Practice Exercise - Session 1

September 20, 2024

0.0.1 Interview Questions

[3]:
```python
#Import all the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Supress warnings
import warnings
warnings.filterwarnings("ignore")
```

0.0.2 I - Virat Kohli Dataset

[5]:
```python
df = pd.read_csv("virat.csv")
```

[6]:
```python
df.head()
```

[6]:
```
   Runs Mins  BF  4s  6s     SR  Pos Dismissal  Inns    Opposition  \
0    12   33  22   1   0  54.54    2       lbw     1  v Sri Lanka
1    37   82  67   6   0  55.22    2    caught     2  v Sri Lanka
2    25   40  38   4   0  65.78    1   run out     1  v Sri Lanka
3    54   87  66   7   0  81.81    1    bowled     1  v Sri Lanka
4    31   45  46   3   1  67.39    1       lbw     2  v Sri Lanka

           Ground Start Date
0        Dambulla  18-Aug-08
1        Dambulla  20-Aug-08
2  Colombo (RPS)  24-Aug-08
3  Colombo (RPS)  27-Aug-08
4  Colombo (RPS)  29-Aug-08
```

Spread in Runs Question 1: Analyse the spread of Runs scored by Virat in all his matches and report the difference between the scores at the 50th percentile and the 25th percentile respectively.

```
a)16.5
b)22.5
c)26.5
```

d)32.5

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 132 entries, 0 to 131
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Runs        132 non-null    object
 1   Mins        132 non-null    object
 2   BF          132 non-null    int64
 3   4s          132 non-null    int64
 4   6s          132 non-null    int64
 5   SR          132 non-null    object
 6   Pos         132 non-null    int64
 7   Dismissal   132 non-null    object
 8   Inns        132 non-null    int64
 9   Opposition  132 non-null    object
 10  Ground      132 non-null    object
 11  Start Date  132 non-null    object
dtypes: int64(5), object(7)
memory usage: 12.5+ KB
```

[9]:
```
df.describe()
```

[9]:
```
               BF          4s          6s         Pos        Inns
count  132.000000  132.000000  132.000000  132.000000  132.000000
mean    50.871212    4.371212    0.545455    3.303030    1.575758
std     38.729716    4.404032    1.086795    0.873174    0.496110
min      0.000000    0.000000    0.000000    1.000000    1.000000
25%     17.750000    1.000000    0.000000    3.000000    1.000000
50%     42.500000    3.000000    0.000000    3.000000    2.000000
75%     82.250000    7.000000    1.000000    4.000000    2.000000
max    140.000000   18.000000    7.000000    7.000000    2.000000
```

[10]:
```
df['Runs'] = df['Runs'].apply(lambda x: int(x[:-1]) if isinstance(x,str) and␣
 ↪x[-1] == '*'else int(x))
df['Runs'] = pd.to_numeric(df['Runs'], errors='coerce')
```

[11]:
```
df['Runs'].describe()
```

[11]:
```
count    132.000000
mean      46.848485
std       41.994635
min        0.000000
25%       10.000000
50%       32.500000
```

```
75%        80.250000
max       154.000000
Name: Runs, dtype: float64
```

[12]:
```
difference = 32.500000 - 10.000000
difference
```

[12]:
```
22.5
```

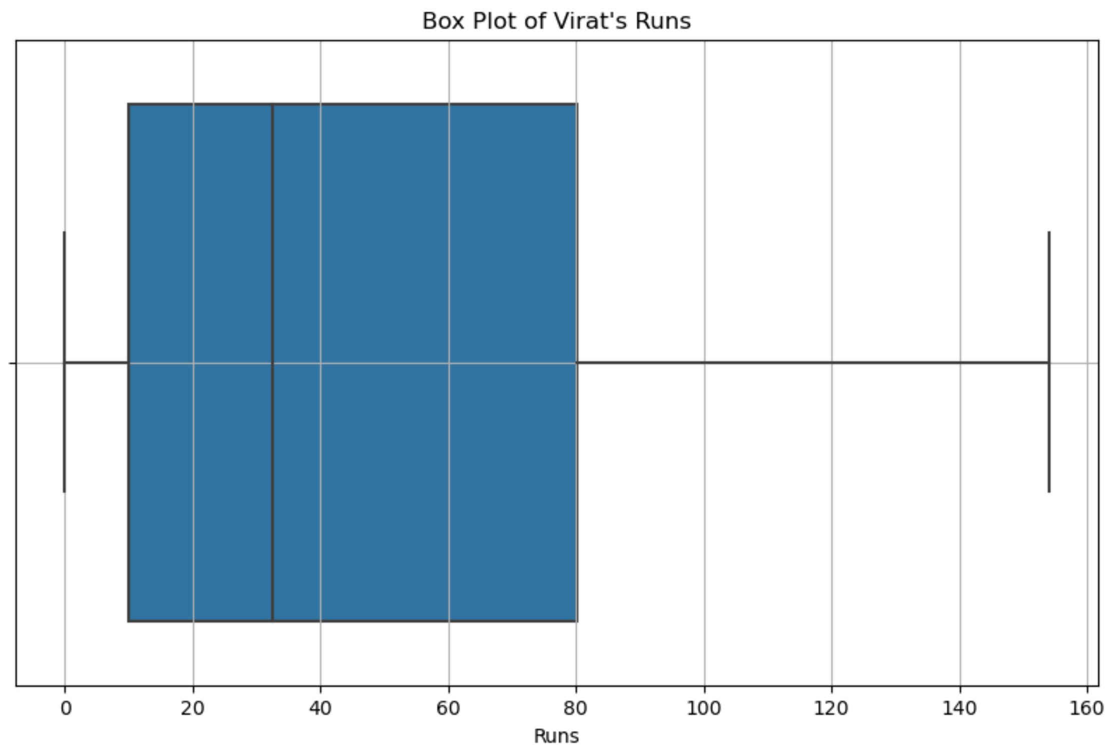
0.0.3 Hence answer for Q1 is option (b)

Box Plots Question 2: Plot a Box Plot to analyse the spread of Runs that Virat has scored. The upper fence in the box plot lies in which interval?

a)100-120
b)120-140
c)140-160
d)160-180

[15]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['Runs'])

plt.title("Box Plot of Virat's Runs")
plt.xlabel("Runs")
plt.grid(True)
plt.show()
```



[16]:
```python
# Calculate the upper fence
Q1 = df['Runs'].quantile(0.25)
Q3 = df['Runs'].quantile(0.75)
IQR = Q3 - Q1
upper_fence = Q3 + 1.5 * IQR
print("Q1 = ",Q1)
print('Q3 =', Q3)
print("IQR = ", IQR)
print("Upper Fence =",upper_fence)
```

```
Q1 =  10.0
Q3 = 80.25
IQR =  70.25
Upper Fence = 185.625
```

0.0.4 Q2.None of the options given is correct

False Statement Q3:Consider the following statements and choose the correct option

```
 I - Virat has played the maximum number of matches in 2011
 II - Virat has the highest run average in the year 2017
 III - Virat has the maximum score in a single match and the highest run average in the year 20
```

Which of the above statements is/are false?

```
a)I and II
b)I and III
c)II
d)III
```

[19]:
```python
# Lets us check for the statement I "Virat has played the maximum number of␣
↪matches in 2011"
df['Start Date'] = df['Start Date'].apply(lambda x: x[-2:])
```

[20]:
```python
df['Start Date'].value_counts()
# Statement I is correct
```

[20]:
```
11    31
13    23
14    17
10    16
12    11
15    10
16    10
09     6
08     5
17     3
```

```
Name: Start Date, dtype: int64
```

[21]:
```
# Lets us see "Virat has the highest run average in the year 2017"
pd.pivot_table(df, values = 'Runs', columns = ['Start Date'], aggfunc=np.mean)

# Statement II is incorrect
```

[21]:
```
Start Date    08         09      10    11         12         13         14  \
Runs        31.8  38.333333  45.375  42.0  40.363636  47.826087  58.529412

Start Date    15    16         17
Runs        30.4  73.9  61.666667
```

[22]:
```
#Virat has the maximum score in a single match and the highest run average in␣
↪the year 2016.
pd.pivot_table(df, values = 'Runs', columns = ['Start Date'], aggfunc=np.max)
# Statement III is correct
```

[22]:
```
Start Date  08   09   10   11   12   13   14   15   16   17
Runs        54  107  118  117  128  115  139  138  154  122
```

0.0.5 Hence answer for Q3 is option (c) or option (a) which is partially correct

Maximum Frequency Q4:Plot a histogram for the Mins column with 15 bins. Among the three ranges mentioned below, which one has the highest frequency?

A - [54.6,68)

B - [68,81.4)

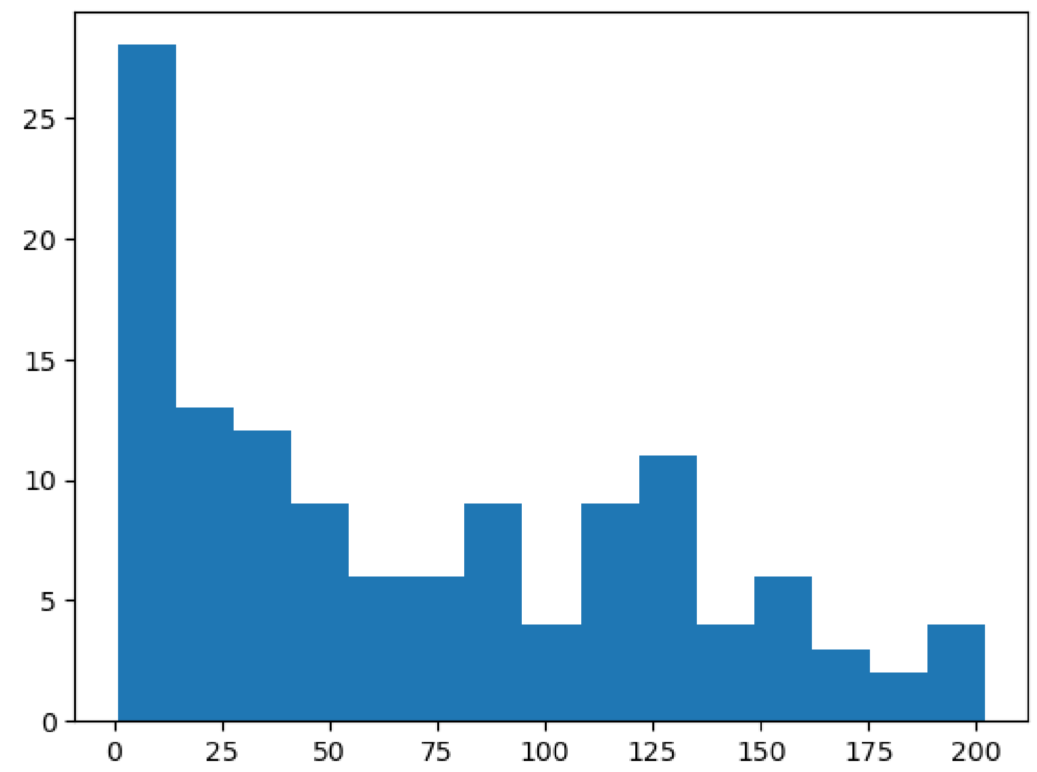
C - [121.6,135)

```
a)A - [54.6,68)
b)B - [68,81.4)
c)C - [121.6,135)
d)None of the bin ranges have the same frequency
```

[25]:
```
#Change the data type for Mins column

df2 = df[-(df['Mins'] == '-')]
df2['Mins']= df2['Mins'].apply(lambda x:int(x))
```

[26]:
```
plt.hist(df2.Mins, bins = 15)
plt.show()
```

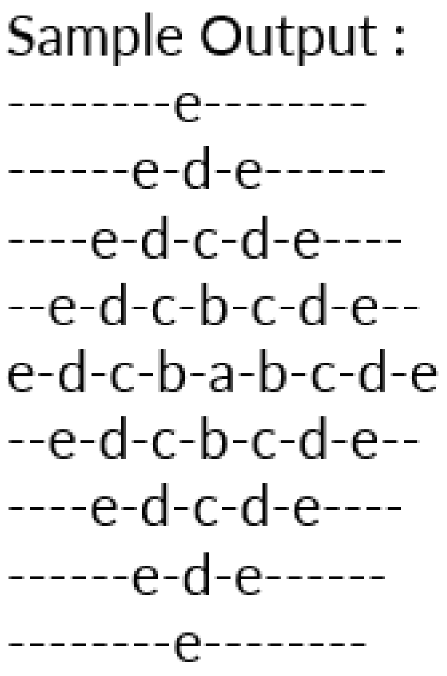


0.0.6 Option (d) is the correct answer

0.0.7 Coding Question :

1) Given a positive integer ‘n’ less than or equal to 26, you are required to print the below pattern

Sample Input: 5


Sample Output :

```
--------e--------
------e-d-e------
----e-d-c-d-e----
--e-d-c-b-c-d-e--
e-d-c-b-a-b-c-d-e
--e-d-c-b-c-d-e--
----e-d-c-d-e----
------e-d-e------
--------e--------
```

Sample Output : -e———

```
[56]: n = int(input())
alpha="abcdefghijklmnopqrstuvwxyz"
s=""
l=[]
for i in range(n):
    s="-".join(alpha[i:n])
    l.append(s[::-1]+s[1:])
length=len(l[0])
for i in range(n-1,0,-1):
    print(l[i].center(length,"-"))
for i in range(n):
    print(l[i].center(length,"-"))
```

```
 5

--------e--------
------e-d-e------
----e-d-c-d-e----
--e-d-c-b-c-d-e--
e-d-c-b-a-b-c-d-e
--e-d-c-b-c-d-e--
----e-d-c-d-e----
------e-d-e------
--------e--------
```

2) Given an integer, print whether it is Even or Odd.

Input: An integer
Output: ‘Even’ or ‘Odd’

Sample input: 3

Sample output: Odd

Sample input: 6

Sample output: Even

[63]:
```
num=int(input())
if num%2==0:
    print("Even")
else:
    print("Odd")
```

```
 4
```

```
Even
```

3) You’re trying to automate your alarm clock by writing a function for it. You’re given a day of the week encoded as 1=Mon, 2=Tue, … 6=Sat, 7=Sun, and whether you are on vacation as a boolean value (a boolean object is either True or False. Google “booleans python” to get a better understanding). Based on the day and whether you’re on vacation, write a function that returns a time in form of a string indicating when the alarm clock should ring.

When not on a vacation, on weekdays, the alarm should ring at “7:00” and on the weekends (Saturday and Sunday) it should ring at “10:00”.

While on a vacation, it should ring at “10:00” on weekdays. On vacation, it should not ring on weekends, that is, it should return “off”.

Input: The input will be a list of two elements. The first element will be an integer from 1 to 7, and the second element will be a boolean value.
Output: The output will be a string denoting the time alarm will ring or ‘off’

Sample input: [7, True]

Sample output: off

[71]:
```
def alarm_time(day_of_the_week, is_on_vacation):
    weekend = {6, 7}
    if is_on_vacation:
        if day_of_the_week in weekend:
            return 'Off'
        else:
            return '10:00'
    else:
        if day_of_the_week in weekend:
            return '10:00'
        else:
            return '07:00'
input_str = input("Enter a list of two elements [day_of_the_week,␣
 ↪is_on_vacation]: ")
input_list = eval(input_str)
if len(input_list) != 2 or not isinstance(input_list[0], int) or not␣
 ↪isinstance(input_list[1], bool):
    print("Invalid input. Please provide a list with an integer from 1 to 7 and␣
 ↪a boolean value.")
else:
    print(alarm_time(input_list[0], input_list[1]))
```

```
Enter a list of two elements [day_of_the_week, is_on_vacation]:  6,True

Off
```

4) Any number, say n is called an Armstrong number if it is equal to the sum of its digits, where each is raised to the power of number of digits in n. For example: 153=13+53+33

Write Python code to determine whether an entered three digit number is an Armstrong number or not. Assume that the number entered will strictly be a three digit number. Print "True" if it is an Armstrong number and print "False" if it is not. Sample Input: 153 Sample Output: True

[78]:
```
n=int(input())
sum = 0
temp = n
while temp > 0:
   digit = temp % 10
   sum += digit ** 3
   temp //= 10
if n == sum:
   print("True")
else:
   print("False")
```

```
 5

False
```

5) A pascal's triangle is a very interesting mathematical concept. Each number here is a sum of the two numbers directly above it. Following is an 8 level Pascal's triangle:

You can read about Pascal's triangle here. Your task is to print an nth level of Pascal's triangle. The input will contain an integer n. The output will contain 1 line of the list of numbers representing the nth row of Pascal's triangle.

Sample Input: 6 Sample Output:

[1, 5, 10, 10, 5, 1]

[81]:
```
n=int(input())
def generate_pascals_triangle_row(n):
    row = [1]
    for k in range(1, n):
        row.append(row[-1] * (n - k) // k)
    return row
print(generate_pascals_triangle_row(n))
```

```
 5
```

```
[1, 4, 6, 4, 1]
```

6) Given two strings, one of the strings will contain an extra character. Find the extra character. The number of all the other characters in both the strings will be the same. Check the sample input/output for more clarification.

The code will be case sensitive.

Input: Two strings on two separate lines.
Output: One Character which is extra in one of the strings

Sample input: abcd cedab

Sample output: e

[86]:
```
string1 = input().strip()
string2 = input().strip()


#write code to find the extra character here
from collections import Counter

# Count occurrences of each character in both strings
count1 = Counter(string1)
count2 = Counter(string2)

# Find the extra character
extra_char = (count2 - count1) or (count1 - count2)

# Print the extra character
if extra_char:
    print(extra_char.popitem()[0])
else:
    print("No extra character found.")
```

```
 abcd
 cedab
```

```
e
```

6) While extracting data from different sources, often numeric values come in string format and with commas like 1,000 or 23,321 and also sometimes with spaces in start and beginning of the string. For simplicity, we will consider only integer values imbedded with commas. You will take the input and print the cleaned integer without commas and spaces.

Input: One line input of string, it will consist of only spaces commas and digits
Output: Cleaned number

Sample input: 3,213

Sample output: 3213

[89]:
```python
value=input().strip()
cleaned_number = value.replace(',', '').replace(' ', '')
print(cleaned_number)
```

```
 3,213
3213
```

7) Write a program that computes the value of n+nn+nnn+nnnn+… nn…n ntimes with a given number as the value of n.

For example, if n=3 , then you have to find the value of 3+33+333 if n=10, then you have to find the value of 10 + 1010 + 101010 + 10101010 + 1010101010 + 101010101010 + 10101010101010 + 1010101010101010 +101010101010101010+ 10101010101010101010

Note: n will always be a positive number

[94]:
```python
n=int(input())
current_term = str(n)
total_sum = 0
for i in range(1, n + 1):
    # Add the current term to the total sum
    total_sum += int(current_term)
    current_term += str(n)
print(total_sum)
```

```
 3
369
```

[ ]:
```python

```