```
import pandas as pd
```

Q-1: Write a program to create an empty series.

```
# code here
pd.Series()
```

```
C:\Users\rajti\AppData\Local\Temp\ipykernel_19100\195155927.py:2: FutureWarning: The default dtype for empty Series will be 'object' ins
  pd.Series()
Series([], dtype: float64)
```

Q-2: Write a Pandas program to add, subtract, multiple and divide two Pandas Series.

```
# code here
a = pd.Series([2, 4, 6, 8, 10])
b = pd.Series([1, 3, 5, 7, 10])

print(a+b)
print(a-b)
print(a*b)
print(a/b)
```

```
0     3
1     7
2    11
3    15
4    20
dtype: int64
0    1
1    1
2    1
3    1
4    0
dtype: int64
0      2
1     12
2     30
3     56
4    100
dtype: int64
0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    1.000000
dtype: float64
```

Q-3 Write a Pandas program to compare the elements of the two Pandas Series.

Sample Series: [2, 4, 6, 8, 10], [1, 3, 5, 7, 10]

```
# code here
a = pd.Series([2, 4, 6, 8, 10])
b = pd.Series([1, 3, 5, 7, 10])

print(a==b)
print(a<b)
print(a>b)
```

```
0    False
1    False
2    False
3    False
4     True
dtype: bool
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

```
0     True
1     True
2     True
3     True
4    False
dtype: bool
```

Q-5. Write a function to change the data type of given a column or a Series. Function takes series and data type as input, returns the converted series.

```
series = pd.Series([1,2,'Python', 2.0, True, 100])
change to float type data
```

Note: Read about pd.to_numeric()

```
# code here
# Changing the dtype to numeric (int, float)
def change_dtype(series):
    return pd.to_numeric(series, errors = 'coerce')

series = pd.Series([1,2,'Python', 2.0, True, 100])
change_dtype(series)
```

```
0      1.0
1      2.0
2      NaN
3      2.0
4      1.0
5    100.0
dtype: float64
```

Download data - https://drive.google.com/file/d/1LRhXwbEodeWXtzPhJCX0X9Lf_BECzvqb/view?usp=share_link All Batsman runs series in IPL 2008 to 2022.

Below questions are based on this data.

```
data = pd.read_csv("batsman_runs_series.csv")
```

Q-6 Find top 10 most run getter from the series.

```
# code here
data.sort_values("batsman_run", ascending = False).head(10)
```

	batter	batsman_run
569	V Kohli	6634
462	S Dhawan	6244
130	DA Warner	5883
430	RG Sharma	5881
493	SK Raina	5536
27	AB de Villiers	5181
108	CH Gayle	4997
339	MS Dhoni	4978
452	RV Uthappa	4954
256	KD Karthik	4377

Q-7 No of players having runs above 3000

```
# code here
data[data["batsman_run"]>3000].shape[0]
```

```
20
```

Q-8 No of players having runs above mean value?

```
# code here
data[data["batsman_run"] > data["batsman_run"].mean()].shape[0]
```

```
128
```

Download data - https://drive.google.com/file/d/1QZuZ5bypUInfVvarHACLAi8tXXHvb8xd/view?usp=share_link

file name - items.csv

Q-9

i. Read `items.csv` making `item_name` as index.
ii. Show no of nan values
ii. Item price is given in $, so convert it to rupees without currency symbol.
iii. Make data type of newly made series as float.
iv. Fill nan with mean of the series

How csv file looks

```
item_name    item_price
Chips and Fresh Tomato Salsa    $2.39
Izze    $3.39
Nantucket Nectar    $3.39
Chips and Tomatillo-Green Chili Salsa    $2.39
Chicken Bowl    $16.98
```

```
# code here
#1
items = pd.read_csv("items.csv", index_col = ['item_name']).squeeze("columns")
items
```

```
item_name
Chips and Fresh Tomato Salsa              $2.39
Izze                                      $3.39
Nantucket Nectar                          $3.39
Chips and Tomatillo-Green Chili Salsa     $2.39
Chicken Bowl                             $16.98
                                          ...
Steak Burrito                            $11.75
Steak Burrito                            $11.75
Chicken Salad Bowl                       $11.25
Chicken Salad Bowl                        $8.75
Chicken Salad Bowl                        $8.75
Name: item_price, Length: 4622, dtype: object
```

```
#2
items.isna().sum()
```

```
50
```

```
#2ii
def rupees(x):
    try:
        y = x[1:]
    except:
        # Sometimes Dollar sign is not there which throws an exception
        y = x
    return float(y)*82.49
items.apply(rupees)
```

```
item_name
Chips and Fresh Tomato Salsa              197.1511
Izze                                      279.6411
```

```
Nantucket Nectar                          279.6411
Chips and Tomatillo-Green Chili Salsa     197.1511
Chicken Bowl                             1400.6802
                                            ...
Steak Burrito                             969.2575
Steak Burrito                             969.2575
Chicken Salad Bowl                        928.0125
Chicken Salad Bowl                        721.7875
Chicken Salad Bowl                        721.7875
Name: item_price, Length: 4622, dtype: float64
```

```
#3
items = items.apply(rupees)
```

```
#4
items.fillna(items.mean())
```

```
item_name
Chips and Fresh Tomato Salsa              197.1511
Izze                                      279.6411
Nantucket Nectar                          279.6411
Chips and Tomatillo-Green Chili Salsa     197.1511
Chicken Bowl                             1400.6802
                                            ...
Steak Burrito                             969.2575
Steak Burrito                             969.2575
Chicken Salad Bowl                        928.0125
Chicken Salad Bowl                        721.7875
Chicken Salad Bowl                        721.7875
Name: item_price, Length: 4622, dtype: float64
```

Q-10:

i. Find mean price

ii. Find 30th and 6th percentile value

iii. Plot Histogram on price with bin size 50

iv. No of items price lies between [1000 to 2000]

```
# code here
#1
items.mean()
```

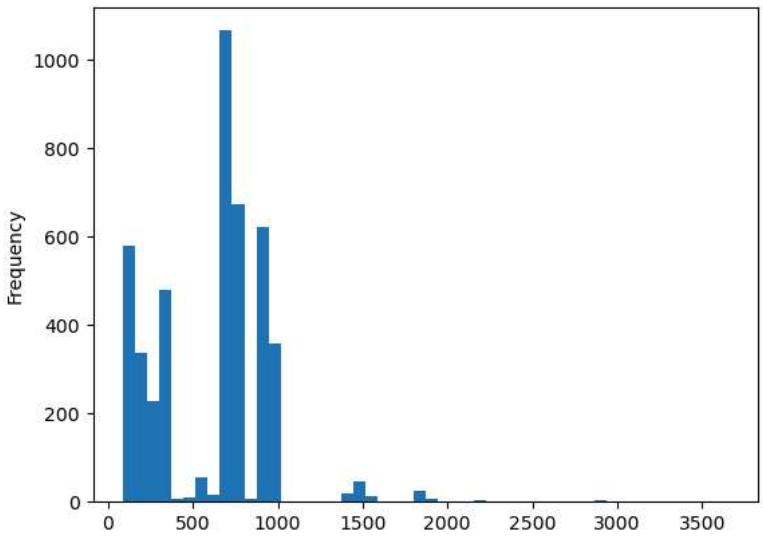
```
615.6254681102482
```

```
#2
items.quantile(q=0.3)
items.quantile(q=0.06)
```

```
103.1125
```

```
#3
items.plot.hist(bins=50)
```

```
<AxesSubplot:ylabel='Frequency'>
```




```
items[(items>1000) & (items<2000)]
```

```
item_name
Chicken Bowl           1400.6802
Chicken Salad Bowl     1856.0250
Steak Burrito          1483.1702
Chicken Burrito        1443.5750
Chicken Bowl           1443.5750
                          ...
Chicken Bowl           1856.0250
Steak Bowl             1938.5150
Chicken Bowl           1443.5750
Chips and Guacamole    1468.3220
Chicken Salad Bowl     1443.5750
Name: item_price, Length: 116, dtype: float64
```

Start coding or generate with AI.