PRESIDENCY UNIVERSITY

Presidency University Act, 2013 of the Karnataka Act No. 41 of 2013 | Established under Section 2(f) of UGC Act, 1956
Approved by AICTE, New Delhi

Data Visualization Assignment

Sai Ram. K
20181CSE0621
7-CSE-10

Exercise-1 - Numpy

1. **Write a NumPy program to convert a list of numeric value into a one-dimensional NumPy array.**

Expected Output:

Original List: [12.23, 13.32, 100, 36.32]

One-dimensional NumPy array: [12.23 13.32 100. 36.32]

In [1]:
```
import numpy as np
lst = [12.23, 13.32, 100, 36.32]
print("Original List:",lst)
#a = np.array(l)
print("One-dimensional NumPy array: ",np.array(lst))
```

```
Original List: [12.23, 13.32, 100, 36.32]
One-dimensional NumPy array:  [ 12.23  13.32 100.    36.32]
```

2. Write a NumPy program to create a 3x3 matrix with values ranging from 2 to 10.

Expected Output:

[[2 3 4] [5 6 7] [8 9 10]]

In [2]:
```
import numpy as np
print(np.arange(2, 11).reshape(3,3))
```

```
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

3. *Write a NumPy program to sort an along the first, last axis of an array.**

Sample array: [[2,5],[4,4]]

Expected Output:

Original array: [[4 6] [2 1]]

Sort along the first axis: [[2 1] [4 6]]

Sort along the last axis: [[1 2] [4 6]]

In [3]:
```python
import numpy as np
a = np.array([[4, 6],[2, 1]])
print("Original array: ")
print(a)
print("Sort along the first axis: ")
x = np.sort(a, axis=0)
print(x)
print("Sort along the last axis: ")
y = np.sort(x, axis=1)
print(y)
```

```
Original array:
[[4 6]
 [2 1]]
Sort along the first axis:
[[2 1]
 [4 6]]
Sort along the last axis:
[[1 2]
 [4 6]]
```

4. **Write a NumPy program to create a contiguous flattened array.**

Expected Output:

Original array: [[10 20 30] [20 40 50]]

New flattened array: [10 20 30 20 40 50]

In [4]:
```python
import numpy as np
x = np.array([[10, 20, 30], [20, 40, 50]])
print("Original array:")
print(x)
y = np.ravel(x)
print("New flattened array:")
print(y)
```

```
Original array:
[[10 20 30]
 [20 40 50]]
New flattened array:
[10 20 30 20 40 50]
```

5. **Write a NumPy program to display all the dates for the month of March, 2017.**

Expected Output:

March, 2017

['2017-03-01' '2017-03-02' '2017-03-03' '2017-03-04' '2017-03-05' '2017-03-06' '2017-03-07' '2017-03-08' '2017-03-09' '2017-03-10' '2017-03-11' '2017-03-12' '2017-03-13' '2017-03-14' '2017-03-15' '2017-03-16' '2017-03-17' '2017-03-18' '2017-03-19' '2017-03-20' '2017-03-21' '2017-03-22' '2017-03-23' '2017-03-24' '2017-03-25' '2017-03-26' '2017-03-27' '2017-03-28' '2017-03-29' '2017-03-30' '2017-03-31']

In [5]:
```python
print(np.arange('2017-03', '2017-04', dtype='datetime64[D]'))
```

```
['2017-03-01' '2017-03-02' '2017-03-03' '2017-03-04' '2017-03-05'
 '2017-03-06' '2017-03-07' '2017-03-08' '2017-03-09' '2017-03-10'
 '2017-03-11' '2017-03-12' '2017-03-13' '2017-03-14' '2017-03-15'
 '2017-03-16' '2017-03-17' '2017-03-18' '2017-03-19' '2017-03-20'
 '2017-03-21' '2017-03-22' '2017-03-23' '2017-03-24' '2017-03-25'
 '2017-03-26' '2017-03-27' '2017-03-28' '2017-03-29' '2017-03-30'
 '2017-03-31']
```

6. **Write a NumPy program to generate six random integers between 10 and 30.**

Expected Output:

[20 28 27 17 28 29]

In [6]:
```python
x = np.random.randint(low=10, high=30, size=6)
print(x)
```

```
[26 15 16 29 19 12]
```

7. **Write a NumPy program to create a 5x5 array with random values and find the minimum and maximum values.**

Expected Output

Original Array:

[[0.96336355 0.12339131 0.20295196 0.37243578 0.88105252] [0.93228246 0.67470158 0.38103235 0.32242645 0.40610231] [0.3113495 0.31688 0.79189089 0.08676434 0.60829874] [0.30360149 0.94316317 0.98142491 0.77222542 0.51532195] [0.97392305 0.16669609 0.81377917 0.2165645 0.00121611]]

Minimum and Maximum Values:

0.00121610921757 0.981424910368

In [7]:
```python
import numpy as np
x = np.random.random((5,5))
print("Original Array:")
print(x)
xmin, xmax = x.min(), x.max()
print("Minimum and Maximum Values:")
print(xmin, xmax)
```

```
Original Array:
[[0.88318017 0.16058759 0.45361049 0.89883431 0.65591906]
 [0.2656476  0.39620209 0.50234847 0.70037713 0.47614227]
 [0.29906519 0.33867444 0.93651369 0.52909684 0.83765474]
 [0.59195027 0.75803818 0.65942243 0.37598773 0.65311666]
 [0.49746955 0.50738806 0.99571647 0.6706042  0.49886878]]
Minimum and Maximum Values:
0.16058758596874245 0.9957164706215259
```

8. **Write a NumPy program to get the powers of an array values element-wise.**

Note: First array elements raised to powers from second array

Expected Output:

Original array

[0 1 2 3 4 5 6]

First array elements raised to powers from second array, element-wise:

[0 1 8 27 64 125 216]

In [8]:
```python
import numpy as np
x = np.arange(7)
print("Original array")
print(x)
print("First array elements raised to powers from second array, element-wise:")
print(np.power(x, 3))
```

```
Original array
[0 1 2 3 4 5 6]
First array elements raised to powers from second array, element-wise:
[  0   1   8  27  64 125 216]
```

9. **Write a NumPy program to create a structured array from given student name, height, class and their data types. Now sort the array on height.**

Expected Output:

Original array:

[(b'James', 5, 48.5) (b'Nail', 6, 52.5) (b'Paul', 5, 42.1) (b'Pit', 5, 40.11)]

Sort by height

[(b'Pit', 5, 40.11) (b'Paul', 5, 42.1) (b'James', 5, 48.5) (b'Nail', 6, 52.5)]

In [9]:
```python
import numpy as np
data_type = [('name', 'S15'), ('class', int), ('height', float)]
students_details = [('James', 5, 48.5), ('Nail', 6, 52.5),('Paul', 5, 42.10), ('Pit
# create a structured array
students = np.array(students_details, dtype=data_type)
print("Original array:")
print(students)
print("Sort by height")
print(np.sort(students, order='height'))
```

```
Original array:
[(b'James', 5, 48.5 ) (b'Nail', 6, 52.5 ) (b'Paul', 5, 42.1 )
 (b'Pit', 5, 40.11)]
Sort by height
[(b'Pit', 5, 40.11) (b'Paul', 5, 42.1 ) (b'James', 5, 48.5 )
 (b'Nail', 6, 52.5 )]
```

10. **Write a NumPy program to sort the student id with increasing height of the students from given students id and height. Print the integer indices that describes the sort order by multiple columns and the sorted data.**

Expected Output:

Sorted indices

[4 0 5 3 6 1 2]

Sorted data:

1682 38.0 1023 40.0 5241 40.0 1671 41.0 4532 42.0 5202 42.0 6230 45.0

In [10]:
```python
import numpy as np
student_id = np.array([1023, 5202, 6230, 1671, 1682, 5241, 4532])
student_height = np.array([40., 42., 45., 41., 38., 40., 42.0])
#Sort by studen_id then by student_height
indices = np.lexsort((student_id, student_height))
print("Sorted indices:")
print(indices)
print("Sorted data:")
for n in indices:
  print(student_id[n], student_height[n])
```

```
Sorted indices:
[4 0 5 3 6 1 2]
Sorted data:
1682 38.0
1023 40.0
5241 40.0
1671 41.0
4532 42.0
5202 42.0
6230 45.0
```

Exercise-2 - Pandas

1. **Consider the Data given below**

sales = [100,130,119,92,35]

customer_account = ['B100','J101','X102','P103','R104']

city = ['BOS','LA','NYC','SF','CHI']

2. **Create the DataFrame with the above data**

In [11]:
```python
import pandas as pd
sales = [100,130,119,92,35]
customer_account = ['B100','J101','X102','P103','R104']
city = ['BOS','LA','NYC','SF','CHI']
d = {"sales":sales,"customer_account":customer_account,"city":city}
df = pd.DataFrame(d)
print(df)
```

```
   sales customer_account city
0    100             B100  BOS
1    130             J101   LA
2    119             X102  NYC
3     92             P103   SF
4     35             R104  CHI
```

3. **What is the name of the First Column.**

In [12]:
```
print(df.columns[0])
```
```
sales
```

4. **Sort the DataFrame by city in descending order (check the documentation for sort).**

In [13]:
```
df.sort_values(ascending=False,inplace=True,by=["sales"])
df
```
Out[13]:

	sales	customer_account	city
1	130	J101	LA
2	119	X102	NYC
0	100	B100	BOS
3	92	P103	SF
4	35	R104	CHI

5. **Which customer is in the last row of the DataFrame?.**

In [14]:
```
print(df.iloc[-1,])
```
```
sales                  35
customer_account     R104
city                  CHI
Name: 4, dtype: object
```

6. **Reorder the columns with customer in the first column.**

In [15]:
```
column_names = ["city", "customer_account", "sales"]
df=df[["customer_account", "city", "sales"]]
df
```
Out[15]:

	customer_account	city	sales
1	J101	LA	130
2	X102	NYC	119
0	B100	BOS	100
3	P103	SF	92
4	R104	CHI	35

7. **Write a Python program to add, subtract, multiple and divide two Pandas Series.**

Sample Series: [2, 4, 6, 8, 10], [1, 3, 5, 7, 9]

```
In [16]:
import pandas as pd
a = pd.Series([20, 40, 60, 80, 100])
b = pd.Series([10, 30, 50, 70, 90])
print(f"Addition :-\n {a+b}")
print(f"\nSubtraction :-\n {a-b}")
print(f"\nMultiplication :-\n {a*b}")
print(f"\nDivision :-\n {a/b}")
```

```
Addition :-
 0     30
1     70
2    110
3    150
4    190
dtype: int64

Subtraction :-
 0    10
1    10
2    10
3    10
4    10
dtype: int64

Multiplication :-
 0     200
1    1200
2    3000
3    5600
4    9000
dtype: int64

Division :-
 0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    1.111111
dtype: float64
```

8. **Write a Pandas program to sort a given Series.** Sample Output:

Original Data Series:	Output Data Series:
0 100	0 100
1 200	1 200
2 python	3 300.12
3 300.12	4 400
4 400	2 python

In [17]:
```python
import pandas as pandas
s = pd.Series(['100', '200', 'python', '300.12', '400'])
print("Original Data Series:")
print(s)
new_s = pd.Series(s).sort_values()
print(new_s)
```

```
Original Data Series:
0       100
1       200
2    python
3    300.12
4       400
dtype: object
0       100
1       200
3    300.12
4       400
2    python
dtype: object
```

9. **Write a Pandas program to compute the minimum, 25th percentile, median, 75th, and maximum of a given series.**

In [18]:
```python
import pandas as pd
import numpy as np
x = np.random.RandomState(100)
y = pd.Series(x.normal(10, 4, 20))
print("Original Series:")
print(y)
result = np.percentile(y, q=[0, 25, 50, 75, 100])
print("\nMinimum, 25th percentile, median, 75th, and maximum of a given series:")
print(result)
```

```
Original Series:
0      3.000938
1     11.370722
2     14.612143
3      8.990256
4     13.925283
5     12.056875
6     10.884719
7      5.719827
8      9.242017
9     11.020006
10     8.167892
11    11.740654
12     7.665620
13    13.267388
14    12.690883
15     9.582355
16     7.874878
17    14.118931
18     8.247458
19     5.526727
dtype: float64

Minimum, 25th percentile, median, 75th, and maximum of a given series:
[ 3.00093811  8.09463867 10.23353705 12.21537733 14.61214321]
```

10. **Write a Pandas program to find the positions of numbers that are multiples of 5 of a given series.**

In [19]:
```python
import pandas as pd
import numpy as np
s = pd.Series(np.random.randint(1, 25,15))
print(s[s%5==0])
```

```
0     15
11     5
dtype: int32
```

11. **Write a Pandas program to display a summary of the basic information about a specified DataFrame and its data.**

Sample Python dictionary data and list labels:

```
     exam_data =
           {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michae
l', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
            'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
            'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
            'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no',
 'yes']}
             labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

In [20]:
```python
exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes'
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)

print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, a to j
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   name      10 non-null     object
 1   score     8 non-null      float64
 2   attempts  10 non-null     int64
 3   qualify   10 non-null     object
dtypes: float64(1), int64(1), object(2)
memory usage: 400.0+ bytes
None
```

12. **Write a Pandas program to convert all the string values to upper, lower cases in a given pandas series. Also find the length of the string values.**

In [21]:

```python
import pandas as pd
import numpy as np
s = pd.Series(['X', 'Y', 'Z', 'Aaba', 'Baca', np.nan, 'CABA', None, 'bird', 'horse'
print("Original series:")
print(s)
print("\nConvert all string values of the said Series to upper case:")
print(s.str.upper())
print("\nConvert all string values of the said Series to lower case:")
print(s.str.lower())
print("\nLength of the string values of the said Series:")
print(s.str.len())
```

```
Original series:
0         X
1         Y
2         Z
3      Aaba
4      Baca
5       NaN
6      CABA
7      None
8      bird
9     horse
10      dog
dtype: object

Convert all string values of the said Series to upper case:
0         X
1         Y
2         Z
3      AABA
4      BACA
5       NaN
6      CABA
7      None
8      BIRD
9     HORSE
10      DOG
dtype: object

Convert all string values of the said Series to lower case:
0         x
1         y
2         z
3      aaba
4      baca
5       NaN
6      caba
7      None
8      bird
9     horse
10      dog
dtype: object

Length of the string values of the said Series:
0     1.0
1     1.0
2     1.0
3     4.0
4     4.0
5     NaN
6     4.0
```

```
7     NaN
8     4.0
9     5.0
10    3.0
dtype: float64
```

13. **Write a Pandas program to check whether only numeric values present in a given column of a DataFrame.**

In [22]:
```python
import pandas as pd
df = pd.DataFrame({
    'company_code': ['Company','Company a001', '2055', 'abcd', '123345'],
    'date_of_sale ': ['12/05/2002','16/02/1999','25/09/1998','12/02/2022','15/09/19
    'sale_amount': [12348.5, 233331.2, 22.5, 2566552.0, 23.0]})

print("Original DataFrame:")
print(df)
print("\nNumeric values present in company_code column:")
df['company_code_is_digit'] = list(map(lambda x: x.isdigit(), df['company_code']))
print(df)
```

```
Original DataFrame:
   company_code date_of_sale   sale_amount
0       Company    12/05/2002      12348.5
1  Company a001    16/02/1999     233331.2
2          2055    25/09/1998         22.5
3          abcd    12/02/2022    2566552.0
4        123345    15/09/1997         23.0

Numeric values present in company_code column:
   company_code date_of_sale   sale_amount  company_code_is_digit
0       Company    12/05/2002      12348.5                  False
1  Company a001    16/02/1999     233331.2                  False
2          2055    25/09/1998         22.5                   True
3          abcd    12/02/2022    2566552.0                  False
4        123345    15/09/1997         23.0                   True
```

Exercise-3 - Matplotlib

1. **Create data frame for month and total profit, Read Total profit of all months and show it using a line plot Total profit data provided for each month.**

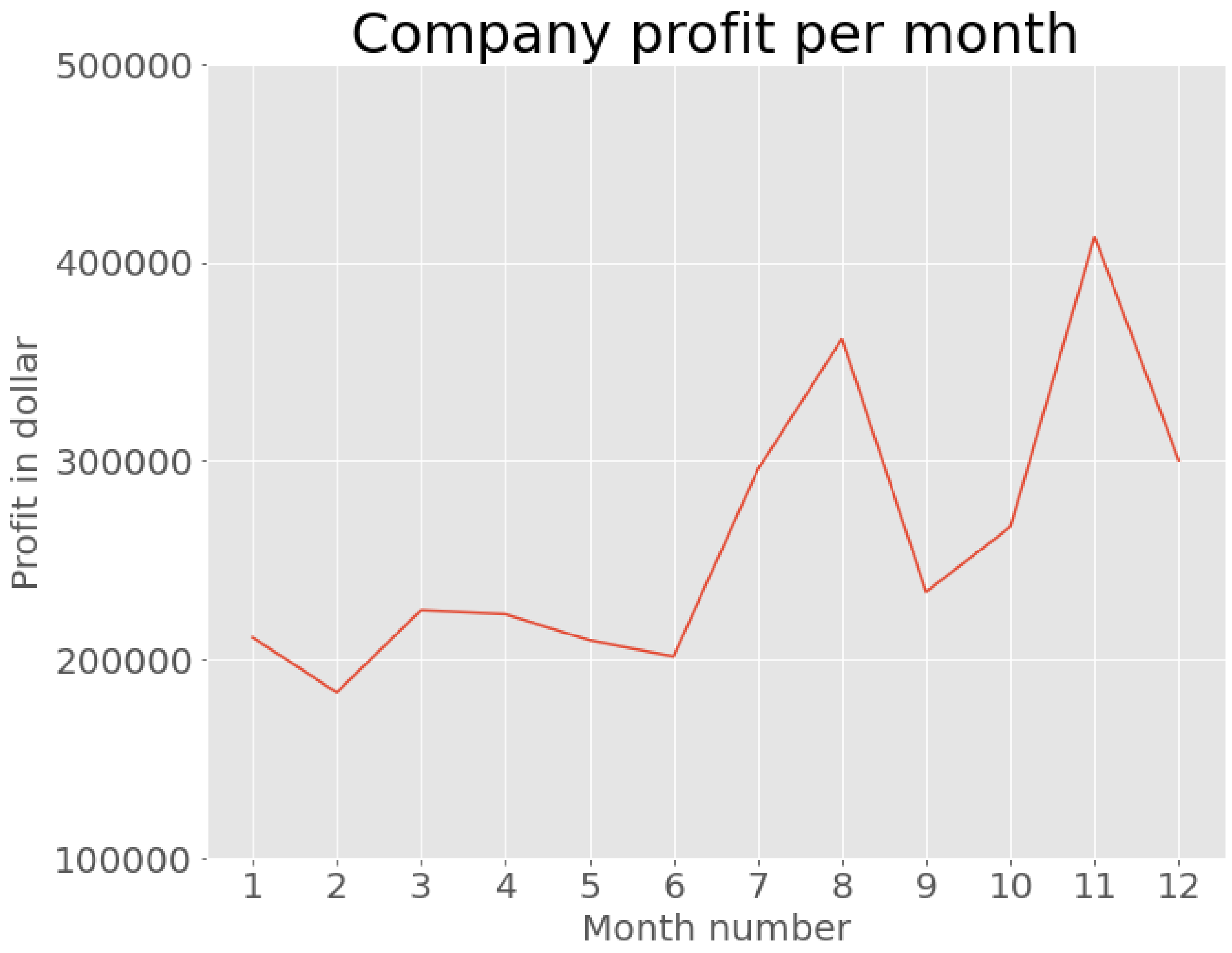
Generated line plot must include the following properties: –

X label name = Month Number

Y label name = Total profit

In [23]:

```python
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("company_sales_data.csv")
profitList = df ['total_profit'].tolist()
monthList  = df ['month_number'].tolist()
with plt.style.context('ggplot'):
  plt.figure(figsize=(10,8))
  plt.plot(monthList, profitList, label = 'Month-wise Profit data of last year')
  plt.xlabel('Month number',size=20);plt.ylabel('Profit in dollar',size=20)
  plt.xticks(monthList,size=20)
  plt.title('Company profit per month',size=30)
  plt.yticks([100000, 200000, 300000, 400000, 500000],size=20)
  plt.show()
```



2. **Get total profit of all months and show line plot with the following Style properties**

Generated line plot must include following Style properties: –

Line Style dotted and Line-color should be red

Show legend at the lower right location.

X label name = Month Number

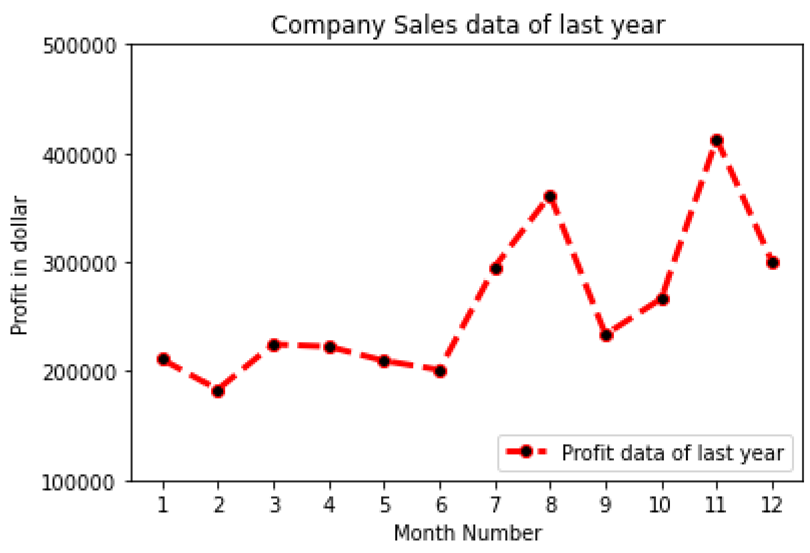
Y label name = Sold units number

Add a circle marker.

Line marker color as read

Line width should be 3

In [24]:
```python
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("company_sales_data.csv")
profitList = df ['total_profit'].tolist()
monthList  = df ['month_number'].tolist()

plt.plot(monthList, profitList, label = 'Profit data of last year',
      color='r', marker='o', markerfacecolor='k',
      linestyle='--', linewidth=3)

plt.xlabel('Month Number')
plt.ylabel('Profit in dollar')
plt.legend(loc='lower right')
plt.title('Company Sales data of last year')
plt.xticks(monthList)
plt.yticks([100000, 200000, 300000, 400000, 500000])
plt.show()
```



3. **Automobile Land Speed Records (GR 5-10)** In the first recorded automobile race in 1898, Count Gaston de Chasseloup-Laubat of Paris, France, drove 1 kilometer in 57 seconds for an average speed of 39.2 miles per hour(mph) or 63.1 kilometers per hour (kph).

In 1904, Henry Ford drove his Ford Arrow across frozen Lake St. Clair, MI, at an average speed of 91.4 mph. Now, the North American Eagle is trying to break a land speed record of 800 mph.

The Federation International deL'Automobile (FIA), the world's governing body for motor sport and land speed records,recorded the following land speed records.

[Land Records Data Set (https://drive.google.com/file/d/1-1aCUkshQPYgErMrKTB4MzPU_c_ypi1K/view?usp=sharing)](https://drive.google.com/file/d/1-1aCUkshQPYgErMrKTB4MzPU_c_ypi1K/view?usp=sharing)

Use the above dataset to show your innovative skills using Bar Plots.

In [25]:
```
land = pd.read_csv(r'LandRecords (1).csv')
land.columns
```

Out[25]:
```
Index(['Speed (mph)', 'Driver', 'Car', 'Engine', 'Date'], dtype='object')
```

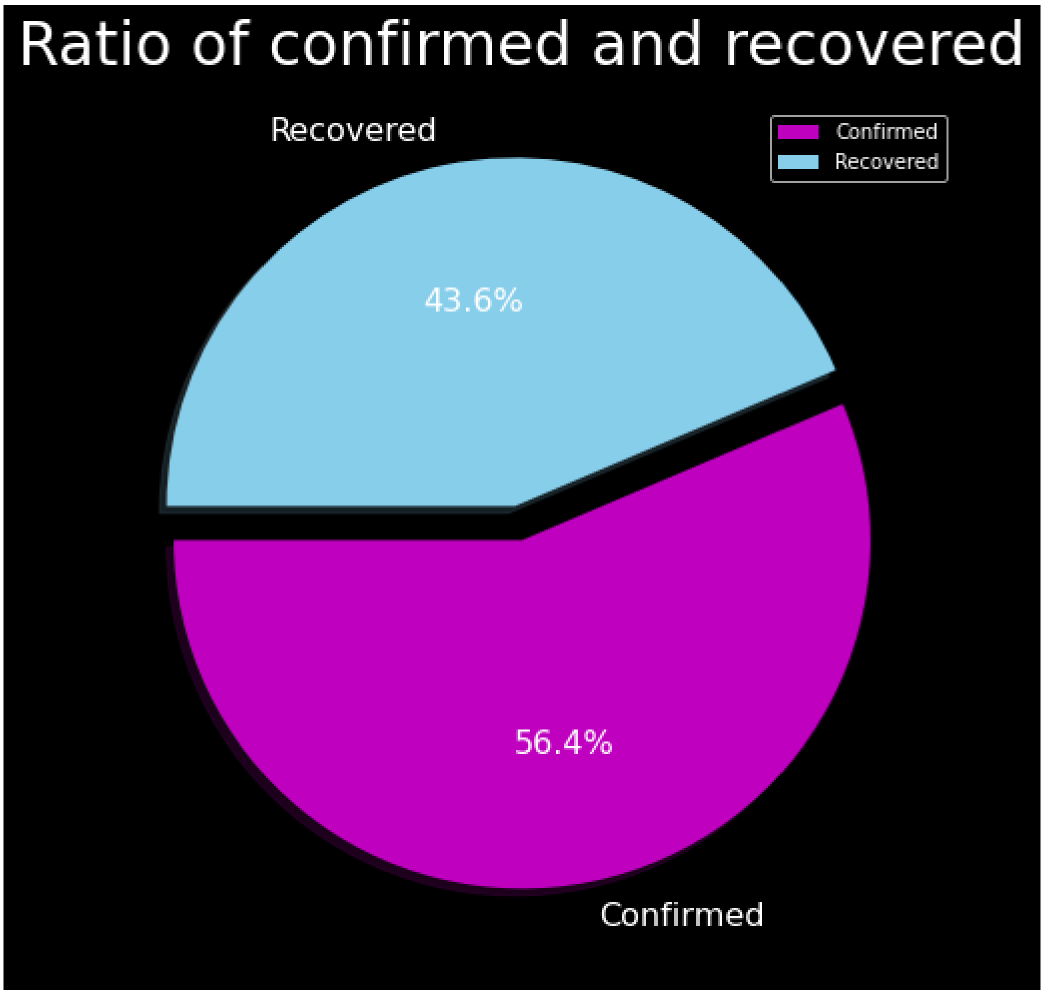
In [ ]:
```
with plt.style.context('dark_background'):
  fig = plt.figure(figsize=(10,6))
  plt.ylabel("Speed",size=20,color='yellow',labelpad=20) ; plt.xlabel("Engine",size
  plt.title('Speed of Engines',size=20,color='yellow',pad=20)
  plt.xticks(fontsize=16,color='yellow') ; plt.yticks(fontsize=16,color='palegreen'
  plt.bar(land['Engine'],land['Speed (mph)'],color='cyan')
```

4. **Covid 19 India Data as on 5th Sept 2020**

[Covid 19 - India Data Set (https://drive.google.com/file/d/1zXaeHnNPQL7vCPQ3JU61adU17rG9aluo/view?usp=sharing)](https://drive.google.com/file/d/1zXaeHnNPQL7vCPQ3JU61adU17rG9aluo/view?usp=sharing)

Use the above dataset to show your innovative skills using Pie Charts.

In [26]:
```python
cov = pd.read_csv(r'Covid_19.csv')
c =  cov['total__confirmed'].sum()
r = cov['total__recovered'] .sum()
with plt.style.context('dark_background'):
  fig = plt.figure(figsize =(8, 8))
  labels = ["Confirmed","Recovered"] ; explode =(0,0.1)
  plt.pie([c,r],labels=labels,colors=['m','skyblue'],
          startangle=180,explode=explode, autopct='%.1f%%', shadow = True,textprops
  plt.legend(labels, loc = 'upper right',prop={'size': 10})
  plt.title('Ratio of confirmed and recovered',pad=20,size=30)
```



Exercise-4 - Histogram

1. **Show your creatviity using Histogram with Covid Data.**

Plot Histograms(Atleast 2 Plots) on Covid Data.

In [27]:

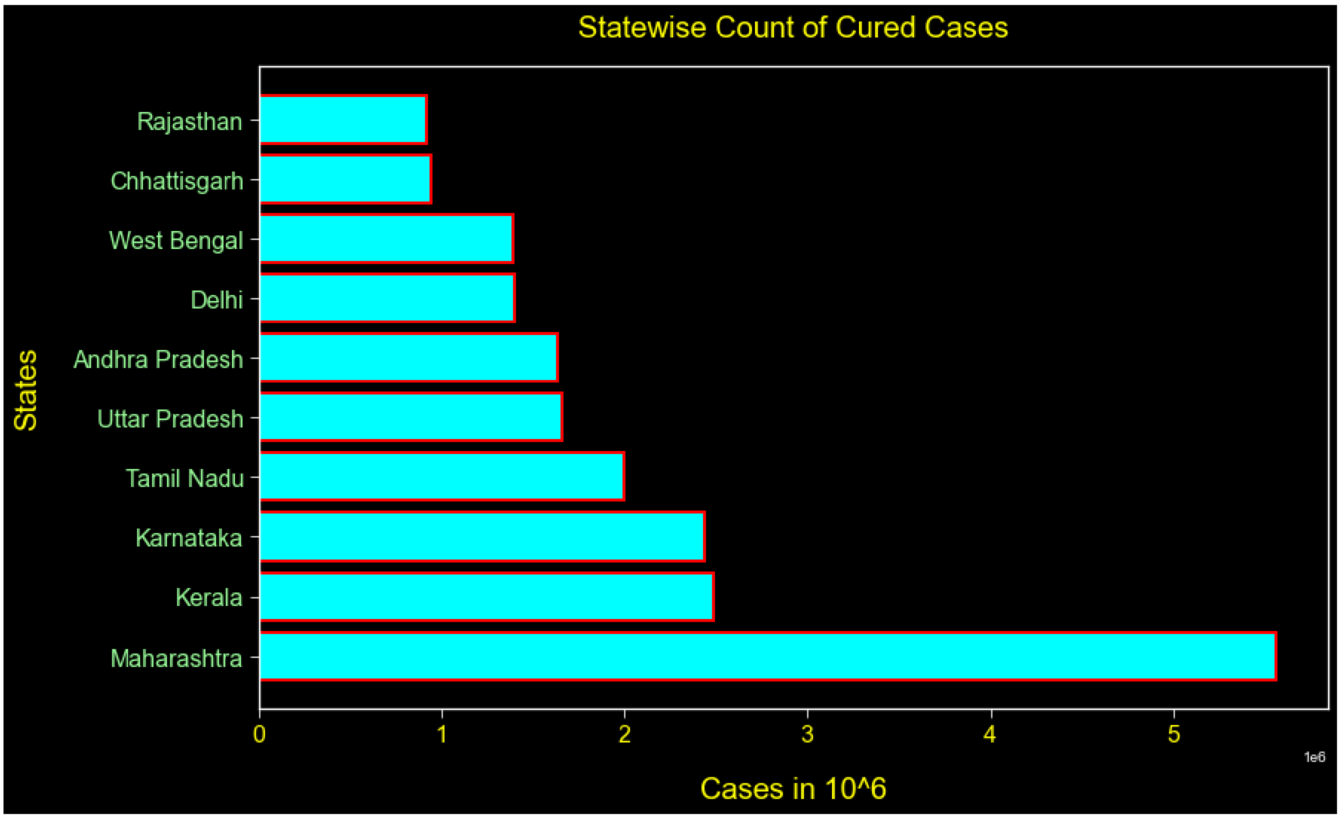
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import warnings; warnings.simplefilter('ignore')
plt.style.use(['dark_background', 'ggplot','seaborn-ticks'])
df = pd.read_csv(r'covid_19_india.csv')
df.head()
df['Date'] = pd.to_datetime(df['Date'])
india_cases = df[df['Date'] == df['Date'].max()].copy().fillna(0)
india_cases.index = india_cases["State/UnionTerritory"]
india_cases = india_cases.drop(['State/UnionTerritory','Date'], axis=1)
india_cases.head()
```

Out[27]:

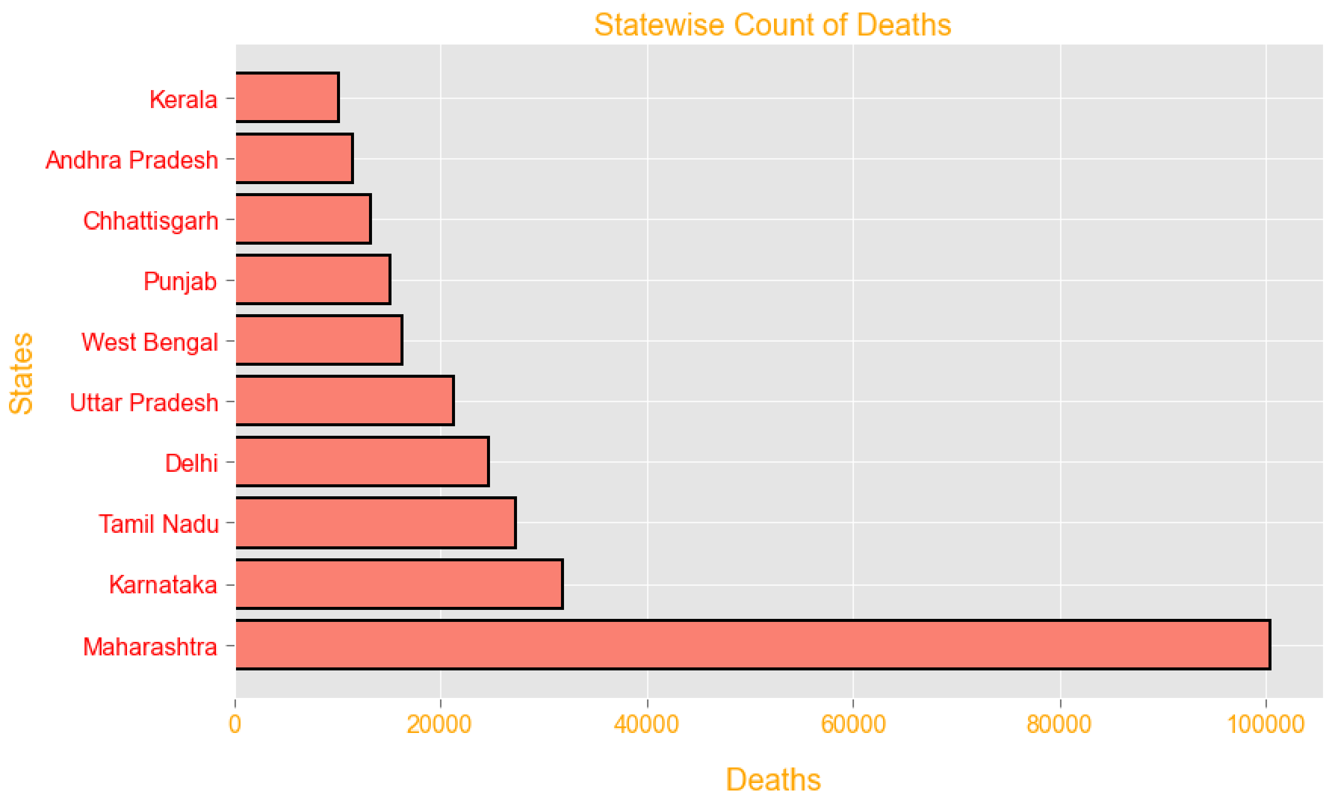
	Sno	Time	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Con
State/UnionTerritory							
Andaman and Nicobar Islands	15771	8:00 AM	-	-	6912	123	
Andhra Pradesh	15772	8:00 AM	-	-	1637149	11552	1
Arunachal Pradesh	15773	8:00 AM	-	-	26131	125	
Assam	15774	8:00 AM	-	-	385032	3695	
Bihar	15775	8:00 AM	-	-	700224	5424	

In [28]:

```python
top_10_cured_cases = india_cases.sort_values('Cured',ascending = False)[:10]
with plt.style.context('dark_background'):
  top_10_cured_cases.head()
  fig = plt.figure(figsize=(10,6))
  ax = fig.add_axes([0,0,1,1])
  ax.barh(top_10_cured_cases.index,top_10_cured_cases.Cured,color='cyan',edgecolor=
  plt.ylabel("States",size=20,color='yellow',labelpad=20) ; plt.xlabel("Cases in 10
  plt.title('Statewise Count of Cured Cases',size=20,color='yellow',pad=20)
  plt.xticks(fontsize=16,color='yellow') ; plt.yticks(fontsize=16,color='palegreen'
  plt.tight_layout()
  plt.show()
```



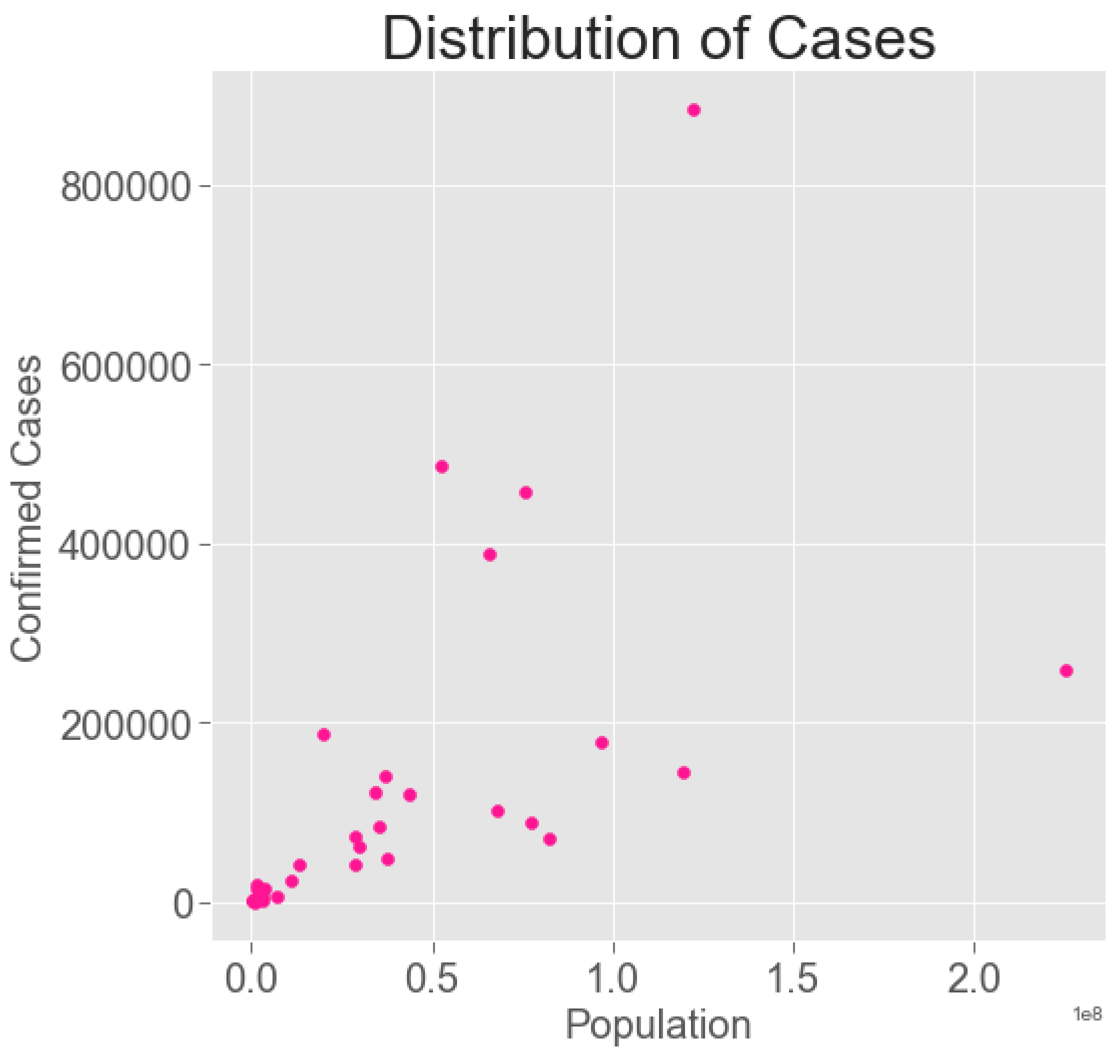
In [29]:
```
top_10_death_states = india_cases.sort_values('Deaths',ascending = False)[:10]

with plt.style.context('ggplot'):
  top_10_death_states.head()
  fig = plt.figure(figsize=(10,6))
  ax = fig.add_axes([0,0,1,1])
  ax.barh(top_10_death_states.index,top_10_death_states.Deaths,color='salmon',edgec
  plt.ylabel("States",size=20,color='Orange') ; plt.xlabel("Deaths ",size=20,color=
  plt.title('Statewise Count of Deaths ',size=20,color='Orange')
  plt.xticks(fontsize=16,color='Orange') ; plt.yticks(fontsize=16,color='red')
  plt.tight_layout()
plt.show()
```



2. **Show your creatviity using Scatter Plot with Covid Data.**

Plot Scatter Plots(Atleast 2 Plots) on Covid Data.

In [30]:
```
new_df = pd.read_csv(r'Covid_19.csv')
with plt.style.context('ggplot'):
  plt.figure(figsize=(8,8))
  plt.xlabel("Population",size=20);plt.ylabel("Confirmed Cases",size=20);
  plt.title("Distribution of Cases",size=30)
  plt.xticks(size=20) ; plt.yticks(size=20)
  plt.scatter(new_df['population'],new_df['total__confirmed'],color='deeppink')
  plt.show()
```



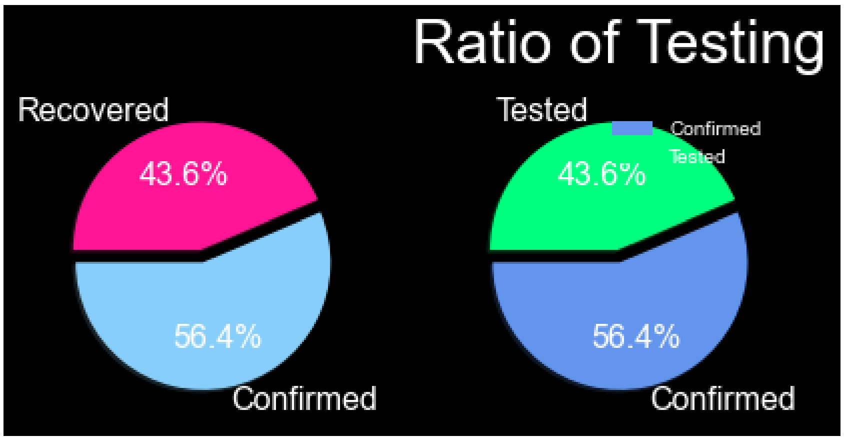
3. **Show your creatviity using Sub Plot with Covid Data.**

Plot SubPlots(Atleast 2 Plots) on Covid Data.

In [ ]:
```
new_df.columns
```

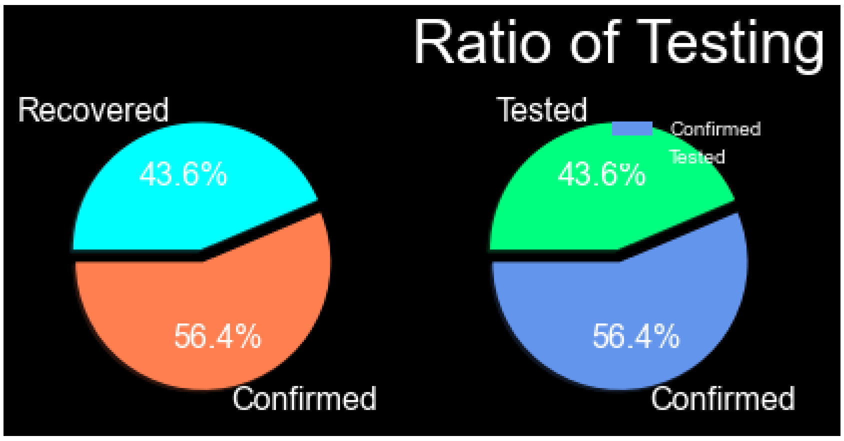
In [31]:
```
c =  new_df['total__confirmed'].sum()
r = new_df['total__recovered'] .sum()
c1 =  new_df['total__confirmed'].sum()
r1 = new_df['total__tested'] .sum()
with plt.style.context('dark_background'):
  plt.figure(figsize=(6, 4))
  fig, (ax1, ax2)  = plt.subplots(1, 2,sharey='row')
  labels = ["Confirmed","Recovered"] ; explode =(0,0.1)
  ax1.pie([c,r],labels=labels,colors=['lightskyblue','deeppink'],
          startangle=180,explode=explode, autopct='%.1f%%', shadow = True,textprops
  plt.legend(labels, loc = 'upper right',prop={'size': 10})
  plt.title('Ratio of confirmed and recovered',pad=20,size=30)

  labels = ["Confirmed","Tested"];explode =(0,0.1)
  ax2.pie([c,r],labels=labels,colors=['cornflowerblue','springgreen'],
          startangle=180,explode=explode, autopct='%.1f%%', shadow = True,textprops
  plt.legend(labels, loc = 'upper right',prop={'size': 10})
  plt.title('Ratio of Testing',pad=20,size=30)
  plt.tight_layout()
  plt.show()


```

```
<Figure size 432x288 with 0 Axes>
```



In [32]:
```
c =  new_df['total__confirmed'].sum()
r = new_df['total__recovered'] .sum()
c1 =  new_df['total__confirmed'].sum()
r1 = new_df['total__tested'] .sum()
with plt.style.context('dark_background'):
  plt.figure(figsize=(6, 4))
  fig, (ax1, ax2)  = plt.subplots(1, 2,sharey='row')
  labels = ["Confirmed","Recovered"] ; explode =(0,0.1)
  ax1.pie([c,r],labels=labels,colors=['coral','cyan'],
          startangle=180,explode=explode, autopct='%.1f%%', shadow = True,textprops
  plt.legend(labels, loc = 'upper right',prop={'size': 10})
  plt.title('Ratio of confirmed and recovered',pad=20,size=30)

  labels = ["Confirmed","Tested"];explode =(0,0.1)
  ax2.pie([c,r],labels=labels,colors=['cornflowerblue','springgreen'],
          startangle=180,explode=explode, autopct='%.1f%%', shadow = True,textprops
  plt.legend(labels, loc = 'upper right',prop={'size': 10})
  plt.title('Ratio of Testing',pad=20,size=30)
  plt.tight_layout()
  plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```


Ratio of Testing
Recovered
Confirmed
Tested
Confirmed
Tested
Confirmed