CLARUSWAY
WAY TO REINVENT YOURSELF

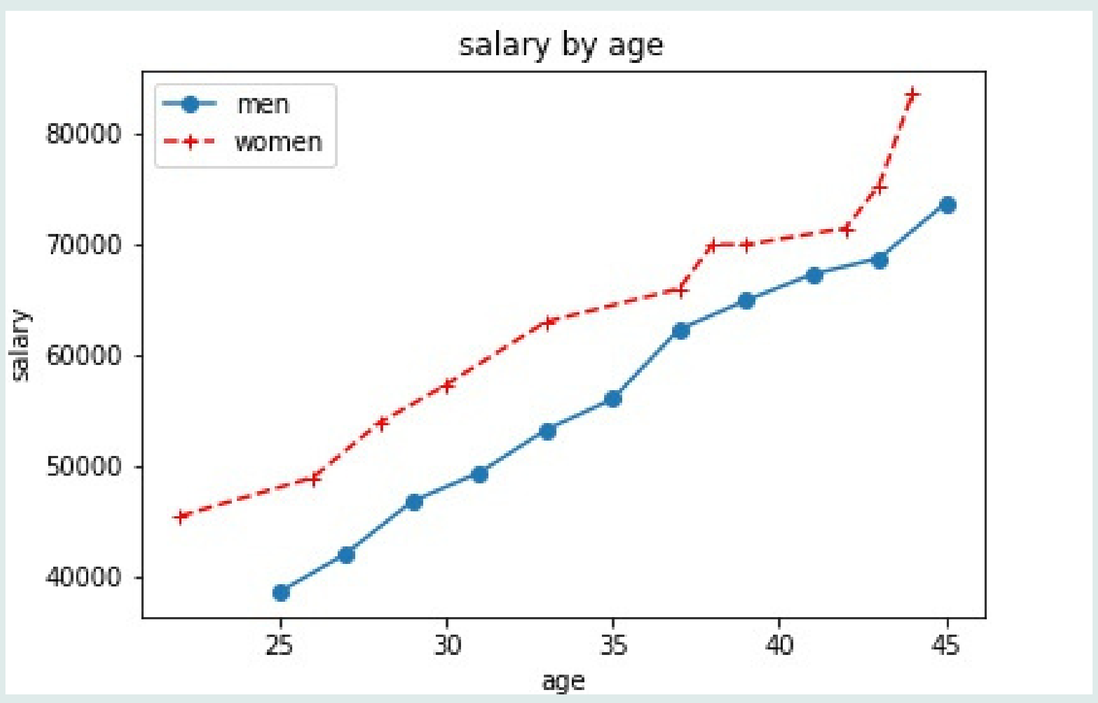
Matplotlib Cheat Sheet

Import Libraries and Create a DataFrame

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
men_age = [25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45]
men_salary = [38496, 42000, 46752, 49320, 53200,
        56000, 62316, 64928, 67317, 68748, 73752]
women_salary = [45372, 48876, 53850, 57287, 63016,
          65998, 70003, 70000, 71496, 75370, 83640]
women_age =[22, 26, 28, 30, 33, 37, 38, 39, 42, 43, 44]
group=["G1", "G2","G3","G4","G5","G6","G7","G8","G9","G10","G11"]
df=pd.DataFrame({"men_age":men_age, "men_salary":men_salary,
"women_age":women_age, "women_salary":women_salary, "group":group})
```

Line Plot (Functional Method)

```
plt.plot(df.men_age, df.men_salary, marker="o", label="men") # label is required
for legend
plt.plot(df.women_age, df.women_salary, marker="+", ls="--", color="r",
label="women")
plt.xlabel("age")
plt.ylabel("salary")
plt.title("salary by age")
plt.legend()
```

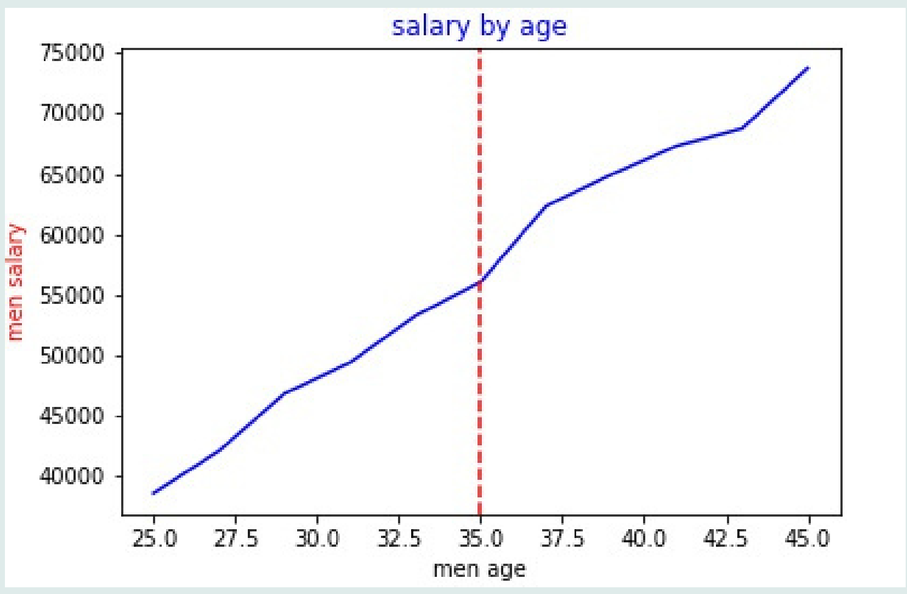


Subplots (Fuctional Method)

```
plt.figure(figsize=(6, 6))
plt.subplot(2,1,1) # nrow=2, ncols=1, index=1
plt.plot(df.men_age, df.men_salary, label="men", color="b")
plt.title("salary by age")
plt.ylabel("salary")
plt.xlabel("age")
plt.legend(loc=0) # loc=0 determines best location for the legend
plt.subplot(2,1,2)
plt.plot(df.women_age, df.women_salary, label="women", color="r")
plt.xlabel("age")
plt.ylabel("salary")
plt.legend(loc=4) # loc=4 puts the legend to bottom right
plt.tight_layout() # to fit the subplots.
```


Line Plot (Object Oriented Method)

```
fig, ax= plt.subplots()
ax.plot(df.men_age, df.men_salary ,"b")
ax.set_xlabel("men age")
ax.set_ylabel("men salary", color="r")
ax.set_title("salary by age", color="b")
ax.axvline(x=35, color="red", ls="--") # creating a vertical line for a given "x"
value
```

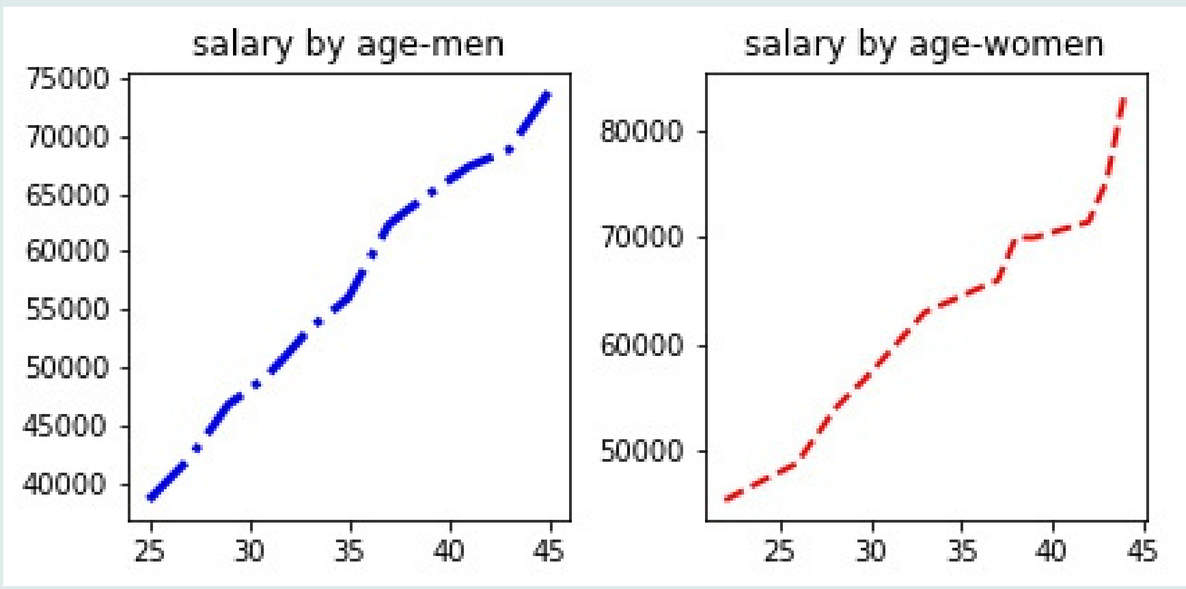


Setting x_lim and y_lim

```
fig, ax= plt.subplots()
ax.plot(df.men_age, df.men_salary ,"b")
ax.set_xlabel("men age")
ax.set_ylabel("men salary", color="r")
ax.set_title("salary by age", color="b")
ax.set_xlim([30,40]) # focusing on the given x values
ax.set_ylim([50000,65000]) # focusing on the given y values
```

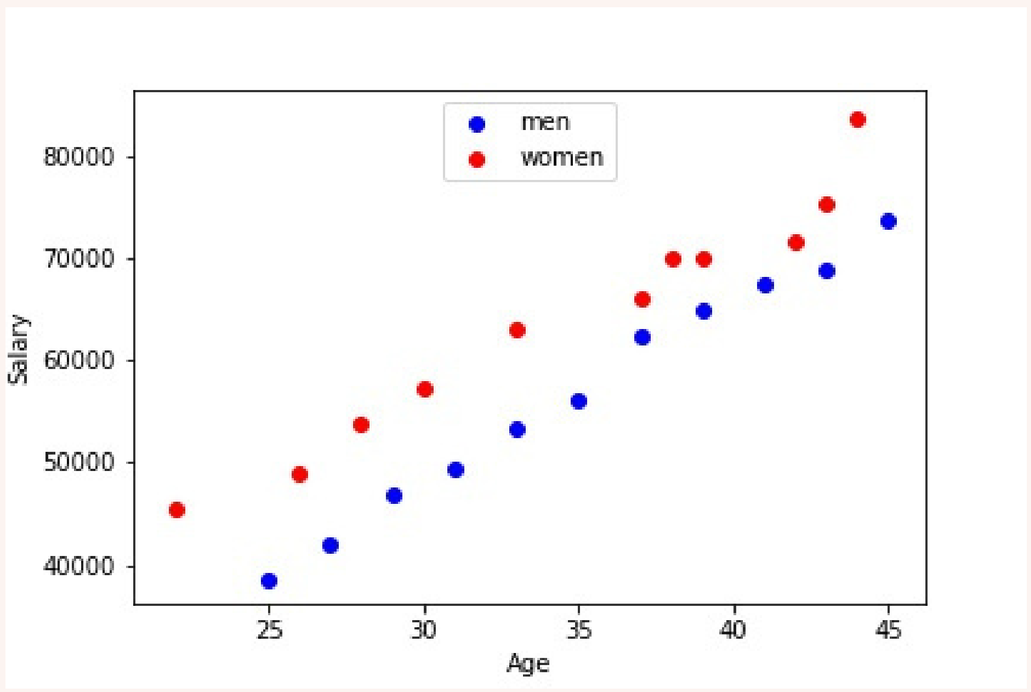
Subplots (Object Oriented Method)

```
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(6,3))
ax[0].plot(df.men_age, df.men_salary ,color="b", lw="3", ls="-.")
ax[0].set_title("salary by age-men")
ax[1].plot(df.women_age, df.women_salary ,color="r", lw="2", ls="dashed")
ax[1].set_title("salary by age-women")
plt.tight_layout()
```



Scatter Plot

```
fig,ax=plt.subplots()
ax.scatter(df.men_age, df.men_salary, color="blue", label="men")
ax.scatter(df.women_age, df.women_salary, color="red", label="women")
ax.set_xlabel("Age")
ax.set_ylabel("Salary")
ax.legend(loc=9) # legend at the top center
```

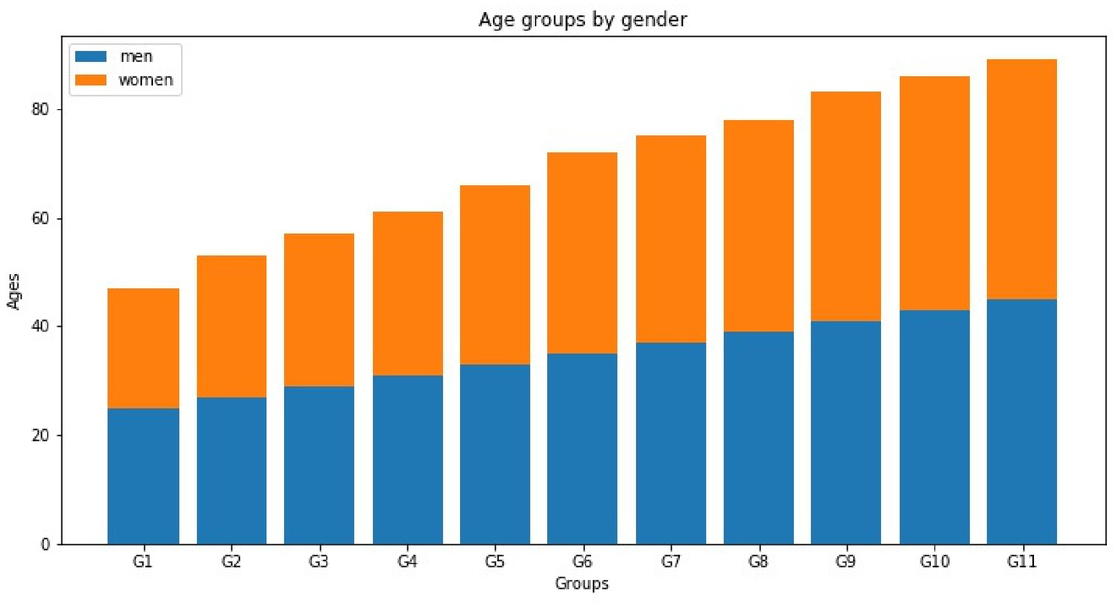


Bar Plot

```
fig,ax=plt.subplots(figsize=(8,5))
ax.bar(df.group, df.men_salary)
ax.set_xlabel("Age Groups")
ax.set_ylabel("Salary")
ax.set_title("Salary by Age Groups")
```

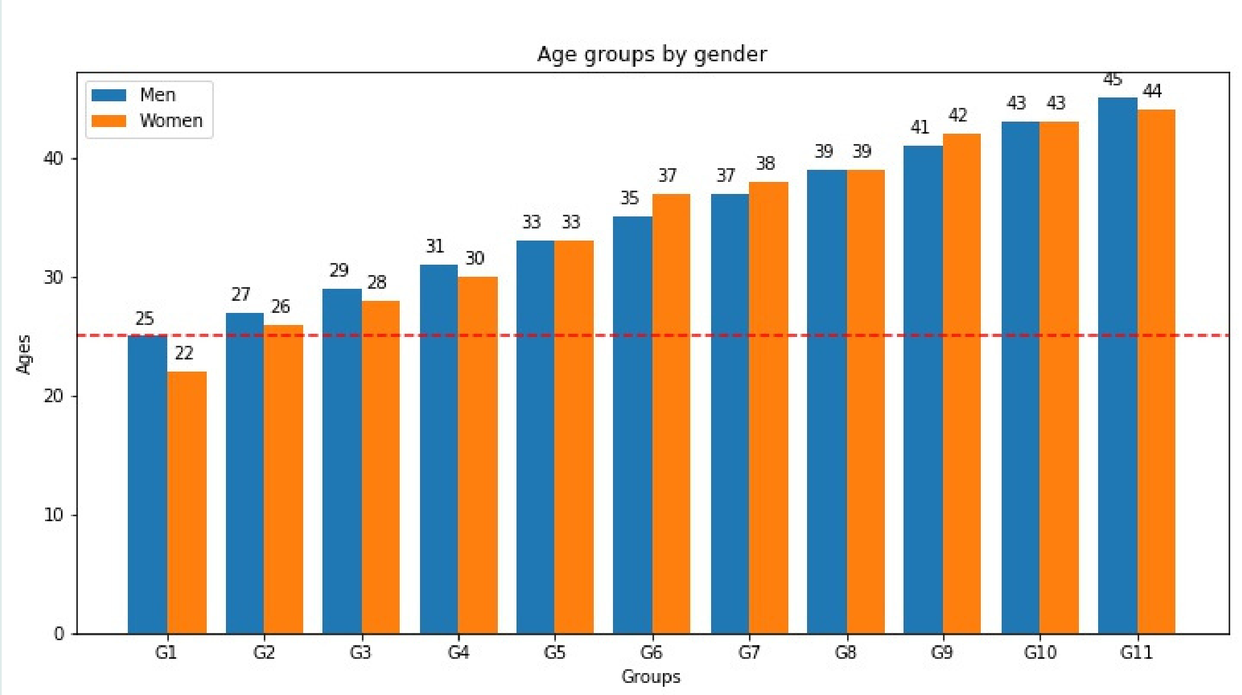
Bar Plot (Stacked)

```
fig,ax=plt.subplots(figsize=(12,6))
ax.bar(df.group, df.men_age, label="men")
ax.bar(df.group, df.women_age, bottom=df.men_age, label="women")
ax.set_xlabel("Groups")
ax.set_ylabel("Ages")
ax.set_title("Age groups by gender")
ax.legend()
```



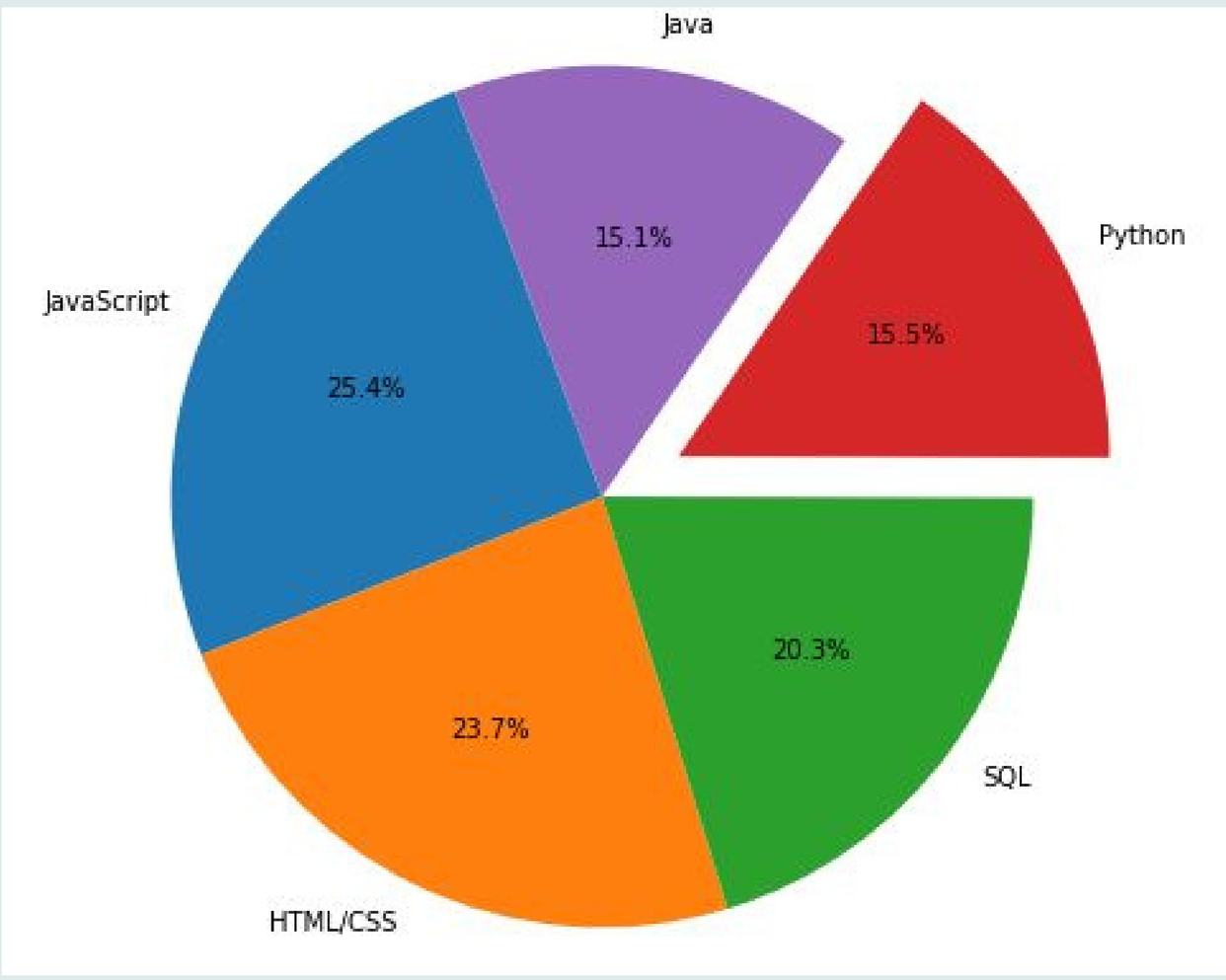
Bar Plot (Not Stacked)

```
x=np.arange(len(df.group))
width=0.4
fig,ax=plt.subplots(figsize=(12,6))
ax.bar(x - width/2, df.men_age, width, label='Men')
ax.bar(x + width/2, df.women_age, width, label='Women')
ax.set_xticks(x)
ax.set_xlabel("Groups")
ax.set_ylabel("Ages")
ax.set_title("Age groups by gender")
ax.axhline(y=25, color="red", ls="--") # adds a horizontal line for a given "y"
value
ax.legend()
ax.set_xticklabels(df.group) #labelling the bars
# to annotate numbers
for p in ax.patches:
    ax.annotate((p.get_height()), (p.get_x()+0.05, p.get_height()+1))
```



Pie Chart

```
slices = [59000, 55000, 47000, 36000, 35000]
langs = ['JavaScript', 'HTML/CSS', 'SQL', 'Python', 'Java']
# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = langs
sizes = slices
explode = (0, 0, 0, 0.2, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')
fig, ax = plt.subplots(figsize=(7,7))
ax.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=False, startangle=110)
ax.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

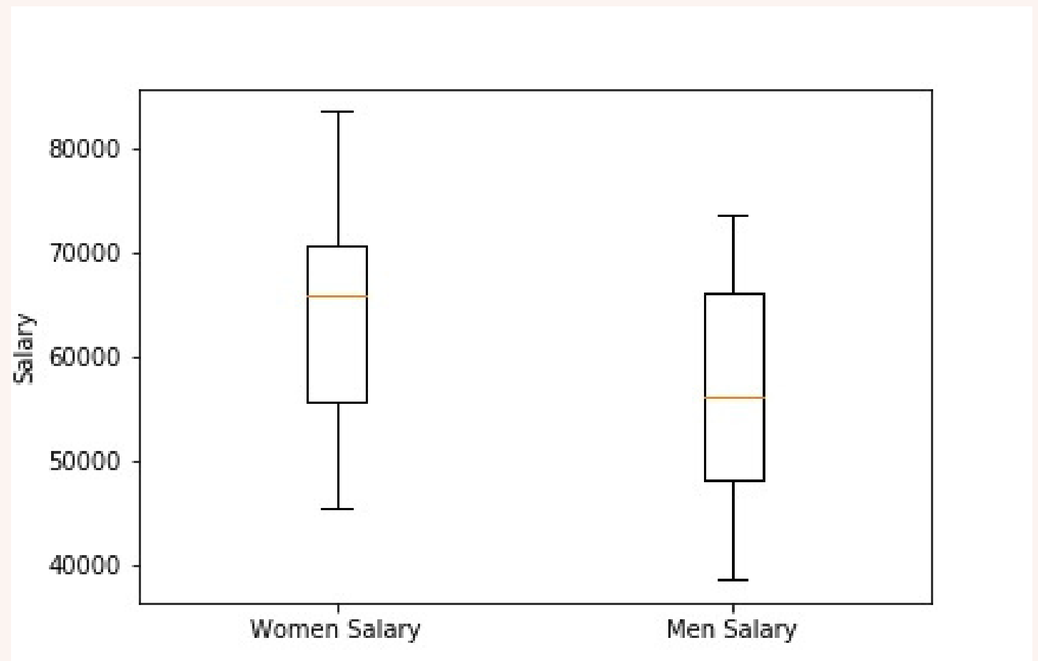


Box Plot (With Subplots)

```
fig,ax=plt.subplots(1,2, figsize=(7,4))
ax[0].boxplot(df.women_salary)
ax[0].set_xlabel("Women Salary")
ax[1].boxplot(df.men_salary)
ax[1].set_xlabel("Men Salary")
plt.tight_layout()
```

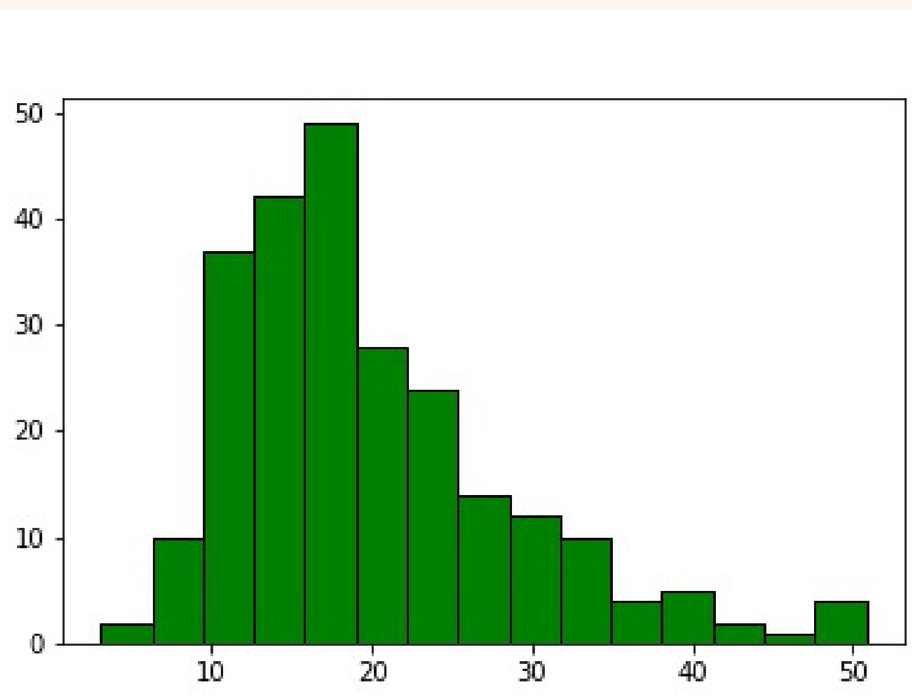
Box Plot (On the same axes)

```
fig,ax=plt.subplots()
ax.boxplot([df.women_salary, df.men_salary])
ax.set_xticklabels(["Women Salary", "Men Salary"])
ax.set_ylabel("Salary")
```



Histogram

```
tips=sns.load_dataset("tips")
fig,ax=plt.subplots()
ax.hist(tips.total_bill, bins=15, edgecolor="black", color="green")
```



- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
- Matplotlib ships with several add-on toolkits, including 3D plotting with mplot3d, axes helpers in axes_grid1 and axis helpers in axisartist.
- For more information please visit https://matplotlib.org/.

CLARUSWAY

WAY TO REINVENT YOURSELF