Python - 5 - Data Analysis

September 30, 2020

1 Python - 5

[110]:
```
# import libraries
import pandas as pd  # standard shortcut names
import numpy as np
```

[1]:
```
#Part 1

#Create a Pandas series that includes three states and their abreviations.
#The output should look like this (states can be changed):
#Washington    WA
#Oregon        OR
#California    CA
```

[112]:
```
state_series = pd.Series({'Washington': 'WA' , 'Oregon': 'OR', 'New York':␣
 ↪'NY'})
print("Series of 3 US states is:\n",state_series)
```

```
Series of 3 US states is:
 Washington    WA
Oregon        OR
New York      NY
dtype: object
```

[113]:
```
#Part 2

#Create Pandas series that includes all even numbers in the rage of 1 to 50.
#Use the range function instead of listing all the numbers individually
```

[114]:
```
even_series = pd.Series([i for i in range(1,51) if i%2 == 0])
print("Even number series between 1 and 50:\n", even_series)
```

```
Even number series between 1 and 50:
 0      2
1      4
2      6
3      8
```

```
4     10
5     12
6     14
7     16
8     18
9     20
10    22
11    24
12    26
13    28
14    30
15    32
16    34
17    36
18    38
19    40
20    42
21    44
22    46
23    48
24    50
dtype: int64
```

[115]:
```
#Part 3
#Using the map function multiply each number in Part 2 by 2
```

[116]:
```
double_even_series = pd.Series(map(lambda x:x*2, even_series))
print("Series with even numbers multiplied by 2:")
print(double_even_series)
```

```
Series with even numbers multiplied by 2:
0       4
1       8
2      12
3      16
4      20
5      24
6      28
7      32
8      36
9      40
10     44
11     48
12     52
13     56
14     60
15     64
16     68
```

```
17     72
18     76
19     80
20     84
21     88
22     92
23     96
24    100
dtype: int64
```

[118]:
```
#Part 4
#From the series in Part 2 select all the numbers that are between 10 and 20
```

[120]:
```
#Including even numbers between 10 and 20
finalSeries = even_series[(even_series > 9) & (even_series < 21)]
print("Even numbers between 10 and 20 from above series:")
print(finalSeries)
```

```
Even numbers between 10 and 20 from above series:
4    10
5    12
6    14
7    16
8    18
9    20
dtype: int64
```

[ ]:
```
#Part 5
#Print the top 2 elements in the series in Part 4
```

[121]:
```
#Using Head Function
print("Top 2 elements:")
finalSeries.head(2)
```

```
Top 2 elements:
```

[121]:
```
4    10
5    12
dtype: int64
```

[122]:
```
#Part 6
#Print the bottom 2 elements in then series in Part 4
```

[123]:
```
#Using Tail Function
print("Bottom 2 elements:")
finalSeries.tail(2)
```

```
Bottom 2 elements:
```

[123]:
```
8     18
9     20
dtype: int64
```

[124]:
```
#Part 7
#For the following data points in x, show the descriptive statistics
#(i.e. count, mean, std, min, 25%, 50%, 75%, and max)

x = pd.Series([8.10, 8.97, 9.88, 11.58, 11.00, 7.41, 6.15, 9.77, 9.17, 10.04, 5.
 ↪70, 10.97, 14.62, 8.56, 12.05, 11.33, 8.92, 12.74, 13.86, 11.78])
```

[125]:
```
x.describe()
#Using Describe function to get the descriptive statistics
#Count = 20.00
#mean = 10.13
#std = 2.3584
#min = 5.7
#25% = 8.83
#50% = 9.96
#75% = 11.63
#max = 14.62
```

[125]:
```
count    20.000000
mean     10.130000
std       2.358412
min       5.700000
25%       8.830000
50%       9.960000
75%      11.630000
max      14.620000
dtype: float64
```

[126]:
```
#Part 8
#Return the numbers in x that are in even positions
```

[127]:
```
print("Numbers in Series x at even positions:")
x[[even for even in range(0,20) if even%2 == 0]]
```

```
Numbers in Series x at even positions:
```

[127]:
```
0      8.10
2      9.88
4     11.00
6      6.15
8      9.17
10     5.70
12    14.62
14    12.05
16     8.92
18    13.86
```

```
dtype: float64
```

[128]:
```
#Part 9
#Return the numbers in x that are greater than the mean
```

[129]:
```
x_mean = x.mean()
print("Mean of numbers in series x:", x_mean)
print("\nNumbers in series x greater than mean:")
x1 = x > x_mean
x[x1]
```

```
Mean of numbers in series x: 10.129999999999999

Numbers in series x greater than mean:
```

[129]:
```
3     11.58
4     11.00
11    10.97
12    14.62
14    12.05
15    11.33
17    12.74
18    13.86
19    11.78
dtype: float64
```

[78]:
```
#Part 10
#Return (if any) the numbers in x that are greater than the mean plus one␣
↪standard deviation
```

[132]:
```
x_mean = x.mean() #Mean of series x
x_std = x.std() #Standard deviation of series x
new_x = x_mean + x_std #Sum of mean and one standard deviation
print("Mean plus one standard deviation:",new_x)
print("\nNumbers in series greater than mean and one standard deviation:")
x2 = x > new_x
x[x2]
```

```
Mean plus one standard deviation: 12.488411597486301

Numbers in series greater than mean and one standard deviation:
```

[132]:
```
12    14.62
17    12.74
18    13.86
dtype: float64
```