Pyber Challenge

4.3 Loading and Reading CSV files

In [1]:

```
# Add Matplotlib inline magic command
%matplotlib inline
# Dependencies and Setup
import matplotlib.pyplot as plt
import pandas as pd

# File to Load (Remember to change these)
city_data_to_load = "city_data.csv"
ride_data_to_load = "ride_data.csv"

# Read the City and Ride Data
city_data_df = pd.read_csv(city_data_to_load)
ride_data_df = pd.read_csv(ride_data_to_load)
```

Merge the DataFrames

In [2]:

```
# Combine the data into a single dataset
pyber_data_df = pd.merge(ride_data_df, city_data_df, how="left", on=["city",
"city"])

# Display the data table for preview
pyber_data_df.head()
```

Out[2]:

	city	date	fare	ride_id	driver_count	type
0	Lake Jonathanshire	1/14/19 10:14	13.83	5.739410e+12	5	Urban
1	South Michelleport	3/4/19 18:24	30.24	2.343910e+12	72	Urban
2	Port Samanthamouth	2/24/19 4:29	33.44	2.005070e+12	57	Urban
3	Rodneyfort	2/10/19 23:22	23.44	5.149250e+12	34	Urban
4	South Jack	3/6/19 4:28	34.58	3.908450e+12	46	Urban

Deliverable 1: Get a Summary DataFrame

In [3]:

```
#  1. Get the total rides for each city type
```

In [4]:

```
# 2. Get the total drivers for each city type
```

In [5]:

```
#  3. Get the total amount of fares for each city type
```

In [6]:

```
#  4. Get the average fare per ride for each city type.
```

In [7]:

```
# 5. Get the average fare per driver for each city type.
```

In [8]:

```
#  6. Create a PyBer summary DataFrame.
```

In [9]:

```
#  7. Cleaning up the DataFrame. Delete the index name
pyber_summary_df.index.name = None
```

In [10]:

```
#  8. Format the columns.
```

Deliverable 2. Create a multiple line plot that shows the total weekly of the fares for each type of city.

In [11]:

```
# 1. Read the merged DataFrame
```

In [12]:

```
# 2. Using groupby() to create a new DataFrame showing the sum of the fares
#  for each date where the indices are the city type and date.
```

In [13]:

```
# 3. Reset the index on the DataFrame you created in #1. This is needed to
use the 'pivot()' function.
# df = df.reset_index()
```

In [14]:

```
# 4. Create a pivot table with the 'date' as the index, the columns ='type',
and values='fare'
# to get the total fares for each type of city by the date.
```

In [15]:

```
# 5. Create a new DataFrame from the pivot table DataFrame using loc on the
given dates, '2019-01-01':'2019-04-29'.
```

In [16]:

```
# 6. Set the "date" index to datetime datatype. This is necessary to use the
resample() method in Step 8.
# df.index = pd.to_datetime(df.index)
```

In [17]:

```
# 7. Check that the datatype for the index is datetime using df.info()
```

In [18]:

```
# 8. Create a new DataFrame using the "resample()" function by week 'W' and
get the sum of the fares for each week.
```

In [19]:

```
# 8. Using the object-oriented interface method, plot the resample DataFrame
using the df.plot() function.

# Import the style from Matplotlib.
from matplotlib import style
# Use the graph style fivethirtyeight.
style.use('fivethirtyeight')
```

```
%matplotlib inline
```

In [4]:

```
# Import dependencies.
import matplotlib.pyplot as plt
```

In [3]:

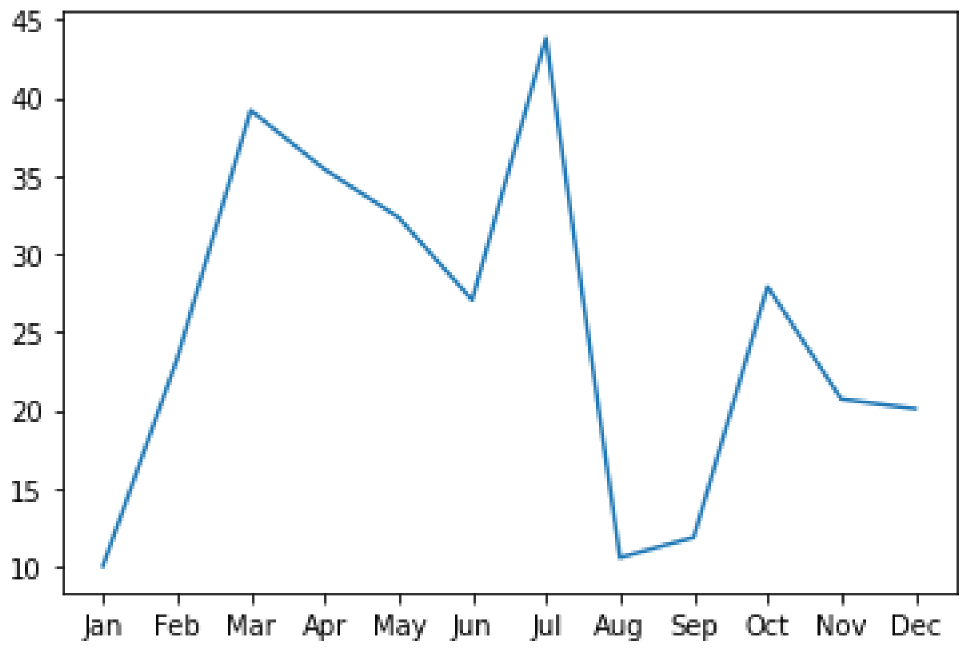
```
#Set the x-axis to alist of strings for each month
x_axis = ["Jan", "Feb",
"Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"]
y_axis =
[10.02,23.24,39.20,35.42,32.34,27.04,43.82,10.56,11.85,27.90,20.71,20.09 ]
```

In [4]:

```
# Create the plot
plt.plot(x_axis, y_axis)
```

Out[4]:

```
[<matplotlib.lines.Line2D at 0x240f92e68b0>]
```



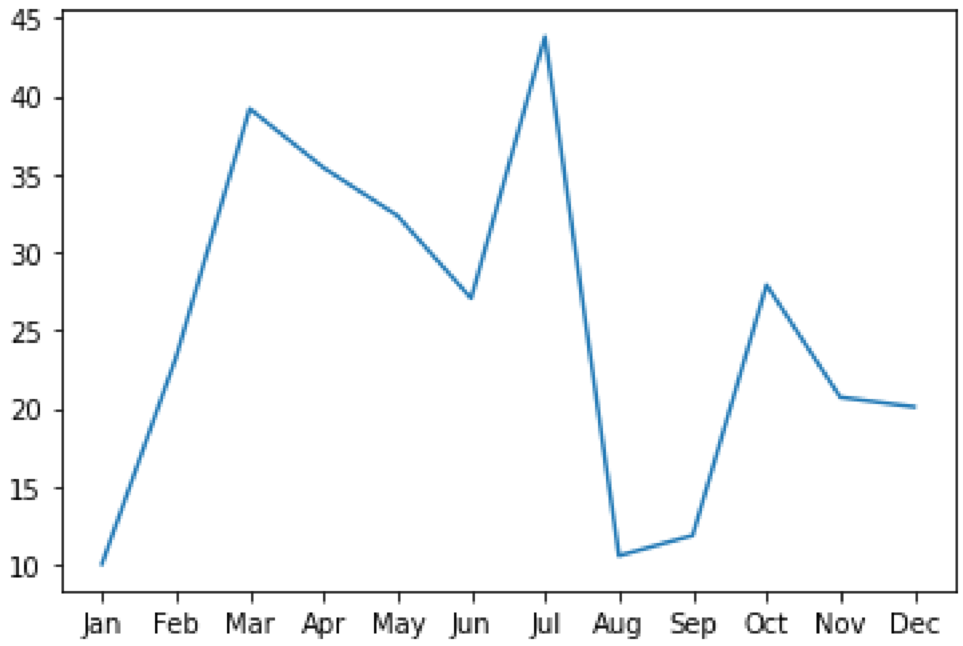
In [5]:

```
#Create the plot with object oriented programming
```

```
fig, ax = plt.subplots()
ax.plot(x_axis, y_axis)
```

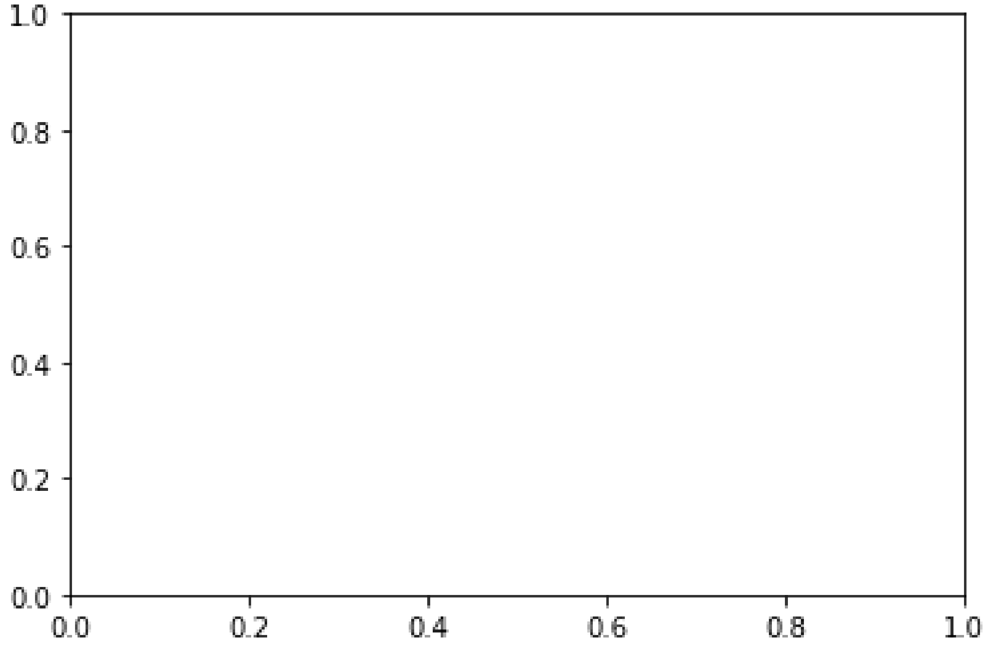
Out[5]:

```
[<matplotlib.lines.Line2D at 0x240f9ab0190>]
```



In [8]:

```
# Create the plot with ax.plt()
fig = plt.figure()
ax = fig.add_subplot()
```



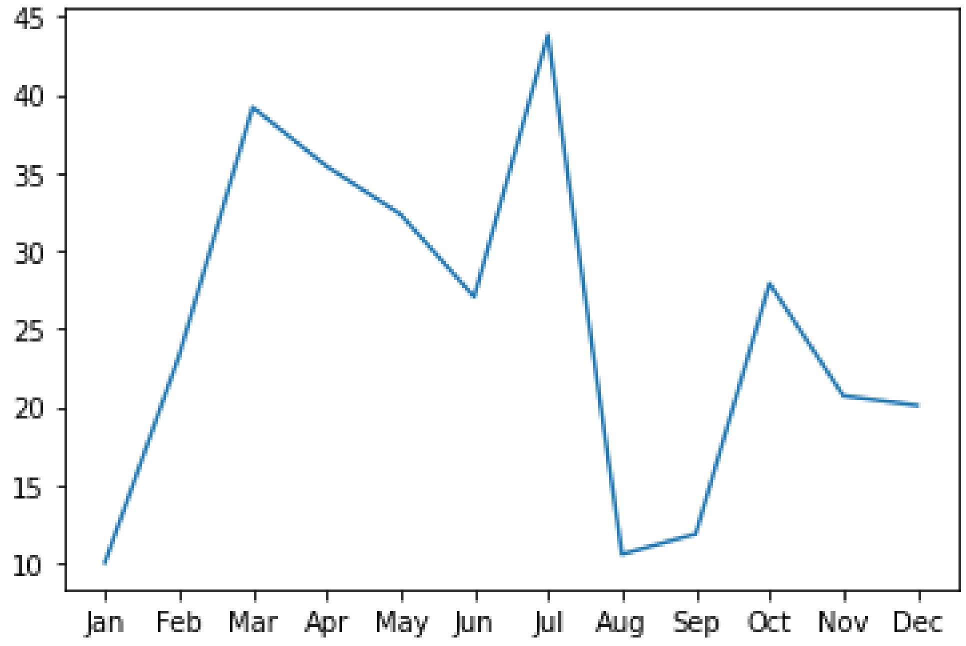
In [9]:

```
# Create the plot with ax.plt()
```

```
fig = plt.figure()
ax = fig.add_subplot()
ax.plot(x_axis, y_axis)
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x240f9d47190>]
```

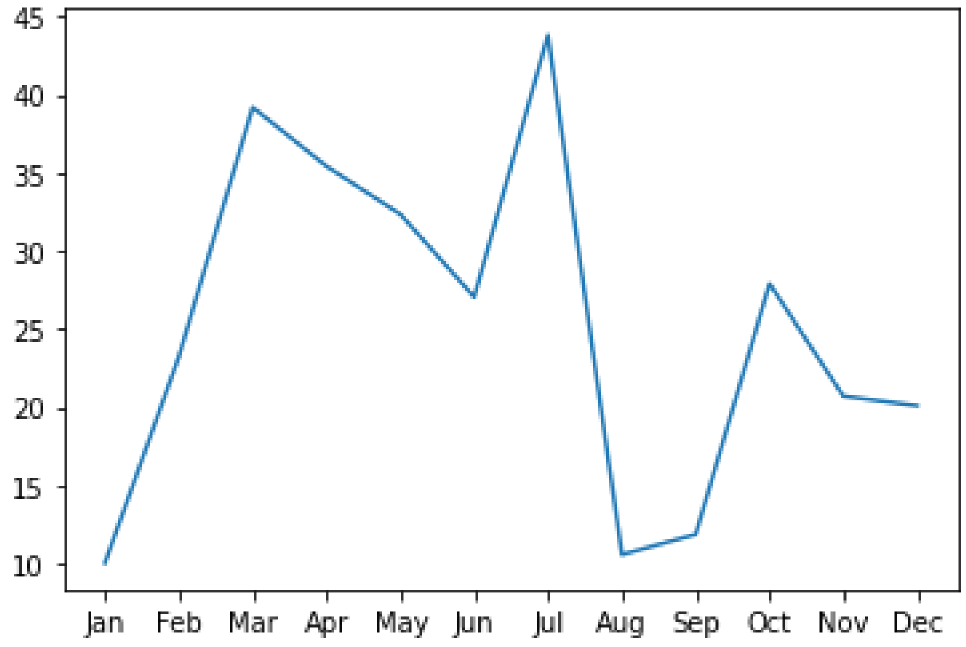


In [10]:

```
#Create the plot with ax.plt
ax = plt.axes()
ax.plot(x_axis, y_axis)
```

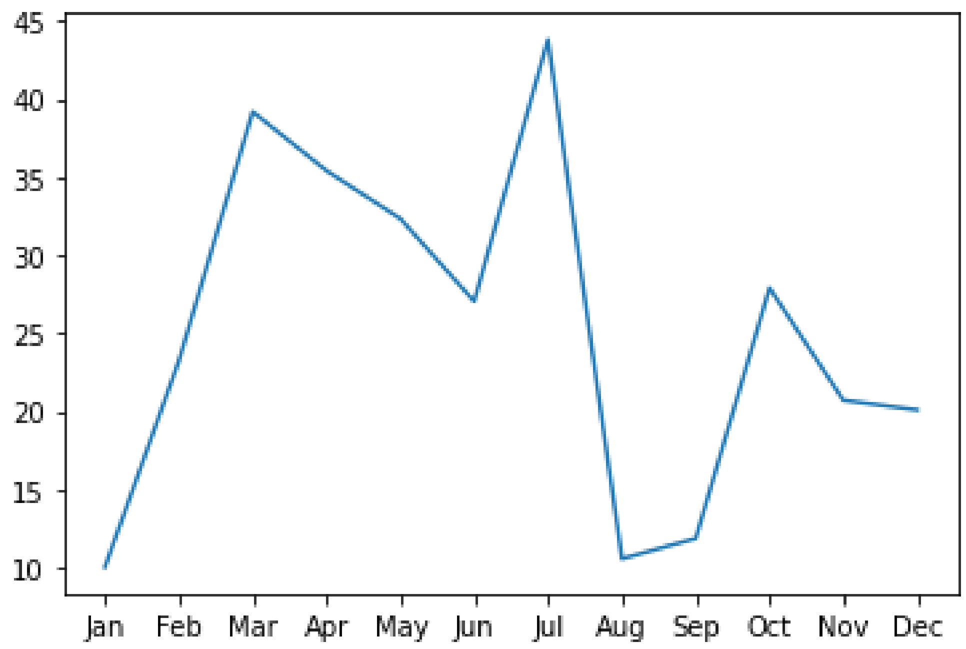
Out[10]:

```
[<matplotlib.lines.Line2D at 0x240f9db72b0>]
```



In [12]:

```
#Create the plot with ax.plt
plt.plot(x_axis, y_axis)
plt.show()
```

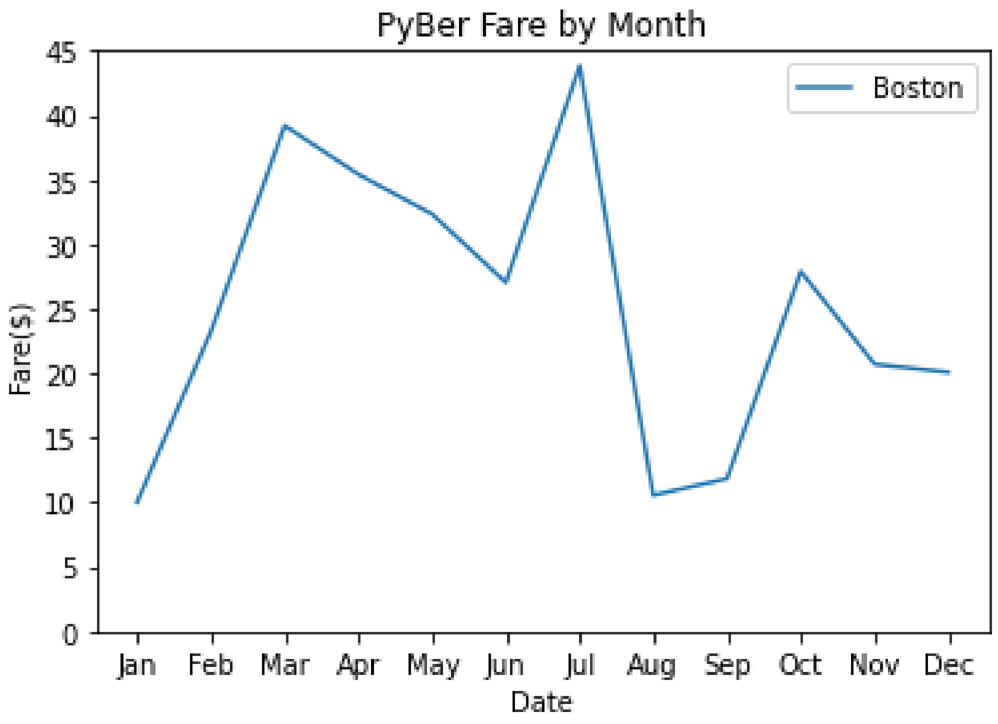


In [16]:

```
#Create the plot and add a label for the legend
plt.plot(x_axis, y_axis, label = "Boston")

#Create lables for x and y axes
plt.xlabel("Date")
plt.ylabel("Fare($)")

#Set the y limit between 0 and 45
plt.ylim(0,45)

#Create a title
plt.title("PyBer Fare by Month")

#Add the legend
plt.legend()
```

Out[16]:

```
<matplotlib.legend.Legend at 0x240fb01cbb0>
```

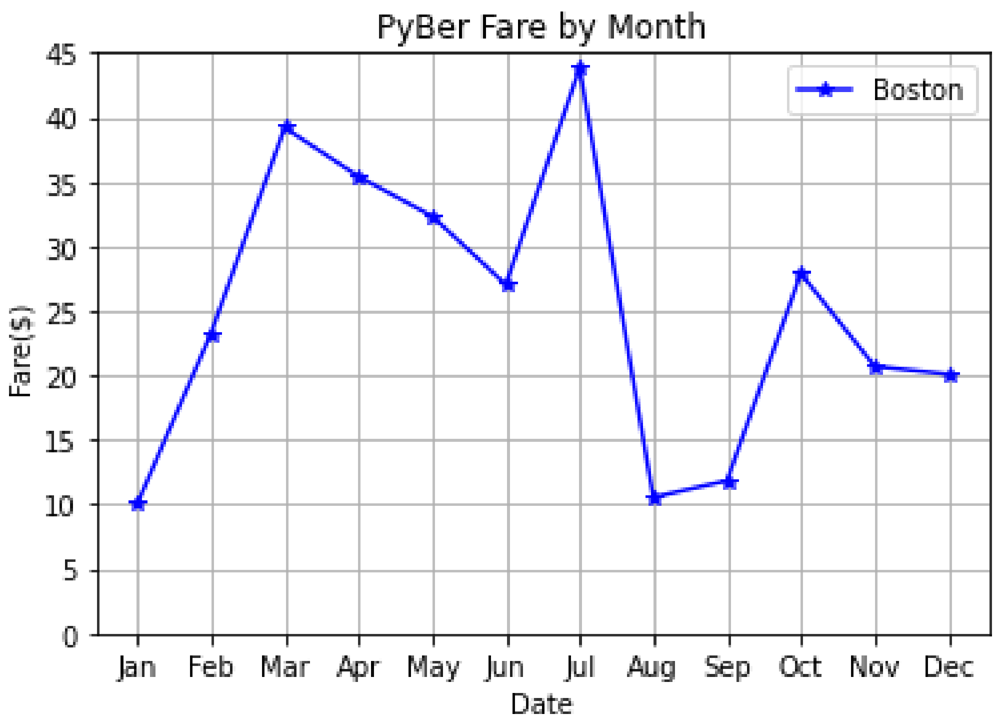


In [23]:

```
# Create the plot and add a label for legend and add color
plt.plot(x_axis, y_axis,marker = "*" , color = "blue", label ="Boston")

# Create labels for x and y axis
plt.xlabel("Date")
plt.ylabel("Fare($)")

#Set the y limit  between 0 and 45
plt.ylim(0,45)

#Create title
plt.title("PyBer Fare by Month")

#Add a grid
plt.grid()

#Add legend
plt.legend()
```

Out[23]:

```
<matplotlib.legend.Legend at 0x240fb23fa30>
```

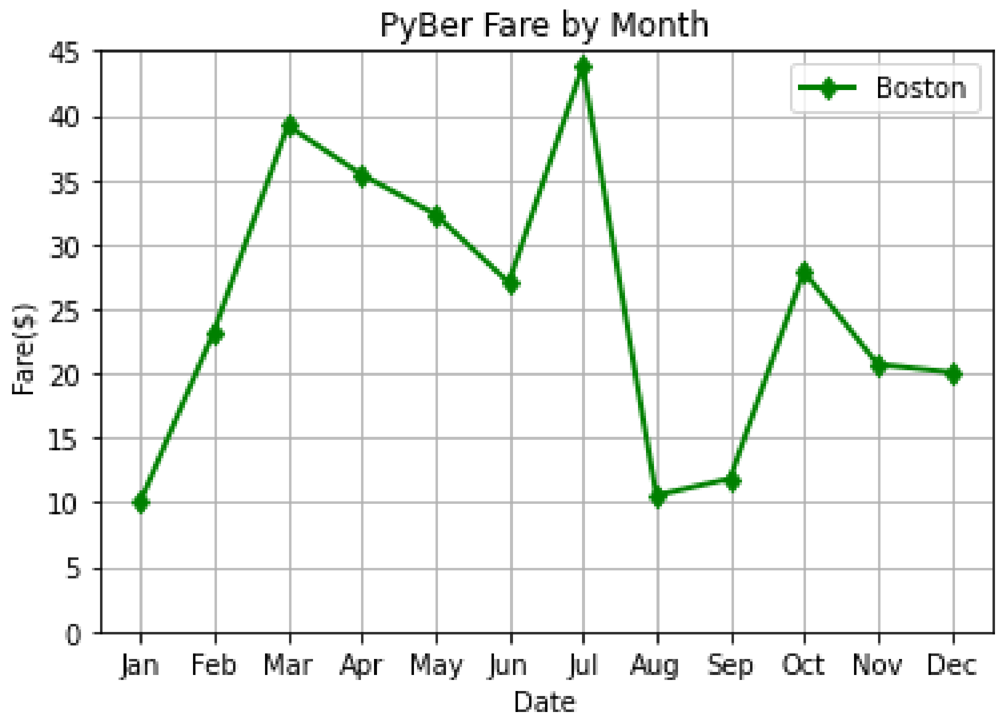


In [33]:

```
# Create the plot and add a label for legend and add color
fig, ax = plt.subplots()
ax.plot(x_axis, y_axis, marker = "d", color = "green", label ="Boston",
linewidth=2)

# Create labels for x and y axis
ax.set_xlabel("Date")
ax.set_ylabel("Fare($)")

#Set the y limit  between 0 and 45
ax.set_ylim(0,45)

#Create title
ax.set_title("PyBer Fare by Month")

#Add a grid
ax.grid()

#Add legend
ax.legend()
```

Out[33]:

```
<matplotlib.legend.Legend at 0x240fb702670>
```



In [34]:

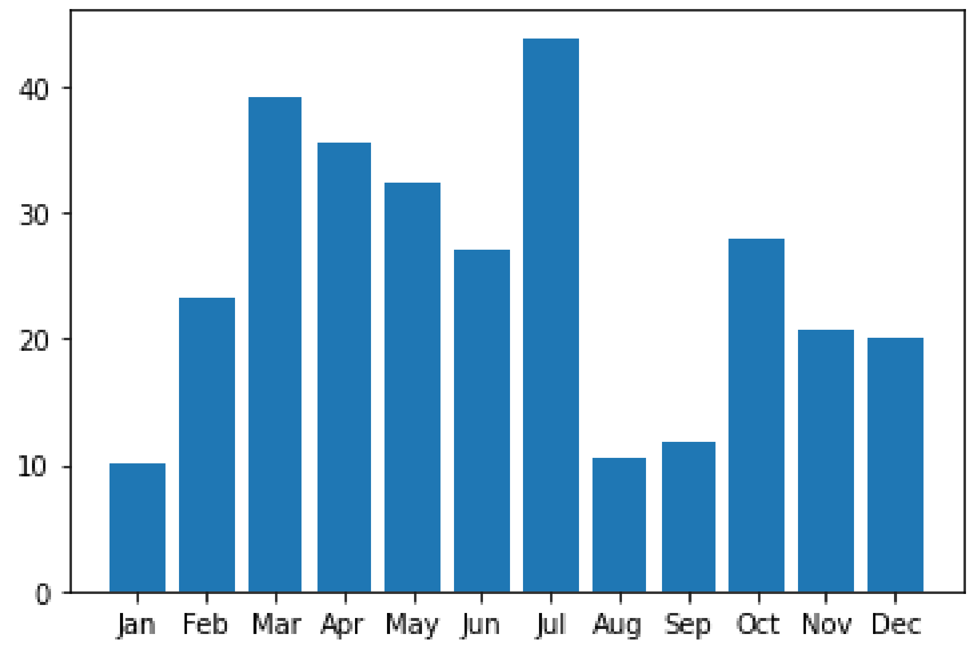
```
#Set the x-axis to alist of strings for each month
x_axis = ["Jan", "Feb",
"Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"]
y_axis =
[10.02,23.24,39.20,35.42,32.34,27.04,43.82,10.56,11.85,27.90,20.71,20.09 ]
```

In [35]:

```
# Create the plot
plt.bar(x_axis, y_axis)
```

Out[35]:

```
<BarContainer object of 12 artists>
```

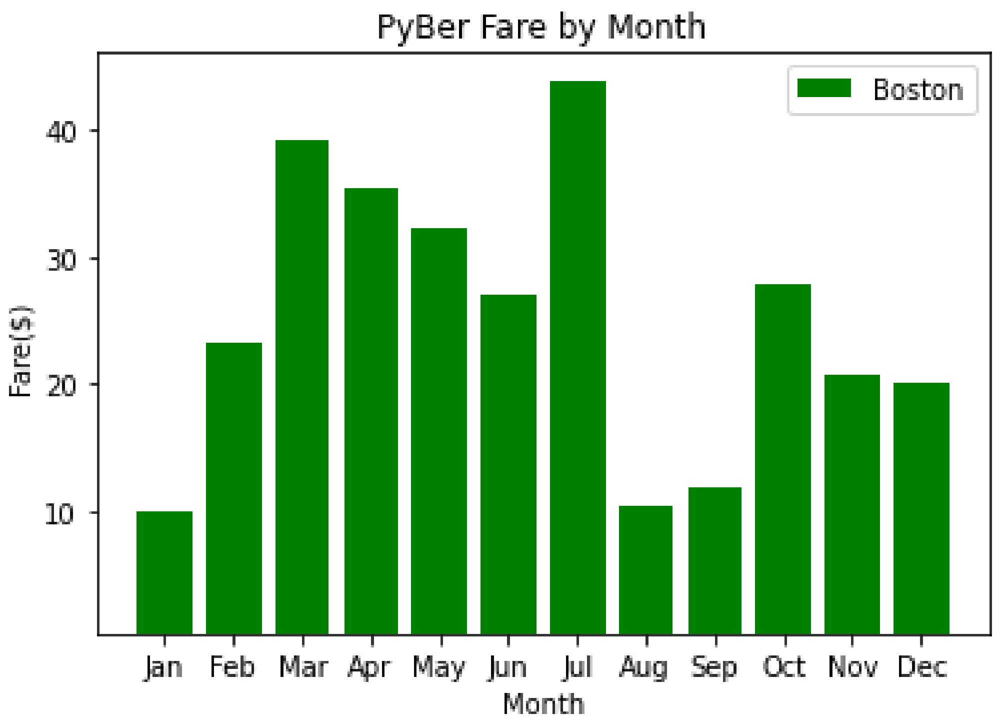


In [37]:

```
# Create the plot
plt.bar(x_axis, y_axis, color = "green", label = "Boston")

plt.xlabel("Month")
plt.ylabel("Fare($)")

plt.ylim(0.45)
plt.title("PyBer Fare by Month")
plt.legend()
```

Out[37]:

```
<matplotlib.legend.Legend at 0x240fb6e74c0>
```

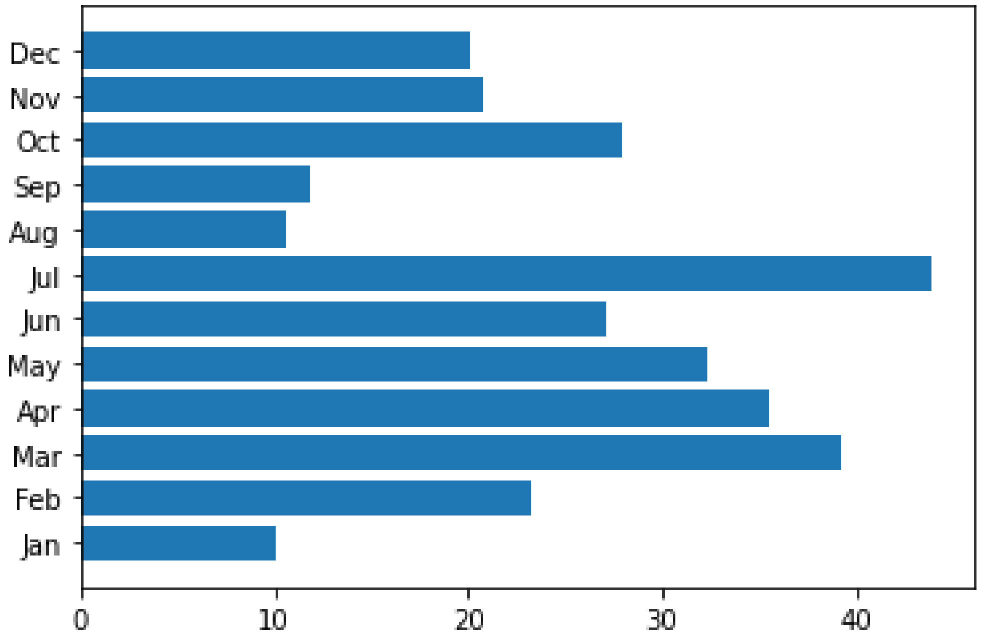


In [38]:

```
# Create the plot
plt.barh(x_axis, y_axis)
```

Out[38]:

```
<BarContainer object of 12 artists>
```

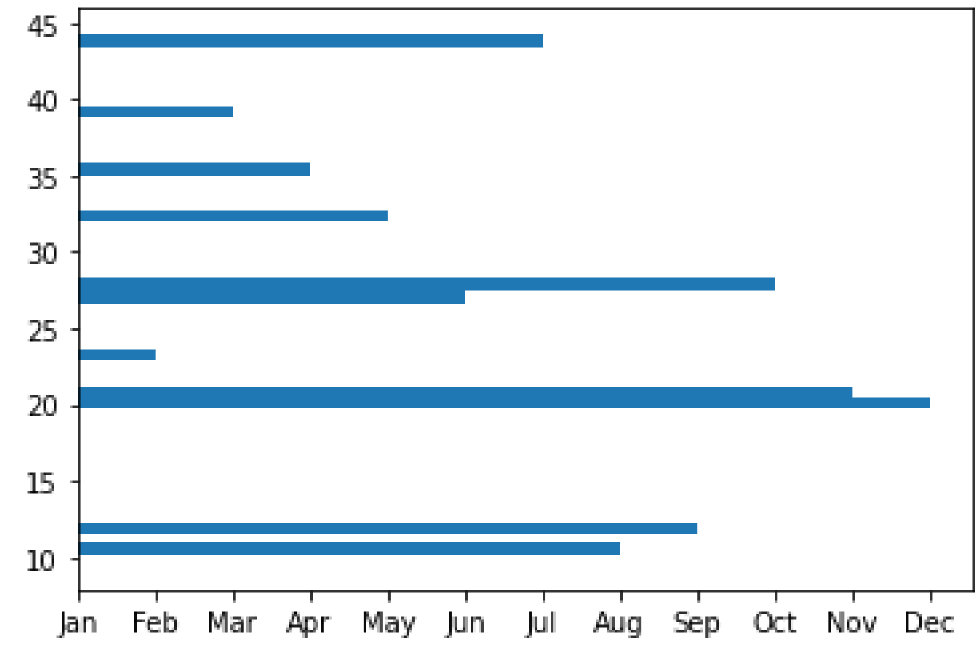


In [39]:

```
plt.barh(y_axis, x_axis)
```

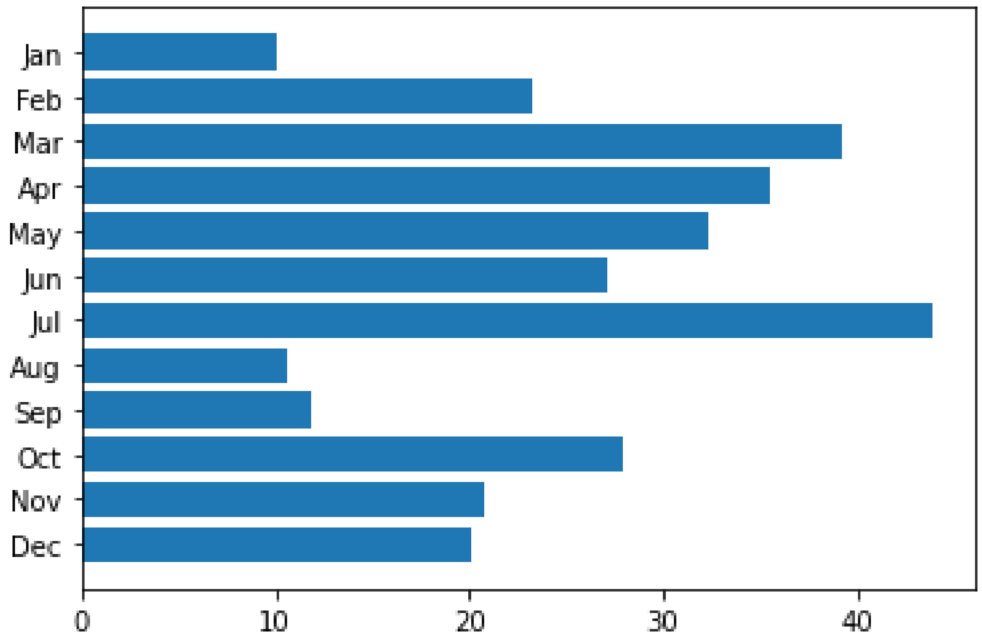
Out[39]:

```
<BarContainer object of 12 artists>
```



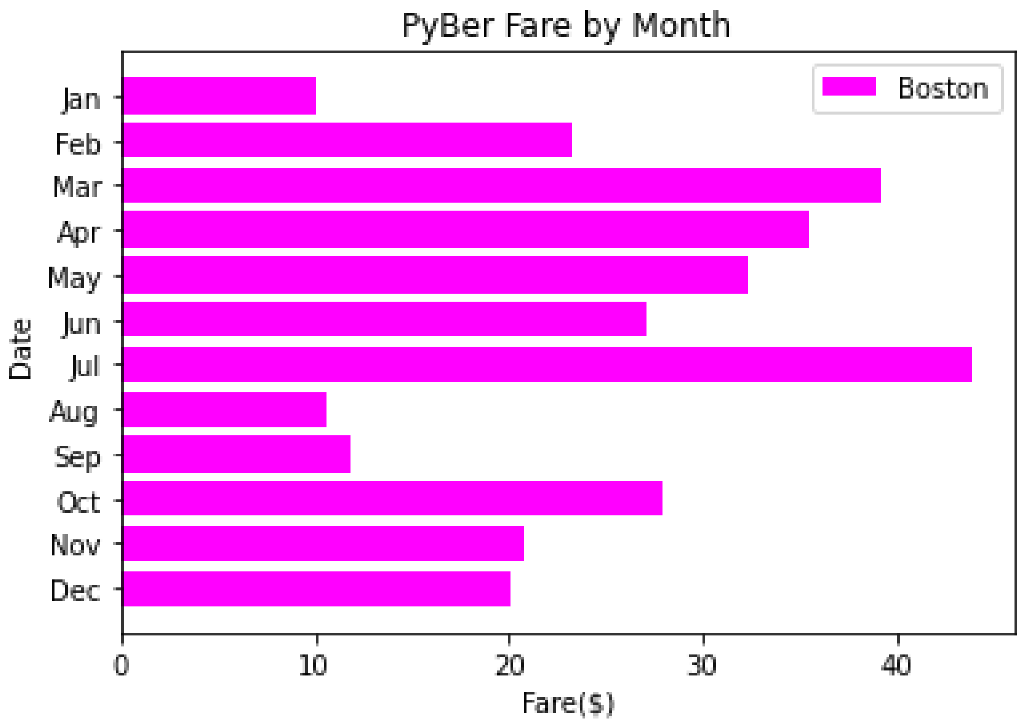
In [40]:

```
# Create the plot.
plt.barh(x_axis, y_axis)
plt.gca().invert_yaxis()
```



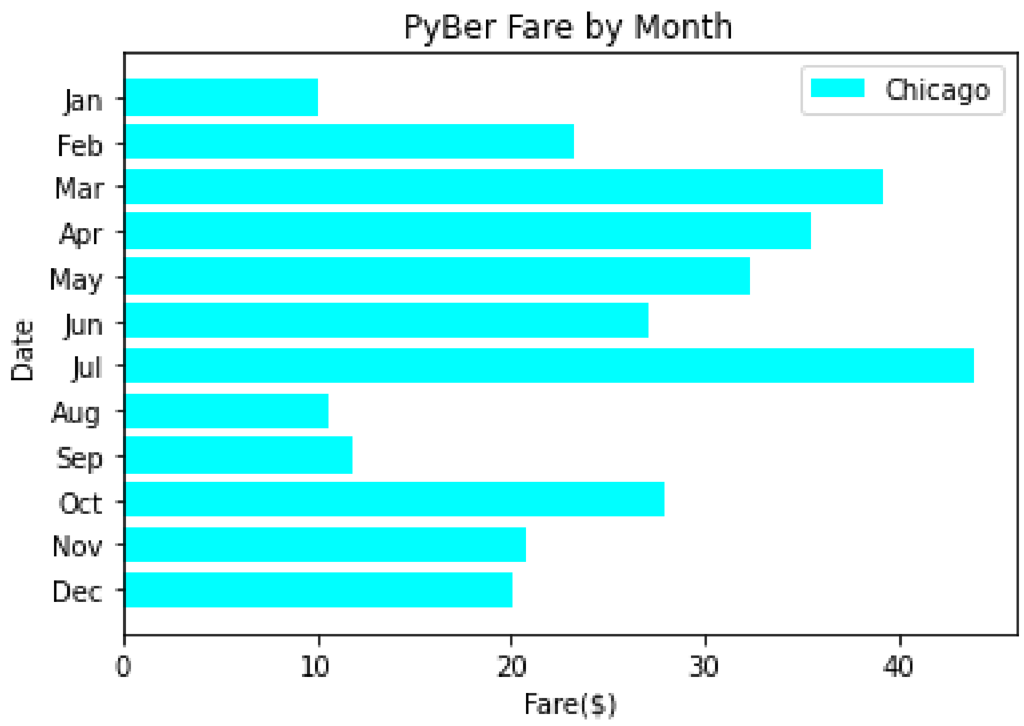
In [42]:

```
plt.barh(x_axis, y_axis, color = "magenta", label = "Boston")
plt.title("PyBer Fare by Month")
plt.xlabel("Fare($)")
plt.ylabel("Date")
plt.legend()
plt.gca().invert_yaxis()
```



In [48]:

```
fig, ax = plt.subplots()

ax.barh(x_axis, y_axis, color = "cyan", label = "Chicago" )
ax.set_xlabel("Fare($)")
ax.set_ylabel("Date")
ax.set_title("PyBer Fare by Month")
ax.legend()
ax.invert_yaxis()
```



In [1]:

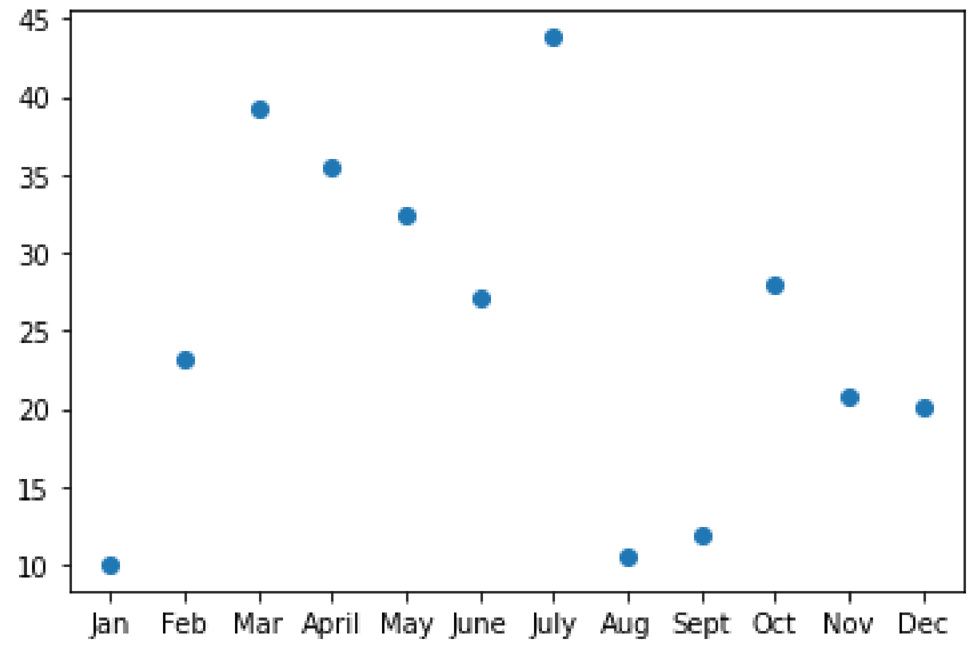
```
# Set the x-axis to a list of strings for each month.
x_axis = ["Jan", "Feb", "Mar", "April", "May", "June", "July", "Aug", "Sept",
"Oct", "Nov", "Dec"]

# Set the y-axis to a list of floats as the total fare in US dollars
accumulated for each month.
y_axis = [10.02, 23.24, 39.20, 35.42, 32.34, 27.04, 43.82, 10.56, 11.85,
27.90, 20.71, 20.09]
```

In [6]:

```
plt.plot(x_axis, y_axis, "o")
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x1f61a6988b0>]
```

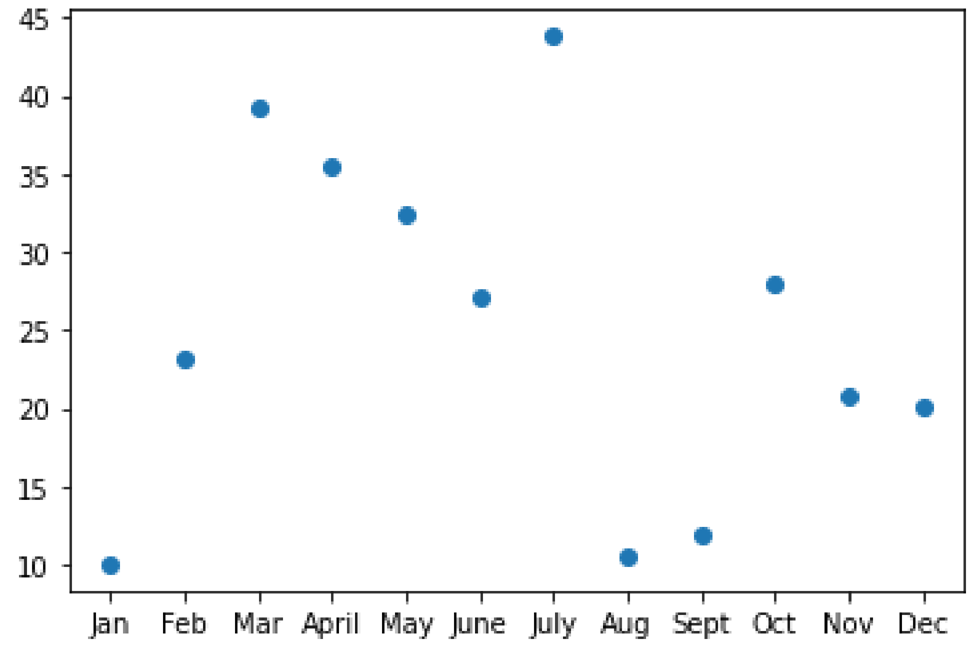


In [7]:

```
plt.scatter(x_axis, y_axis)
```

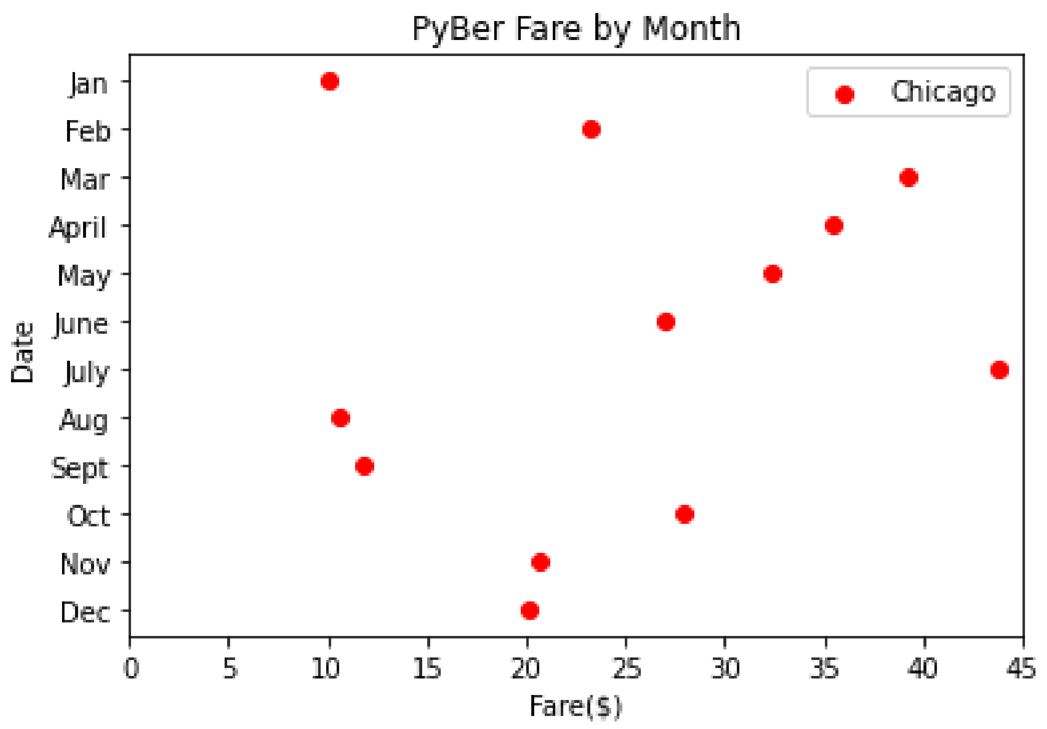
Out[7]:

```
<matplotlib.collections.PathCollection at 0x1f61a707eb0>
```



In [8]:

```
plt.scatter(y_axis, x_axis, c="red", label = "Chicago")
plt.ylabel("Date")
plt.xlabel("Fare($)")
plt.xlim(0,45)
plt.title("PyBer Fare by Month")
plt.legend()
plt.gca().invert_yaxis()
```

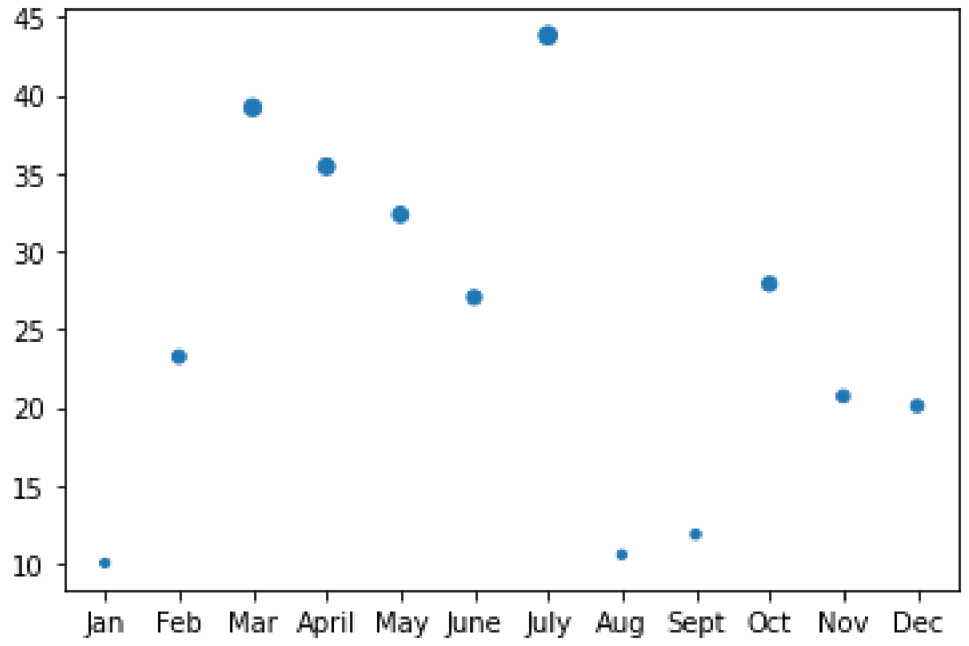


In [10]:

```
plt.scatter(x_axis, y_axis, s=y_axis)
```

Out[10]:

```
<matplotlib.collections.PathCollection at 0x1f61a892520>
```



In [11]:

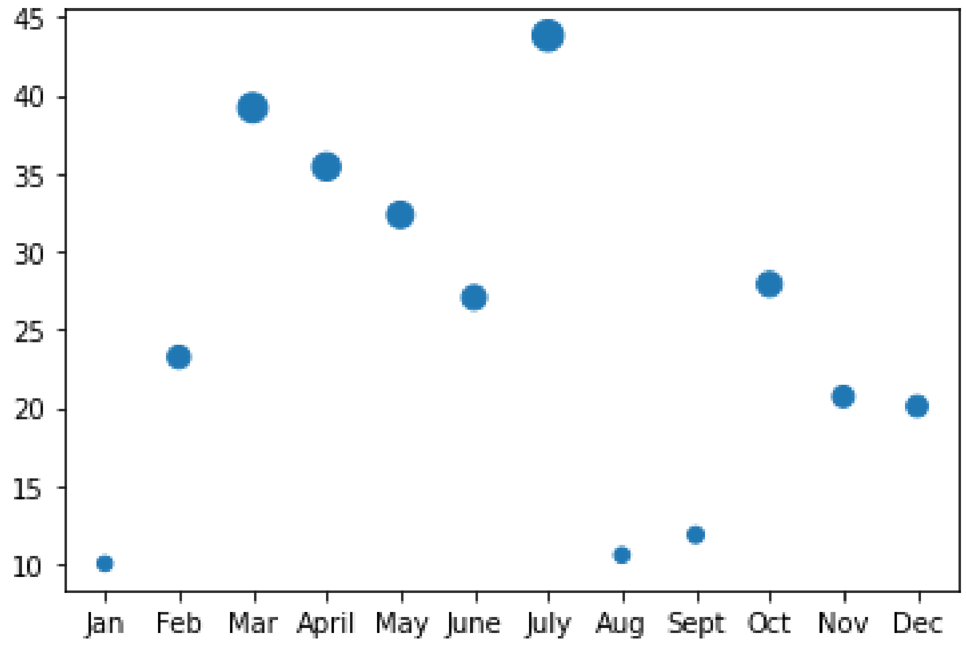
```
y_axis_larger = []
for data in y_axis:
    y_axis_larger.append(data*3)
```

In [12]:

```
plt.scatter(x_axis, y_axis, s=y_axis_larger)
```

Out[12]:

```
<matplotlib.collections.PathCollection at 0x1f61a905c40>
```

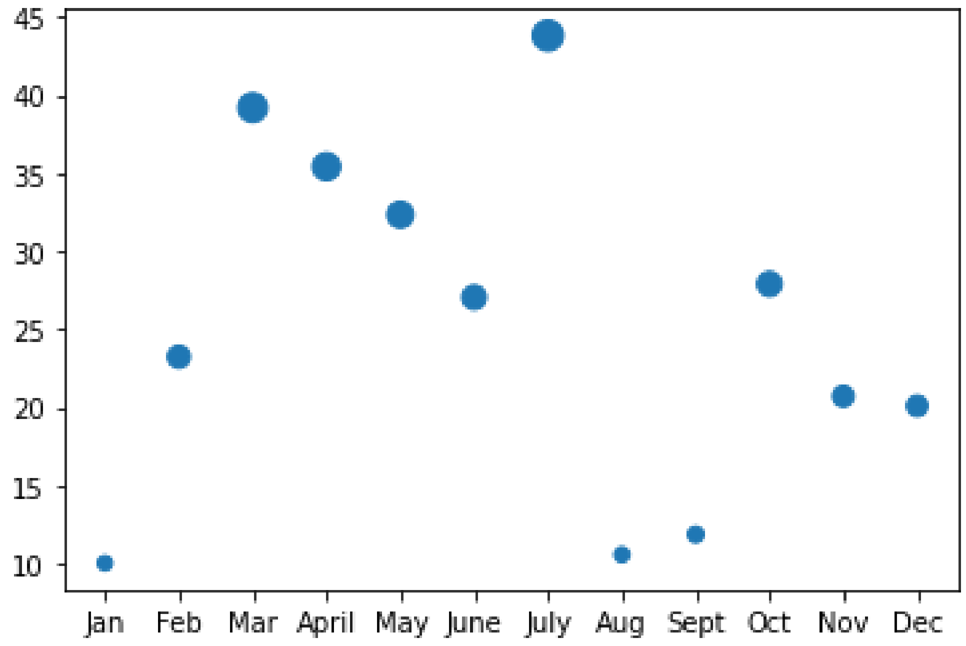


In [14]:

```
plt.scatter(x_axis, y_axis, s = [i * 3 for i in y_axis])
```

Out[14]:

```
<matplotlib.collections.PathCollection at 0x1f61a9843d0>
```

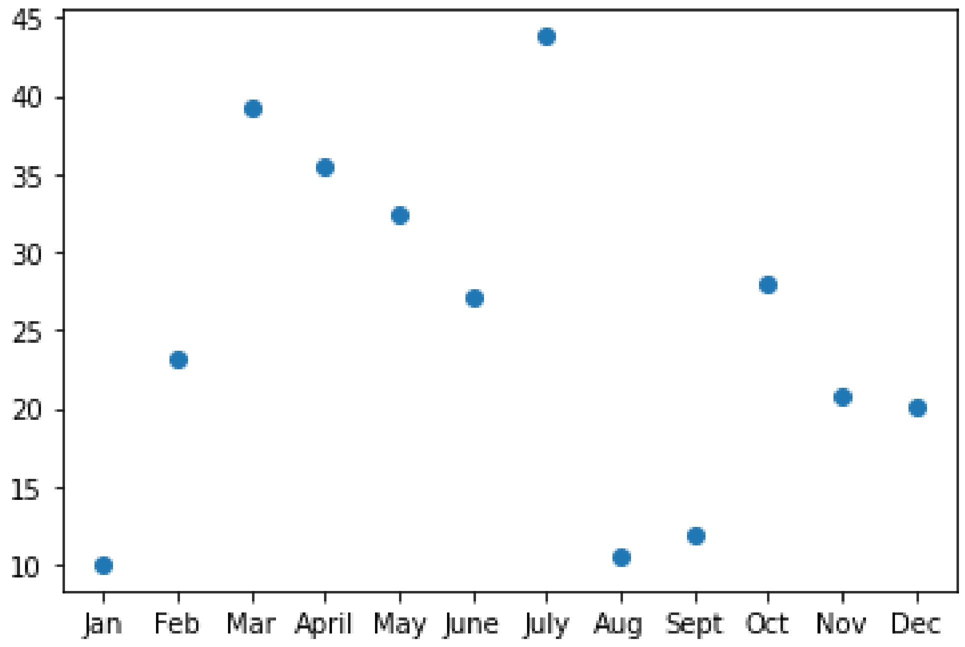


In [15]:

```
#object oriented method

fig, ax = plt.subplots()
ax.scatter(x_axis, y_axis)
```

Out[15]:

```
<matplotlib.collections.PathCollection at 0x1f61a91df70>
```

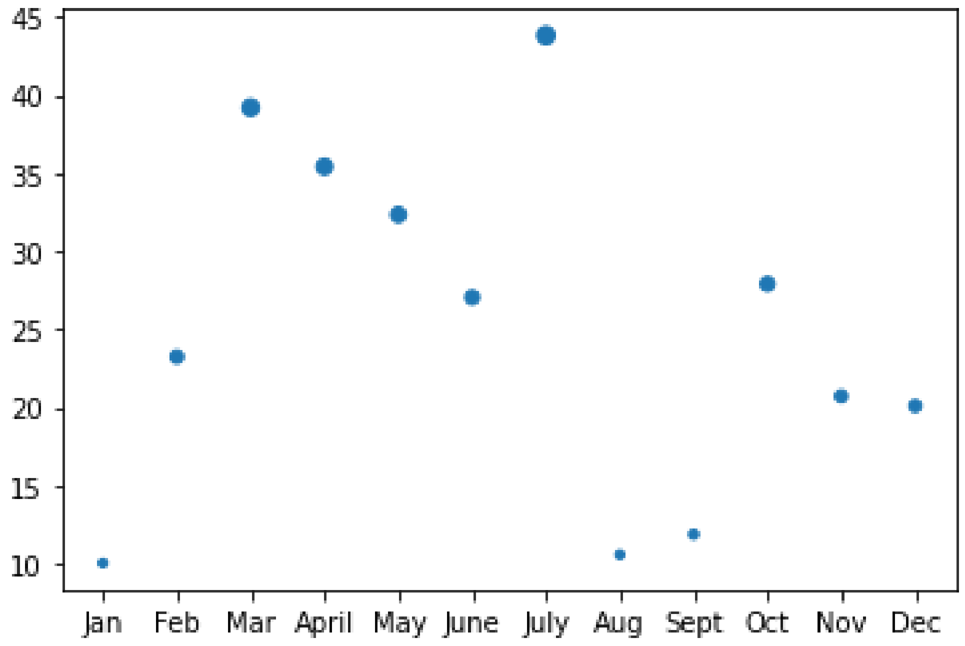


In [17]:

```
fig, ax = plt.subplots()
ax.scatter(x_axis, y_axis, s= y_axis)
```

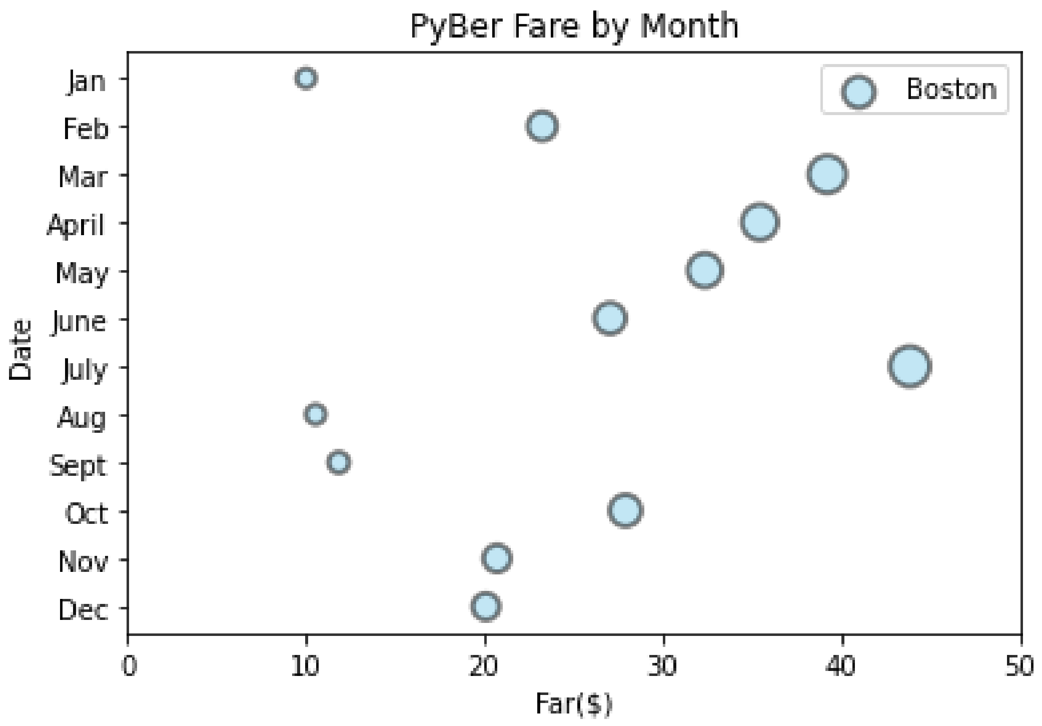
Out[17]:

```
<matplotlib.collections.PathCollection at 0x1f61aa6f7f0>
```



In [30]:

```
fig, ax = plt.subplots()
ax.scatter(y_axis, x_axis,color = "skyblue", alpha = 0.5,edgecolor = "black"
,s = [i * 5 for i in y_axis], linewidth = 2, label = "Boston")
ax.legend()
ax.set_title("PyBer Fare by Month")
ax.set_xlim(0,50)
ax.set_ylabel("Date")
ax.set_xlabel("Far($)")
ax.invert_yaxis()
```

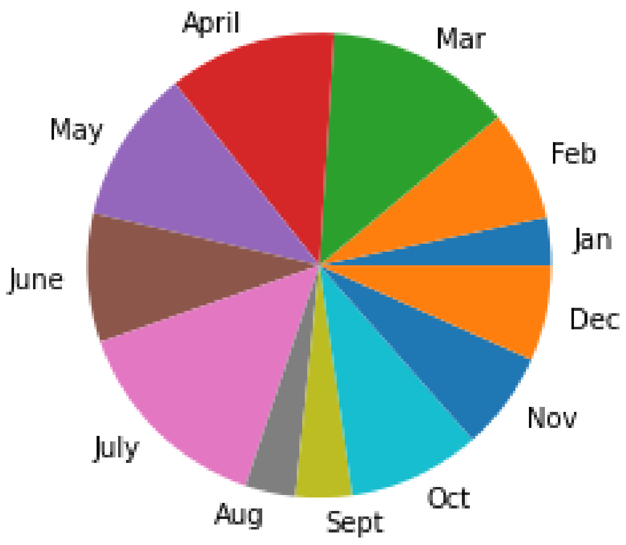


In [32]:

```
plt.pie(y_axis, labels = x_axis)
```

Out[32]:

```
([<matplotlib.patches.Wedge at 0x1f61c1a0310>,
  <matplotlib.patches.Wedge at 0x1f61c1a07f0>,
  <matplotlib.patches.Wedge at 0x1f61bfb1700>,
  <matplotlib.patches.Wedge at 0x1f61c1ad0a0>,
  <matplotlib.patches.Wedge at 0x1f61c1ad580>,
  <matplotlib.patches.Wedge at 0x1f61c1ada60>,
  <matplotlib.patches.Wedge at 0x1f61c1adf40>,
  <matplotlib.patches.Wedge at 0x1f61c1ba460>,
  <matplotlib.patches.Wedge at 0x1f61c1ba940>,
  <matplotlib.patches.Wedge at 0x1f61c1bae20>,
  <matplotlib.patches.Wedge at 0x1f61c1a02e0>,
  <matplotlib.patches.Wedge at 0x1f61c1c9820>],
 [Text(1.0940372721655667, 0.11437852557419124, 'Jan'),
  Text(0.9905193092273052, 0.4784051609753622, 'Feb'),
  Text(0.4998632656180861, 0.9798656620606842, 'Mar'),
  Text(-0.3293082697128627, 1.0495504101750999, 'April'),
  Text(-0.9306200792045569, 0.5864693241605263, 'May'),
  Text(-1.0983369533258995, -0.06046434452452758, 'June'),
  Text(-0.7729302540020312, -0.782674148319948, 'July'),
  Text(-0.2333875250754619, -1.0749559354406817, 'Aug'),
  Text(0.021097559209412746, -1.099797660024518, 'Sept'),
  Text(0.4609812999807629, -0.9987473359504124, 'Oct'),
  Text(0.8868706338501641, -0.6507384104340302, 'Nov'),
  Text(1.0760953148482877, -0.22807646384834376, 'Dec')])
```

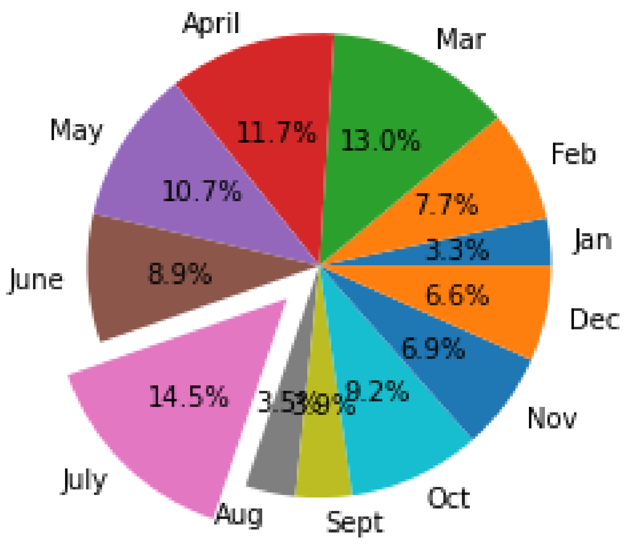


In [38]:

```
explode_values =(0,0,0,0,0,0,0.2,0,0,0,0,0)
plt.pie(y_axis, labels = x_axis, explode= explode_values, autopct = "%.1f%%")
```

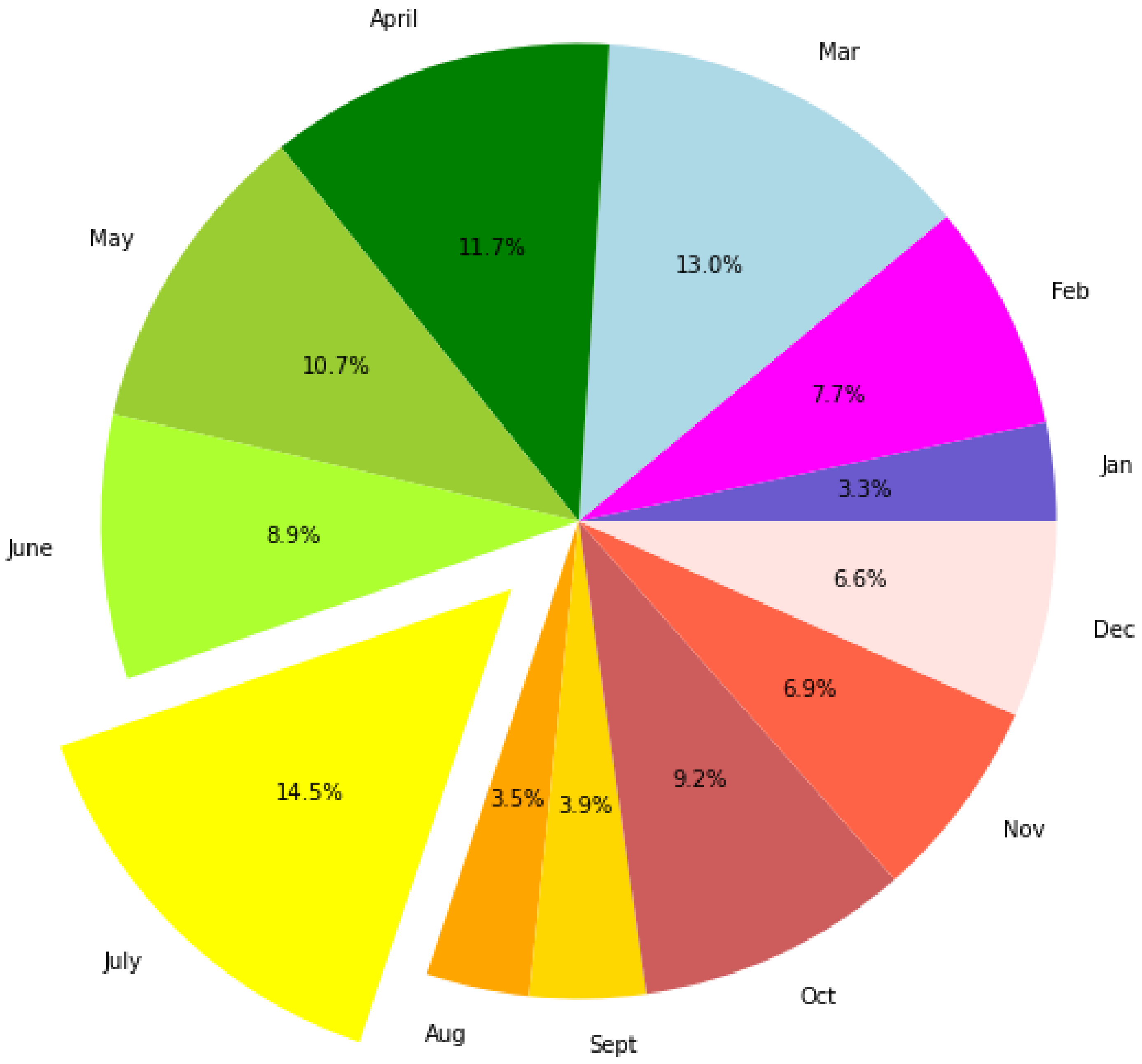
Out[38]:

```
([<matplotlib.patches.Wedge at 0x1f61bc07040>,
  <matplotlib.patches.Wedge at 0x1f61bcb7250>,
  <matplotlib.patches.Wedge at 0x1f61bcb7fd0>,
  <matplotlib.patches.Wedge at 0x1f61bcc4550>,
  <matplotlib.patches.Wedge at 0x1f61bcc4160>,
  <matplotlib.patches.Wedge at 0x1f61bcbf9a0>,
  <matplotlib.patches.Wedge at 0x1f61bcbf520>,
  <matplotlib.patches.Wedge at 0x1f61bc9ba60>,
  <matplotlib.patches.Wedge at 0x1f61bc9b610>,
  <matplotlib.patches.Wedge at 0x1f61bc00ac0>,
  <matplotlib.patches.Wedge at 0x1f61bc070d0>,
  <matplotlib.patches.Wedge at 0x1f61bc95c40>],
 [Text(1.0940372721655667, 0.11437852557419124, 'Jan'),
  Text(0.9905193092273052, 0.4784051609753622, 'Feb'),
  Text(0.4998632656180861, 0.9798656620606842, 'Mar'),
  Text(-0.3293082697128627, 1.0495504101750999, 'April'),
  Text(-0.9306200792045569, 0.5864693241605263, 'May'),
  Text(-1.0983369533258995, -0.06046434452452758, 'June'),
  Text(-0.913463027456946, -0.9249785389235748, 'July'),
  Text(-0.2333875250754619, -1.0749559354406817, 'Aug'),
  Text(0.021097559209412746, -1.099797660024518, 'Sept'),
  Text(0.4609812999807629, -0.9987473359504124, 'Oct'),
  Text(0.8868706338501641, -0.6507384104340302, 'Nov'),
  Text(1.0760953148482877, -0.22807646384834376, 'Dec')],
 [Text(0.5967476029993999, 0.06238828667683158, '3.3%'),
  Text(0.5402832595785301, 0.2609482696229248, '7.7%'),
  Text(0.27265269033713785, 0.5344721793058277, '13.0%'),
  Text(-0.17962269257065236, 0.5724820419136908, '11.7%'),
  Text(-0.5076109522933946, 0.3198923586330143, '10.7%'),
  Text(-0.5990928836323088, -0.03298055155883322, '8.9%'),
  Text(-0.562131093819659, -0.5692175624145076, '14.5%'),
  Text(-0.12730228640479738, -0.5863396011494626, '3.5%'),
  Text(0.011507759568770587, -0.5998896327406462, '3.9%'),
  Text(0.2514443454440525, -0.5447712741547703, '9.2%'),
  Text(0.4837476184637258, -0.35494822387310737, '6.9%'),
  Text(0.5869610808263387, -0.12440534391727841, '6.6%')])
```



In [40]:

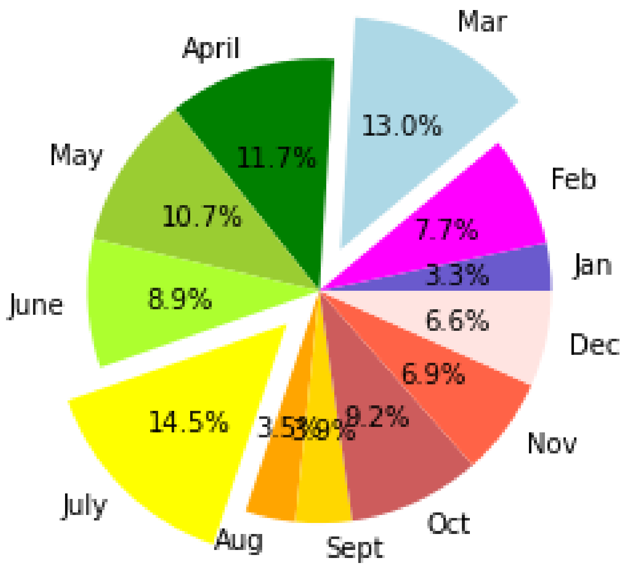
```
# Assign 12 colors, one for each month.
colors = ["slateblue", "magenta", "lightblue", "green", "yellowgreen",
"greenyellow", "yellow", "orange", "gold", "indianred", "tomato",
"mistyrose"]
explode_values = (0, 0, 0, 0, 0, 0, 0.2, 0, 0, 0, 0, 0)
plt.subplots(figsize=(10,10))
plt.pie(y_axis, labels = x_axis, explode= explode_values, colors = colors,
autopct = "%.1f%%")
plt.show()
```



In [52]:

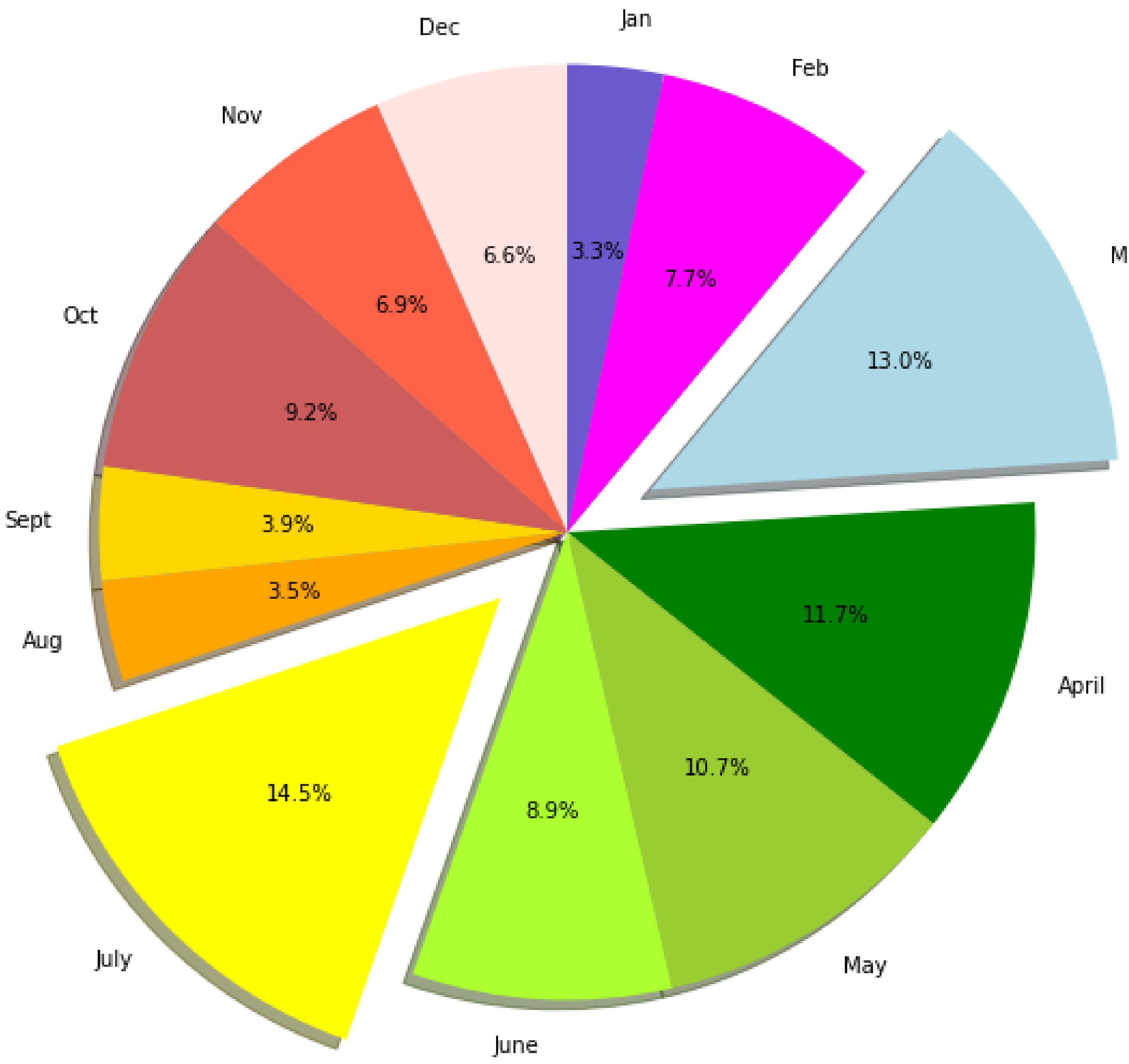
```
# object oriented way
# Set the x-axis to a list of strings for each month.
x_axis = ["Jan", "Feb", "Mar", "April", "May", "June", "July", "Aug", "Sept",
"Oct", "Nov", "Dec"]
```

```
# Set the y-axis to a list of floats as the total fare in US dollars
accumulated for each month.
y_axis = [10.02, 23.24, 39.20, 35.42, 32.34, 27.04, 43.82, 10.56, 11.85,
27.90, 20.71, 20.09]
colors = ["slateblue", "magenta", "lightblue", "green", "yellowgreen",
"greenyellow", "yellow", "orange", "gold", "indianred", "tomato",
"mistyrose"]
explode_values = (0, 0, 0.2, 0, 0, 0, 0.2, 0, 0, 0, 0, 0)
fig, ax = plt.subplots()
ax.pie(y_axis, labels = x_axis, explode= explode_values, colors = colors,
autopct = "%.1f%%")

plt.show()
```



In [64]:

```
# object oriented way
# Set the x-axis to a list of strings for each month.
x_axis = ["Jan", "Feb", "Mar", "April", "May", "June", "July", "Aug", "Sept",
"Oct", "Nov", "Dec"]

# Set the y-axis to a list of floats as the total fare in US dollars
accumulated for each month.
y_axis = [10.02, 23.24, 39.20, 35.42, 32.34, 27.04, 43.82, 10.56, 11.85,
27.90, 20.71, 20.09]
colors = ["slateblue", "magenta", "lightblue", "green", "yellowgreen",
"greenyellow", "yellow", "orange", "gold", "indianred", "tomato",
"mistyrose"]
explode_values = (0, 0, 0.2, 0, 0, 0, 0.2, 0, 0, 0, 0, 0)
fig, ax = plt.subplots(figsize=(10, 10))
ax.pie(y_axis, labels = x_axis, explode= explode_values, colors = colors,
autopct = "%.1f%%",shadow = True, counterclock=False, startangle=90)

plt.show()
```



In [1]:

```
%matplotlib inline
```

In [2]:

```
# Import dependencies.
import matplotlib.pyplot as plt
import statistics
```

In [3]:

```
# Set the x-axis to a list of strings for each month.
x_axis = ["Jan", "Feb", "Mar", "April", "May", "June", "July", "Aug", "Sept",
"Oct", "Nov", "Dec"]

# Set the y-axis to a list of floats as the total fare in US dollars
accumulated for each month.
y_axis = [10.02, 23.24, 39.20, 35.42, 32.34, 27.04, 43.82, 10.56, 11.85,
27.90, 20.71, 20.09]
```

In [5]:

```
# Get the standard deviation of the values in the y-axis.
stdev = statistics.stdev(y_axis)
stdev
```

Out[5]:

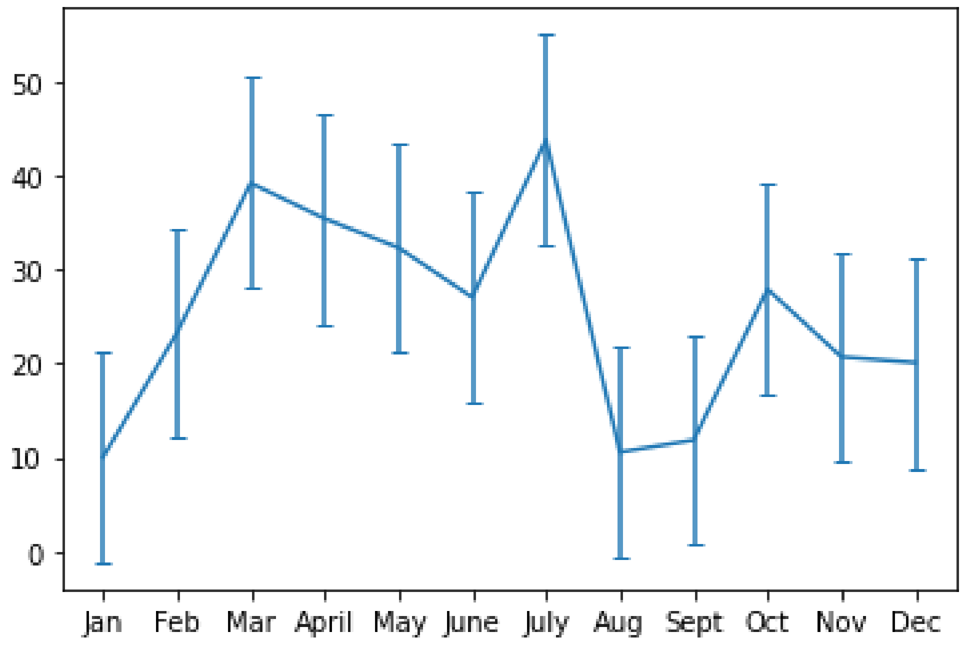
```
11.208367917035753
```

In [70]:

```
plt.errorbar(x_axis, y_axis, yerr=stdev, capsize=3)
```

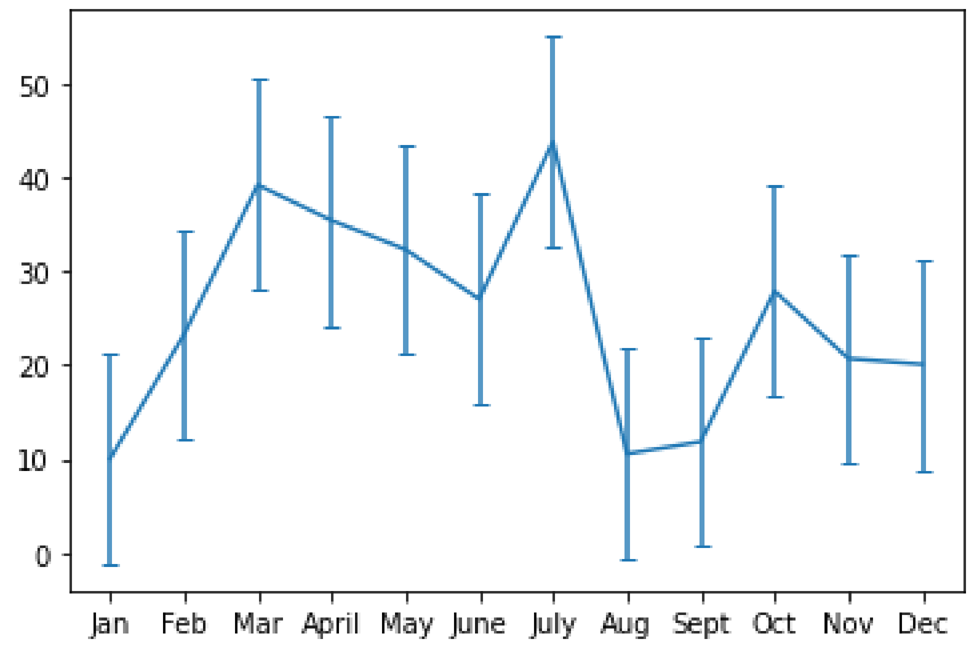
Out[70]:

```
<ErrorbarContainer object of 3 artists>
```



In [71]:

```
fig, ax = plt.subplots()
ax.errorbar(x_axis, y_axis, yerr=stdev, capsize=3)
plt.show()
```

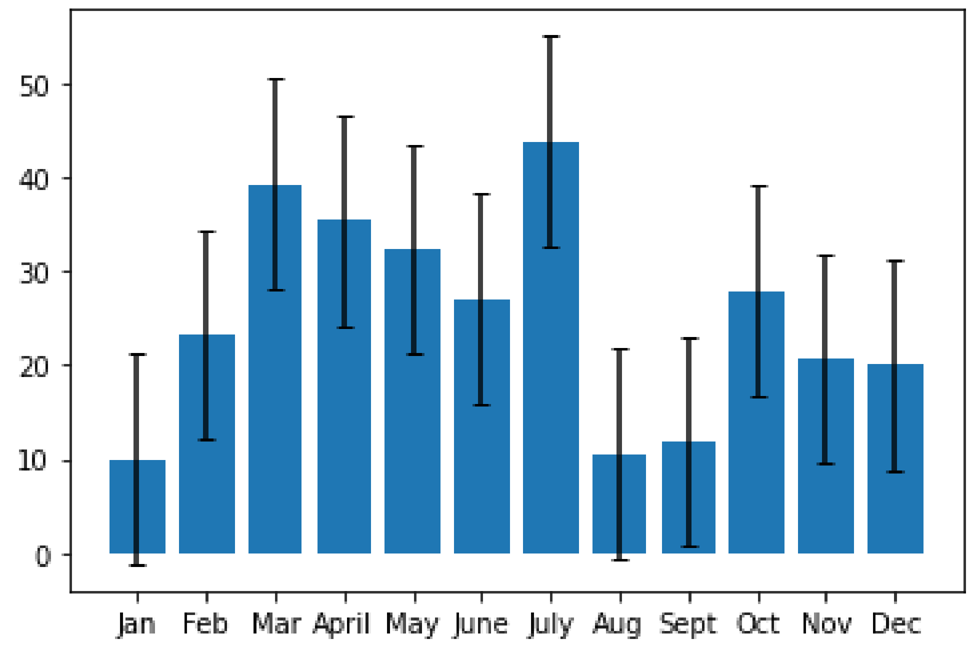


In [72]:

```
plt.bar(x_axis, y_axis, yerr=stdev, capsize=3)
```

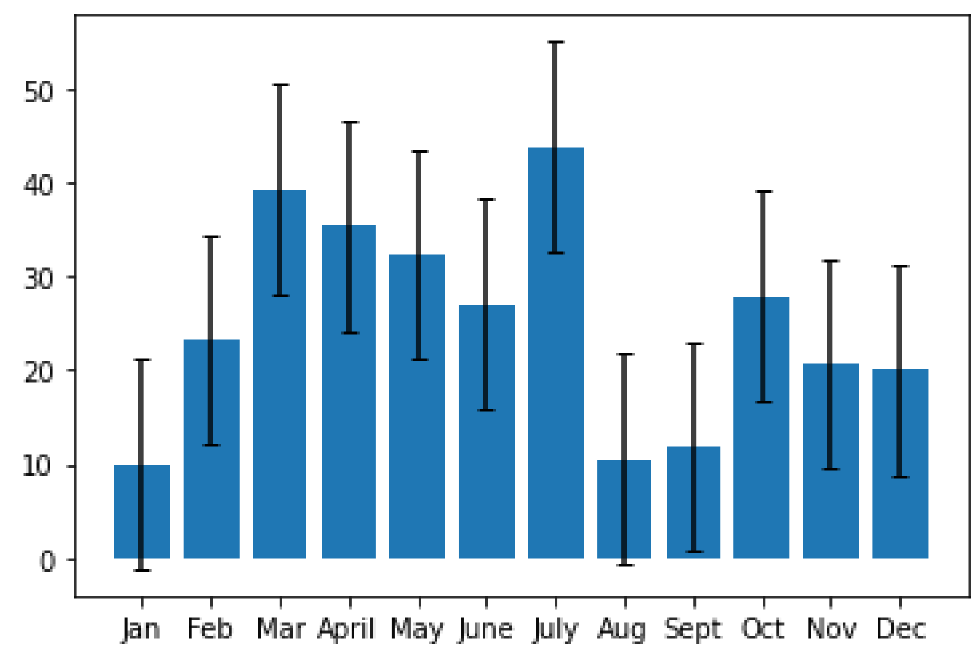
Out[72]:

```
<BarContainer object of 12 artists>
```



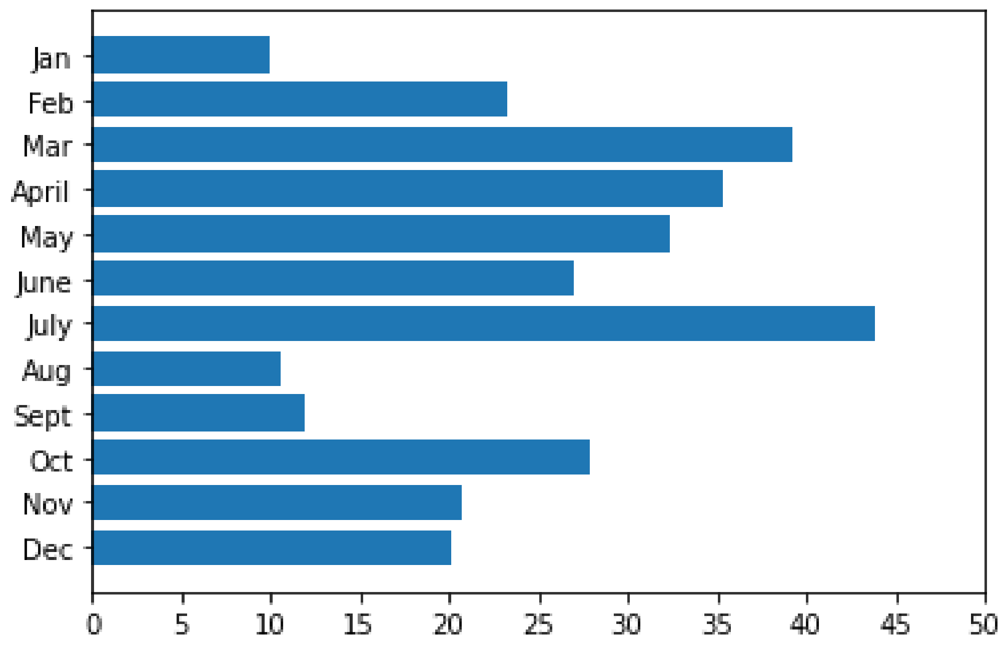
In [73]:

```
fig, ax = plt.subplots()
ax.bar(x_axis, y_axis, yerr=stdev, capsize=3)
plt.show()
```



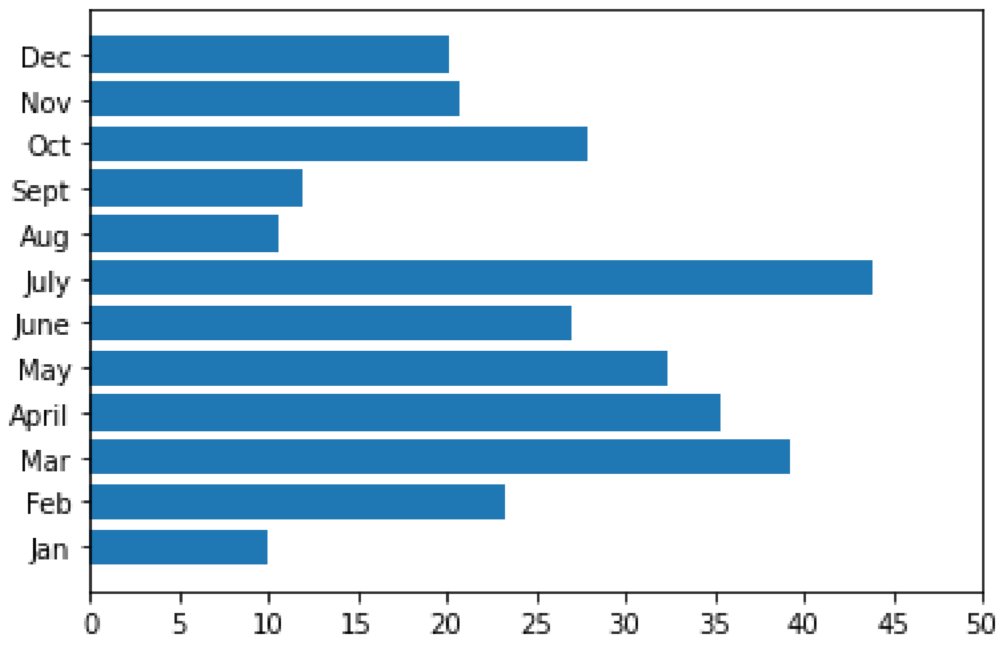
In [74]:

```
import numpy as np
plt.barh(x_axis, y_axis)
plt.xticks(np.arange(0, 51, step=5.0))
plt.gca().invert_yaxis()
```



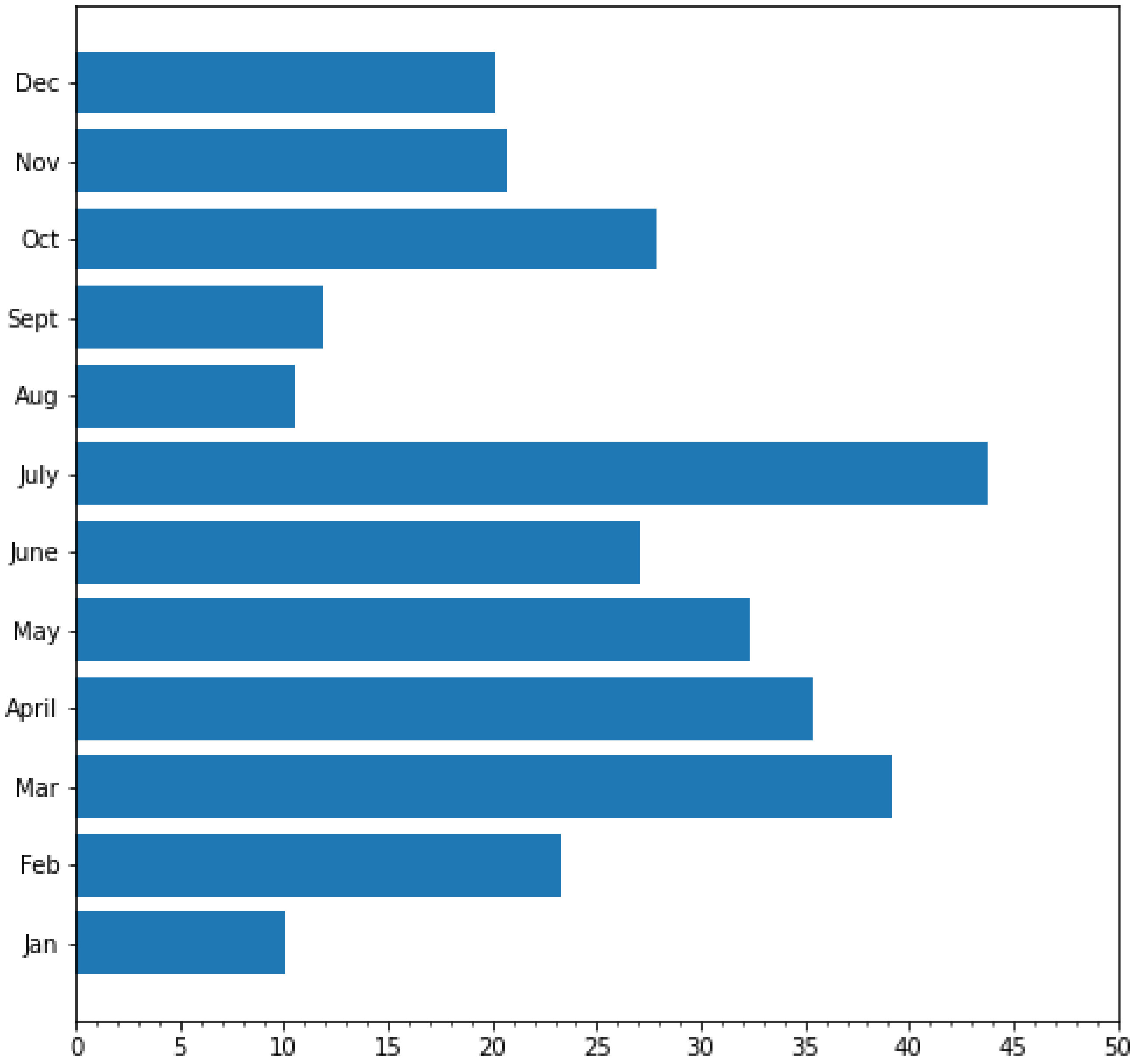
In [75]:

```
fig, ax = plt.subplots()
ax.barh(x_axis, y_axis)
ax.set_xticks(np.arange(0, 51, step=5.0))
plt.show()
```



In [76]:

```
from matplotlib.ticker import MultipleLocator
# Increase the size of the plot figure.
fig, ax = plt.subplots(figsize=(8, 8))
ax.barh(x_axis, y_axis)
ax.set_xticks(np.arange(0, 51, step=5.0))

# Create minor ticks at an increment of 1.
ax.xaxis.set_minor_locator(MultipleLocator(1))
plt.show()
```


Dec
Nov
Oct
Sept
Aug
July
June
May
April
Mar
Feb
Jan
0
5
10
15
20
25
30
35
40
45
50

In [ ]: