 plt.show()
```

참고: matplotlib.pyplot에서 마커의 색상 및 모양을 지정할 수 있는 다양한 값

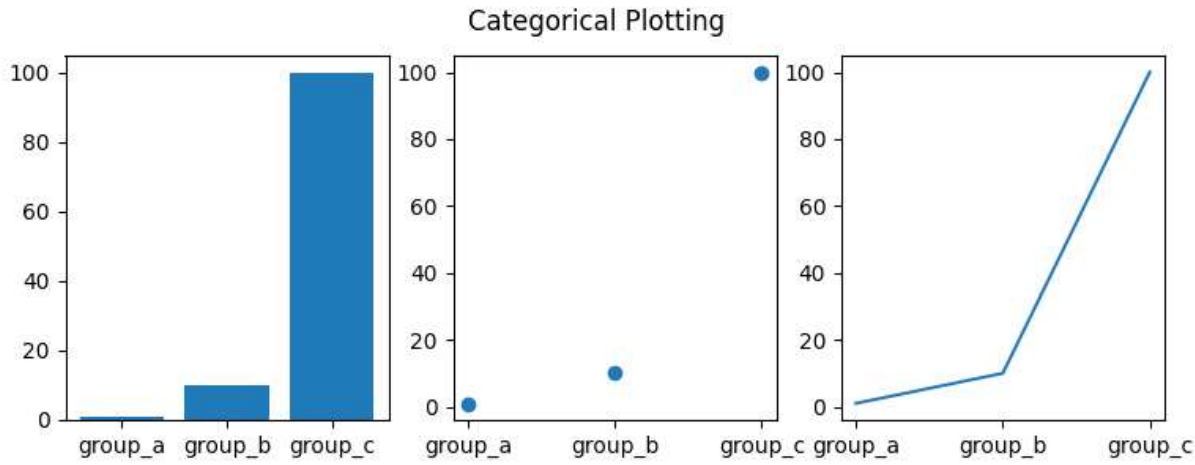
```
#marker
#        '.'    point marker
#        ','    pixel marker
#        'o'    circle marker
#        'v'    triangle_down marker
#        '^'    triangle_up marker
#        '<'    triangle_left marker
#        '>'    triangle_right marker
#        '1'    tri_down marker
#        '2'    tri_up marker
#        '3'    tri_left marker
#        '4'    tri_right marker
#        's'    square marker
#        'p'    pentagon marker
#        '*'    star marker
#        'h'    hexagon1 marker
#        'H'    hexagon2 marker
#        '+'    plus marker
#        'x'    x marker
#        'D'    diamond marker
#        'd'    thin_diamond marker
#        '|'    vline marker
#        '_'    hline marker
#line
#        '-'     solid line style
#        '--'     dashed line style
#        '-.'     dash-dot line style
#        ':'     dotted line style
#color
#        'b'   blue
#        'g'   green
#        'r'    red
#        'c'   cyan
#        'm'   magenta
#        'y'   yellow
#        'k'   black
#        'w'   white
#       hexa코드 혹은 color이름도 사용할 수 있음
```

📌Q5. Matplotlib을 이용해서 아래와 같은 데이터가 주어졌을때, 출력 예시와 같은 형태로 그래프를 출력해보세요.
- 각 그룹별 데이터를 다른 형식의 그래프로 생성해보세요. 각각 막대, 점, 선 그래프입니다.

핵심: 1행 3열 모양의 plot을 생성하고, 각 subplot에서 사용하는 그래프의 모양은 bar, scatter 그리고 line입니다.

1. **3개의 subplot을 시각화합니다:** 1행 3열 모양의 subplot을 생성하고 각 subplot에 해당하는 그래프를 설정해줍니다.

plt.figure(figsize=(9, 3)) **# 전체 figure의 크기를 설정합니다.**
plt.subplot(131) **# 1행3열 모양의 그래프에서 첫 번째 subplot을 의미합니다. 마지막 숫자로 원하는 subplot에 접근합니다.**
plt.bar(names, values) **# bar plot입니다.**
plt.scatter(names, values) **# scatter plot입니다**
plt.plot(names, values) **# plt.plot의 default인 line plot입니다.**



소스코드:

```python
import matplotlib.pyplot as plt

names = ['group_a', 'group_b', 'group_c'] #그래프 이름 리스트입니다.
values = [1, 10, 100] #그래프의 데이터값에 해당하는 리스트입니다.

plt.figure(figsize=(9, 3))

plt.subplot(131) # plot을 1행 3열 모양으로 나눈후, 첫번째 칸에 해당하는 subplot을 지정합니다.
plt.bar(names, values) # bar plot입니다.
plt.subplot(132) # 두번째 칸에 해당하는 subplot을 지정합니다.
plt.scatter(names, values) # scatter plot입니다.
plt.subplot(133) # 세번째 칸에 해당하는 subplot을 지정합니다.
plt.plot(names, values) # plt.plot의 default인 line plot입니다.

plt.suptitle('Categorical Plotting') # plot의 제목입니다.
plt.show()# plot을 출력합니다.
```