C4_S2_Practice

Task 1 :-

In [1]:

```
import pandas as pd
import numpy as np
```

In [9]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [10]:

```
pd.Series()
```

Out[10]:

```
Series([], dtype: float64)
```

In [11]:

```
std_id = np.arange(1001,1041)
std_id
```

Out[11]:

```
array([1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011,
       1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022,
       1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
       1034, 1035, 1036, 1037, 1038, 1039, 1040])
```

In [6]:

```
pd.Series(std_id)
```

Out[6]:

```
0     1001
1     1002
2     1003
3     1004
4     1005
5     1006
6     1007
7     1008
8     1009
9     1010
10    1011
11    1012
12    1013
13    1014
14    1015
15    1016
16    1017
17    1018
18    1019
19    1020
20    1021
21    1022
22    1023
23    1024
24    1025
25    1026
26    1027
27    1028
28    1029
29    1030
30    1031
31    1032
32    1033
33    1034
34    1035
35    1036
36    1037
37    1038
38    1039
dtype: int32
```

Task 2:-

In [12]:

```
Maths_Marks = np.random.randint(50,90,40)
Maths_Marks
```

Out[12]:

```
array([67, 64, 70, 82, 77, 52, 73, 81, 88, 50, 66, 79, 79, 81, 71, 74, 65,
       66, 89, 59, 70, 53, 71, 52, 68, 63, 67, 78, 74, 69, 55, 56, 55, 74,
       61, 65, 61, 82, 56, 83])
```

Task 3 :-

In [16]:

```
Physics_M = np.random.randint(40,95,40)
Physics_M
```

Out[16]:

```
array([84, 43, 48, 50, 73, 92, 78, 72, 79, 91, 73, 43, 51, 54, 87, 42, 57,
       60, 41, 41, 41, 62, 74, 88, 68, 90, 91, 84, 63, 85, 93, 68, 75, 88,
       77, 80, 66, 42, 87, 56])
```

Task 4 :-

In [18]:

```
Student = pd.Series(Maths_Marks,Physics_M, index = std_id)
Student
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_107712\2421868827.py in <module>
----> 1 Student = pd.Series(Maths_Marks,Physics_M, index = std_id)
      2 Student

TypeError: __init__() got multiple values for argument 'index'
```

In [20]:

```
total_Score = Maths_Marks + Physics_M
total_Score = pd.Series(total_Score, index = std_id)
total_Score
```

Out[20]:

```
1001    151
1002    107
1003    118
1004    132
1005    150
1006    144
1007    151
1008    153
1009    167
1010    141
1011    139
1012    122
1013    130
1014    135
1015    158
1016    116
1017    122
1018    126
1019    130
1020    100
1021    111
1022    115
1023    145
1024    140
1025    136
1026    153
1027    158
1028    162
1029    137
1030    154
1031    148
1032    124
1033    130
1034    162
1035    138
1036    145
1037    127
1038    124
1039    143
1040    139
dtype: int32
```

In [21]:

```
x = total_Score[(total_Score > 150)]
x
```

Out[21]:

```
1001    151
1007    151
1008    153
1009    167
1015    158
1026    153
1027    158
1028    162
1030    154
1034    162
dtype: int32
```

Task 5:-

In [24]:

```
M2_marks = np.random.randint(50,100,40)
P2_marks = np.random.randint(30,100,40)
Total_S = M2_marks + P2_marks
Total_S = pd.Series(Total_S, index = std_id)
Total_S
```

Out[24]:

```
1001    131
1002    100
1003    142
1004    102
1005    124
1006    111
1007    149
1008    137
1009    142
1010    133
1011    129
1012    106
1013    106
1014    183
1015    160
1016    113
1017    150
1018    166
1019    154
1020    183
1021    153
1022    139
1023    112
1024    103
1025    168
1026    184
1027    149
1028    133
1029    104
1030    135
1031    177
1032    184
1033    161
1034     93
1035    135
1036    132
1037    140
1038    143
1039    101
1040    140
dtype: int32
```

In [26]:

```
L1_and_L2 = Total_S + total_Score
L1_and_L2
```

Out[26]:

```
1001    282
1002    207
1003    260
1004    234
1005    274
1006    255
1007    300
1008    290
1009    309
1010    274
1011    268
1012    228
1013    236
1014    318
1015    318
1016    229
1017    272
1018    292
1019    284
1020    283
1021    264
1022    254
1023    257
1024    243
1025    304
1026    337
1027    307
1028    295
1029    241
1030    289
1031    325
1032    308
1033    291
1034    255
1035    273
1036    277
1037    267
1038    267
1039    244
1040    279
dtype: int32
```

Task 6 :-

In [29]:

```
Screen = L1_and_L2[(L1_and_L2 > 300)]
Screen
```

Out[29]:

```
1009    309
1014    318
1015    318
1025    304
1026    337
1027    307
1031    325
1032    308
dtype: int32
```

Task 7:

In [30]:

```
Total_Marks = 0.4*L1_and_L2 + 0.6*Screen
Total_Marks
```

Out[30]:

```
1001      NaN
1002      NaN
1003      NaN
1004      NaN
1005      NaN
1006      NaN
1007      NaN
1008      NaN
1009    309.0
1010      NaN
1011      NaN
1012      NaN
1013      NaN
1014    318.0
1015    318.0
1016      NaN
1017      NaN
1018      NaN
1019      NaN
1020      NaN
1021      NaN
1022      NaN
1023      NaN
1024      NaN
1025    304.0
1026    337.0
1027    307.0
1028      NaN
1029      NaN
1030      NaN
1031    325.0
1032    308.0
1033      NaN
1034      NaN
1035      NaN
1036      NaN
1037      NaN
1038      NaN
1039      NaN
1040      NaN
dtype: float64
```

Task 8 :-

In [31]:

```
Math_Olympaid = 0.75*Total_Marks
Math_Olympaid
```

Out[31]:

```
1001       NaN
1002       NaN
1003       NaN
1004       NaN
1005       NaN
1006       NaN
1007       NaN
1008       NaN
1009    231.75
1010       NaN
1011       NaN
1012       NaN
1013       NaN
1014    238.50
1015    238.50
1016       NaN
1017       NaN
1018       NaN
1019       NaN
1020       NaN
1021       NaN
1022       NaN
1023       NaN
1024       NaN
1025    228.00
1026    252.75
1027    230.25
1028       NaN
1029       NaN
1030       NaN
1031    243.75
1032    231.00
1033       NaN
1034       NaN
1035       NaN
1036       NaN
1037       NaN
1038       NaN
1039       NaN
1040       NaN
dtype: float64
```

Task 9:-

In [38]:

```
df = pd.DataFrame(Math_Olympaid, columns = ['Eligible'])
df
```

Out[38]:

	Eligible
1001	NaN
1002	NaN
1003	NaN
1004	NaN
1005	NaN
1006	NaN
1007	NaN
1008	NaN
1009	231.75
1010	NaN
1011	NaN
1012	NaN
1013	NaN
1014	238.50
1015	238.50
1016	NaN
1017	NaN
1018	NaN
1019	NaN
1020	NaN
1021	NaN
1022	NaN
1023	NaN
1024	NaN
1025	228.00
1026	252.75
1027	230.25
1028	NaN
1029	NaN
1030	NaN
1031	243.75
1032	231.00
1033	NaN
1034	NaN
1035	NaN
1036	NaN
1037	NaN
1038	NaN
1039	NaN
1040	NaN

In [ ]:

```
```