Practical Notebook 2

Pandas

In this course, we will use pandas to import the data into DataFrame objects. Pandas is a commonly used library working with and manipulating data in various formats, such as txt, csv, excel format, and more.

You can read more about pandas here, or by searching online.

```
# The first thing we need to do is to import pandas
import pandas as pd

# We will aslo change how the floating point numbers are displayed
pd.set_option("display.float_format", lambda x: f"{x:.5f}")
```

Creating our own dataset to file

We will start by creating our own data set, but later on we will import the data from a file.

```
names = ['Alice', 'Bob', 'Charlie']
animals = ['Dog', 'Cat', None]
age = [27, 12, 43]
sex = ['Female', 'Male', 'Male']
```

We will then merge the lists together using the *zip* function.

```
people = list(zip(names, animals, age, sex))
print(people)
```

```
[('Alice', 'Dog', 27, 'Female'), ('Bob', 'Cat', 12, 'Male'), ('Charlie', None, 43, 'Male')]
```

Now we can make our merged list into a DataFrame object by using pandas.

```
df = pd.DataFrame(data=people, columns=['Names','Animals','Age','Sex'])
print(df)
```

```
     Names Animals  Age     Sex
0    Alice     Dog   27  Female
1      Bob     Cat   12    Male
2  Charlie    None   43    Male
```

You can also export the dataframe to a csv file, where we use the function *to_csv* to export the file. You will find the file you created in the folder you are in. (In colab you will find the folder to the left.) The index parameter is set to *False*, i.e. we won't write the row names to the new file (in this case the row names are *0, 1, 2*). The header parameter is set to *True*, i.e. we will write the column names to the file (in this case the column names are *Names*, *Animals*, *Age*, *Sex*). You can change these parameters yourself to see the difference.

```
df.to_csv('test_people.csv', index=False, header=True)
```

Read a dataset from file

To read the data from a csv file we will use the function *read_csv*.

```
df = pd.read_csv('test_people.csv')
print(df)
```

```
     Names Animals  Age     Sex
0    Alice     Dog   27  Female
1      Bob     Cat   12    Male
2  Charlie     NaN   43    Male
```

We can inspect the numerical values in the data using the function describe.

```
print(df.describe())
```

```
            Age
count   3.00000
mean   27.33333
std    15.50269
min    12.00000
25%    19.50000
50%    27.00000
75%    35.00000
max    43.00000
```

And look at one specific column by using the names of the header.

```
print(f"Here you will see the names: \n{df['Names']}")
print(f"\nHere you will see the animals: \n{df['Animals']}")
print(f"\nHere you will see the ages: \n{df['Age']}")
print(f"\nHere you will see the sex: \n{df['Sex']}")
```

```
Here you will see the names:
0      Alice
1        Bob
2    Charlie
Name: Names, dtype: object

Here you will see the animals:
0    Dog
1    Cat
2    NaN
Name: Animals, dtype: object

Here you will see the ages:
0    27
1    12
2    43
Name: Age, dtype: int64
```

```
Here you will see the sex:
0    Female
1      Male
2      Male
Name: Sex, dtype: object
```

You can also divide the groups into females and males.

```
male, female = df['Sex'].value_counts()
print(f"Here we have {male} male(s) and {female} female(s).")
```

```
Here we have 2 male(s) and 1 female(s).
```

By looking only at one column, as we did before, we can find some interesting data about it as well.

```
# finding the mean value of the ages (with 2 decimals)
print(f"mean: {df['Age'].mean():.2f}")
# and the standard deviation (with 2 decimals)
print(f"std: {df['Age'].std():.2f}")
```

```
mean: 27.33
std: 15.50
```

Titanic

Now we will download and use a larger dataset, to get a better understanding about the pandas library. The dataset contains passenger data from Titanic, and later on we will predict "what sort of people were most likely to survive?". The passenger data has 7 features: Name, Sex, Socio-economic class, Siblings/Spouses Aboard, Parents/Children Aboard and Fare and a binary responce variable "survived".

```
# Downloading the titanic dataset
!wget https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv
```

```
--2022-02-11 11:39:54--  https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/ti
Résolution de web.stanford.edu (web.stanford.edu)... 171.67.215.200
Connexion à web.stanford.edu (web.stanford.edu)|171.67.215.200|:443... connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : 44225 (43K) [text/csv]
Sauvegarde en : « titanic.csv.1 »

titanic.csv.1       100%[===================>]  43,19K  68,9KB/s    ds 0,6s

2022-02-11 11:39:59 (68,9 KB/s) - « titanic.csv.1 » sauvegardé [44225/44225]
```

Assignment a)

```
# ASSIGNMENT:
# Load the data and get familiar with it

df = pd.read_csv('titanic.csv')
print(df)
```

```
     Survived  Pclass                                               Name  \
0           0       3                             Mr. Owen Harris Braund
1           1       1  Mrs. John Bradley (Florence Briggs Thayer) Cum...
2           1       3                              Miss. Laina Heikkinen
3           1       1        Mrs. Jacques Heath (Lily May Peel) Futrelle
4           0       3                            Mr. William Henry Allen
..        ...     ...                                                ...
882         0       2                               Rev. Juozas Montvila
883         1       1                        Miss. Margaret Edith Graham
884         0       3                     Miss. Catherine Helen Johnston
885         1       1                               Mr. Karl Howell Behr
886         0       3                                 Mr. Patrick Dooley

        Sex      Age  Siblings/Spouses Aboard  Parents/Children Aboard  \
0      male 22.00000                        1                        0
1    female 38.00000                        1                        0
2    female 26.00000                        0                        0
3    female 35.00000                        1                        0
4      male 35.00000                        0                        0
..      ...      ...                      ...                      ...
882    male 27.00000                        0                        0
883  female 19.00000                        0                        0
884  female  7.00000                        1                        2
885    male 26.00000                        0                        0
886    male 32.00000                        0                        0

        Fare
0    7.25000
1   71.28330
2    7.92500
3   53.10000
4    8.05000
..       ...
882 13.00000
883 30.00000
884 23.45000
885 30.00000
886  7.75000
```

```
[887 rows x 8 columns]
```

Assignment b)

```python
# ASSIGNMENT:
# Count the number of males and females

male, female = df['Sex'].value_counts()
print(male, female)
```

```
573 314
```

Assignment c)

```python
# ASSIGNMENT:
# Find the mean fare
print(f"mean: {df['Fare'].mean():.2f}")

# Find the standard deviation of the fare
print(f"std: {df['Fare'].std():.2f}")
```

```
mean: 32.31
std: 49.78
```

Assignment d)

```python
# ASSIGNMENT:
# Count how many survived (1) and how many died (0)

died, survived = df['Survived'].value_counts()
print(died, survived)
```

```
545 342
```

Assignment e)

```python
# ASSIGNMENT:
# count how many women and men survived

survivedPeople = df[df['Survived'] == 1]
survivedWomen, survivedMen = survivedPeople['Sex'].value_counts()
print(survivedWomen, survivedMen)
```

```
233 109
```

Assignment f)

```python
# ASSIGNMENT (1):
# Separate the dataset from Titanic into X and y,
# where y is the column Survived, and X is the rest.
# Inspect the data. Look at for instance the function "describe" in pandas

X = df.loc[:, 'Pclass':]
y = df.loc[:, 'Survived']
print(X.describe())
print()
print(y.describe())

print('-------------')

# ASSIGNMENT (2):
# Standardize the data by subtracting the mean and dividing by the standard deviation.
# Inpect the data again to see that the mean is (close to) zero and the standard deviation

X = (X-X.mean())/X.std()
print(X.describe())
```

```
         Pclass       Age  Siblings/Spouses Aboard  Parents/Children Aboard  \
count 887.00000 887.00000                887.00000                887.00000
mean    2.30552  29.47144                  0.52537                  0.38331
std     0.83666  14.12191                  1.10467                  0.80747
min     1.00000   0.42000                  0.00000                  0.00000
25%     2.00000  20.25000                  0.00000                  0.00000
50%     3.00000  28.00000                  0.00000                  0.00000
75%     3.00000  38.00000                  1.00000                  0.00000
max     3.00000  80.00000                  8.00000                  6.00000

           Fare
count 887.00000
mean   32.30542
std    49.78204
min     0.00000
25%     7.92500
50%    14.45420
75%    31.13750
max   512.32920

count    887.00000
mean       0.38557
std        0.48700
min        0.00000
25%        0.00000
50%        0.00000
75%        1.00000
max        1.00000
```

```
Name: Survived, dtype: float64
-------------
            Age      Fare  Parents/Children Aboard    Pclass  \
count 887.00000 887.00000                887.00000 887.00000
mean    0.00000   0.00000                 -0.00000  -0.00000
std     1.00000   1.00000                  1.00000   1.00000
min    -2.05719  -0.64894                 -0.47471  -1.56040
25%    -0.65299  -0.48974                 -0.47471  -0.36517
50%    -0.10420  -0.35859                 -0.47471   0.83006
75%     0.60392  -0.02346                 -0.47471   0.83006
max     3.57803   9.64251                  6.95594   0.83006

       Siblings/Spouses Aboard
count                887.00000
mean                  -0.00000
std                    1.00000
min                   -0.47559
25%                   -0.47559
50%                   -0.47559
75%                    0.42966
max                    6.76640
```

```
/var/folders/qs/9s0640m51y5fhftgg8ysvryh0000gn/T/ipykernel_11972/2397119901.py:18: FutureWar
  X = (X-X.mean())/X.std()
```

Matplotlib

Matplotlib is a commonly used library for visualizing data in Python. Other visualization libraries exist for Python, such as seaborn, plotly, and more. Beyond the first practical notebook, we do not enforce any particular plotting library, but strongly encourage the use of Matplotlib. Below we will use the plotting functions inside of *matplotlib.pyplot*. You can read more about matplotlib here and pyplot here.

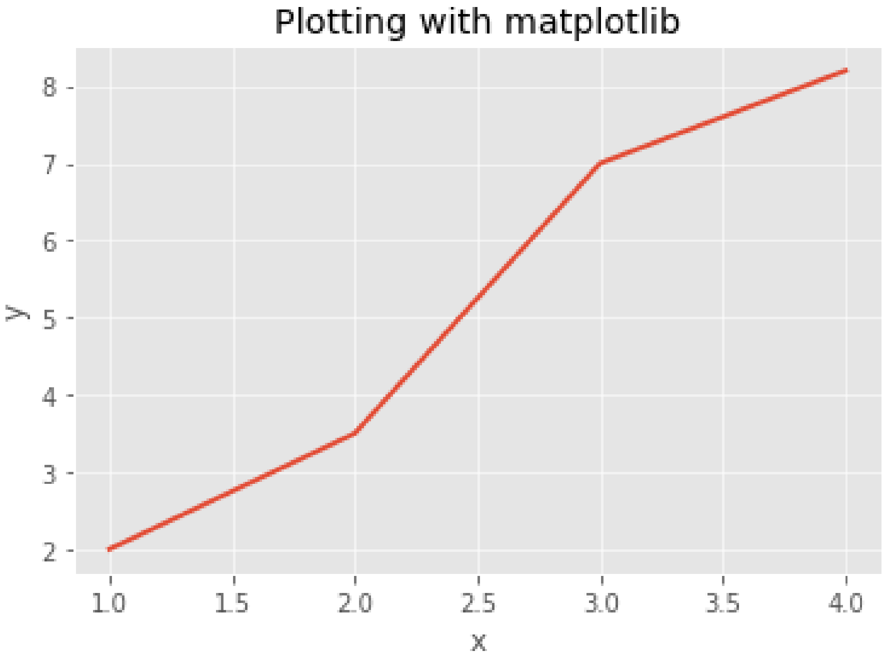
Examples

```python
# import the relevant libraries
import matplotlib.pyplot as plt
import numpy as np
```

We will start by looking at some small lists.

```python
# examples of some datapoint
x = [1,2,3,4]
y = [2,3.5,7,8.2]

# plotting the data using matplotlib.pyplot.plot
plt.plot(x, y)
```

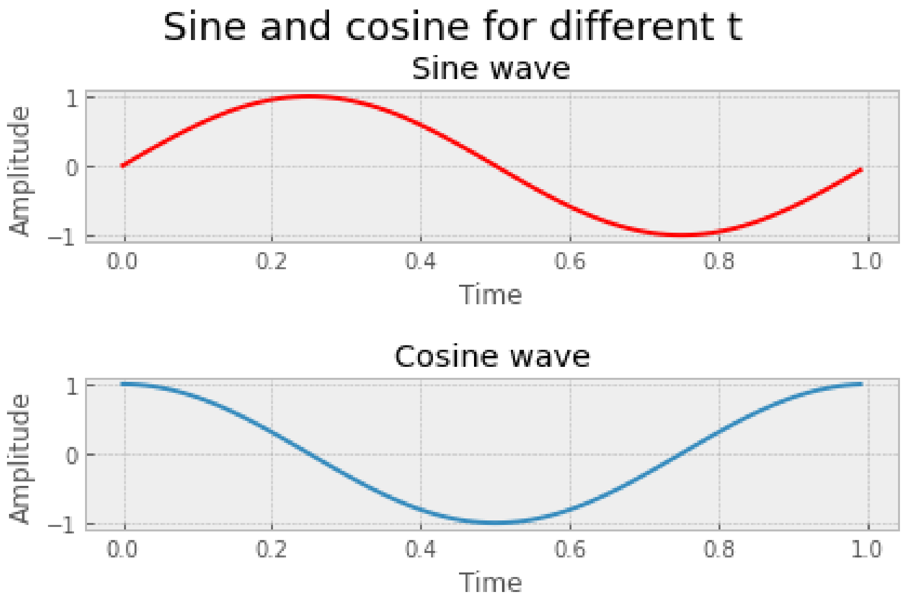
```
# It is important to add labels for the axes and a title
plt.xlabel("x")
plt.ylabel("y")
plt.title("Plotting with matplotlib")
# and always end with show(), which will show you the plot.
plt.show()
```



Plots can also be below each other, or side by side by using subplot.

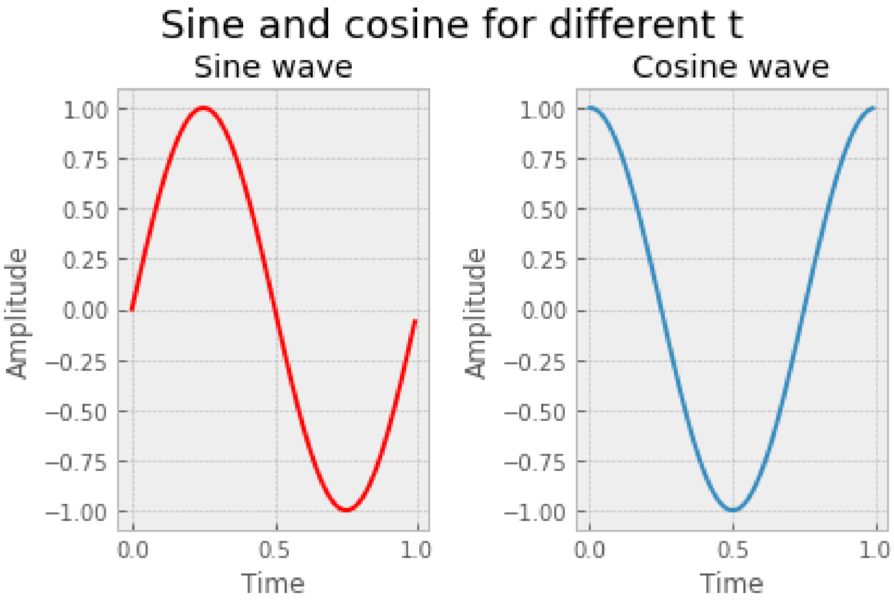
```
# Vertical subplot

plt.style.use('bmh')

t = np.arange(0.0, 1.0, 0.01)
sin = np.sin(2*np.pi*t)
cos = np.cos(2*np.pi*t)

fig = plt.figure()
fig.suptitle("Sine and cosine for different t", fontsize=18)

ax1 = fig.add_subplot(2,1,1)
ax1.plot(t, sin, color='red', lw=2)
ax1.set_ylabel('Amplitude')
```

```
ax1.set_xlabel('Time')
ax1.set_title('Sine wave')

ax2 = fig.add_subplot(2,1,2)
ax2.plot(t, cos)
ax2.set_ylabel('Amplitude')
ax2.set_xlabel('Time')
ax2.set_title('Cosine wave')

fig.tight_layout() # comment out this line to see the difference
fig.subplots_adjust(top=0.85)
plt.show()
```



```
# Horizontal subplot

plt.style.use('bmh')

t = np.arange(0.0, 1.0, 0.01)
sin = np.sin(2*np.pi*t)
cos = np.cos(2*np.pi*t)

fig = plt.figure()
fig.suptitle("Sine and cosine for different t", fontsize=18)

ax1 = fig.add_subplot(1,2,1)  # we have changed (2,1,1) to (1,2,1)
ax1.plot(t, sin, color='red', lw=2)
```

```
ax1.set_ylabel('Amplitude')
ax1.set_xlabel('Time')
ax1.set_title('Sine wave')

ax2 = fig.add_subplot(1,2,2)  # we have changed (2,1,2) to (1,2,2)
ax2.plot(t, cos)
ax2.set_ylabel('Amplitude')
ax2.set_xlabel('Time')
ax2.set_title('Cosine wave')

fig.tight_layout() # comment out this line to see the difference
fig.subplots_adjust(top=0.85)
plt.show()
```



And with different stylings

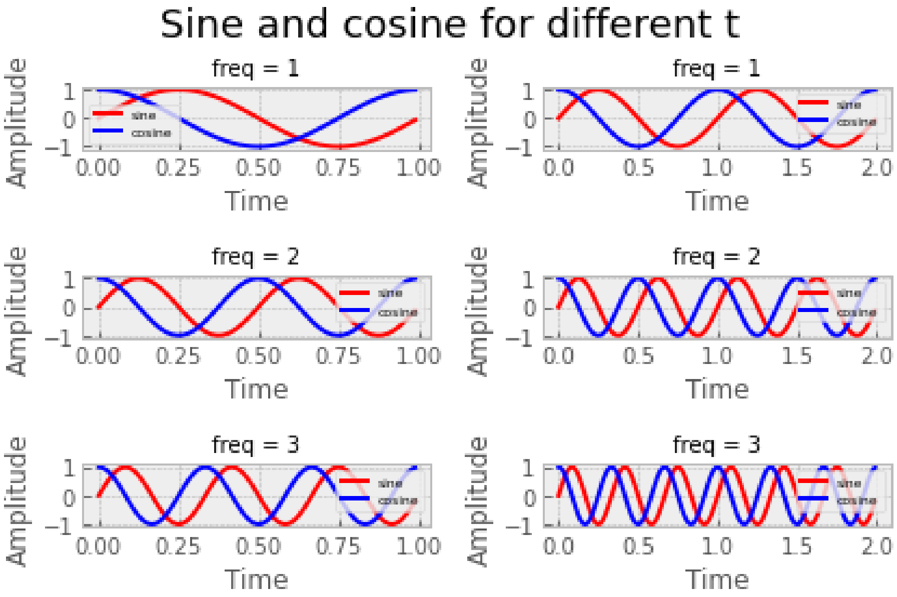
```
# Here are all the different "pre-configured" styles matplot lib supports
# https://matplotlib.org/tutorials/intermediate/artists.html#sphx-glr-tutorials-intermediat
plt.style.available
```

```
['Solarize_Light2',
 '_classic_test_patch',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',
 'seaborn-paper',
 'seaborn-pastel',
 'seaborn-poster',
 'seaborn-talk',
 'seaborn-ticks',
 'seaborn-white',
 'seaborn-whitegrid',
 'tableau-colorblind10']
```

The plotts can also be both below each other and side by side at the same time (as a matrix) as you can see below. Here we have also plotted two graphs together in every figure, and added a color and a label for each one of them.

```
# Matrix subplot

fig = plt.figure()
fig.suptitle("Sine and cosine for different t", fontsize=18)

i = 1
for freq in [1, 2, 3]:
  for t_max in [1, 2]:
    t = np.arange(0.0, t_max, 0.01)
    sin = np.sin(2*freq*np.pi*t)
    cos = np.cos(2*freq*np.pi*t)

    ax = fig.add_subplot(3,2,i)
    ax.plot(t, sin, color='red', lw=2, label='sine')
    ax.plot(t, cos, color='blue', lw=2, label='cosine')
    ax.set_ylabel('Amplitude')
    ax.set_xlabel('Time')
    ax.legend(fontsize=6)
    ax.set_title(f'freq = {freq}', fontsize=10)
    i += 1

fig.tight_layout() # comment out this line to see the difference
fig.subplots_adjust(top=0.85)
```

```
plt.show()
```



Plotting data from Pandas

Now we will plot some of the datapoints from the titanic dataset to visualize it.

```
# Downloading the titanic dataset
!wget https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv
```

```
--2022-02-11 11:40:01--  https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/ti
Résolution de web.stanford.edu (web.stanford.edu)... 171.67.215.200
Connexion à web.stanford.edu (web.stanford.edu)|171.67.215.200|:443... connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : 44225 (43K) [text/csv]
Sauvegarde en : « titanic.csv.2 »

titanic.csv.2       100%[===================>]  43,19K  27,2KB/s    ds 11s

2022-02-11 11:40:13 (3,96 KB/s) - « titanic.csv.2 » sauvegardé [44225/44225]
```

```
# Load the titanic dataset for plotting
import pandas as pd
df = pd.read_csv('titanic.csv')
```

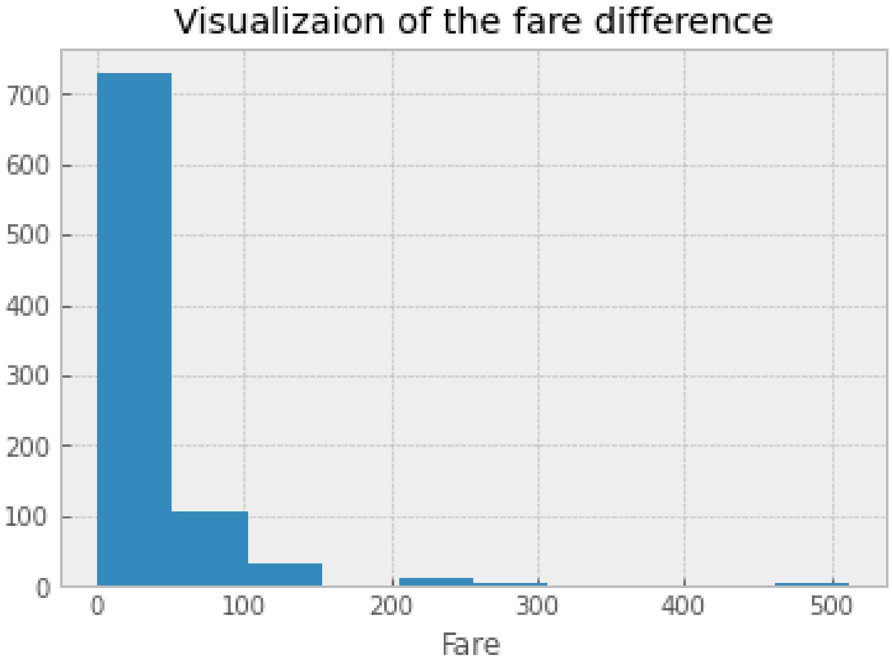
Assignment g)

```
# ASSIGNMENT:
# compare the fare with the class of the ticket

fare_per_pclass = df[['Pclass', 'Fare']].groupby(['Pclass']).mean()
print(fare_per_pclass)
```

```
            Fare
Pclass
1       84.15469
2       20.66218
3       13.70771
```

Assignment h) It might also be a good idea to plot a histogram over the data, to get a better understanding of how the data looks. This can be done using the function *hist* from matplotlib.

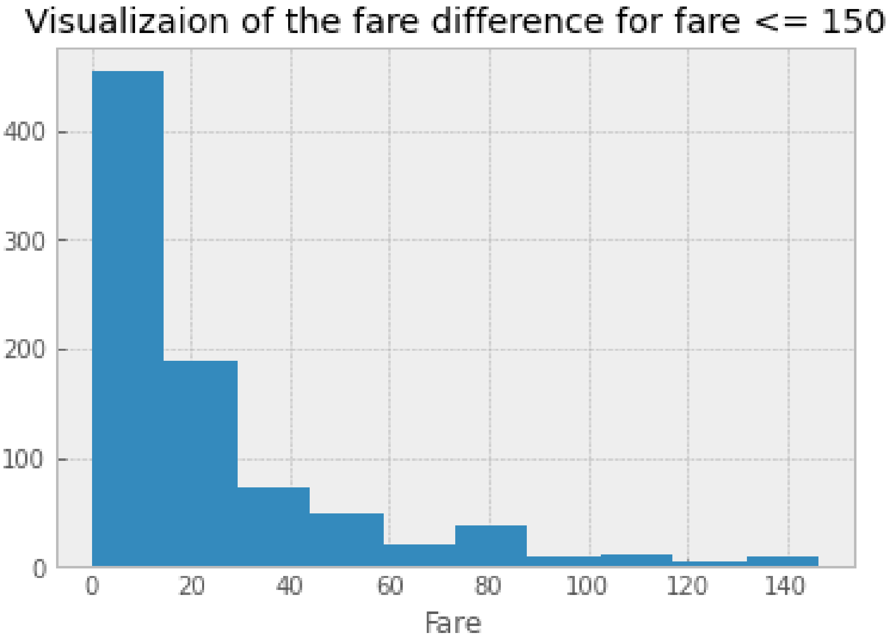
```
fare = df["Fare"]
plt.hist(fare)
plt.xlabel("Fare")
plt.title("Visualizaion of the fare difference")
plt.show()
```



As you can see, most of the people paid less than 150 for the ticket.

```
# ASSIGNMENT:
```

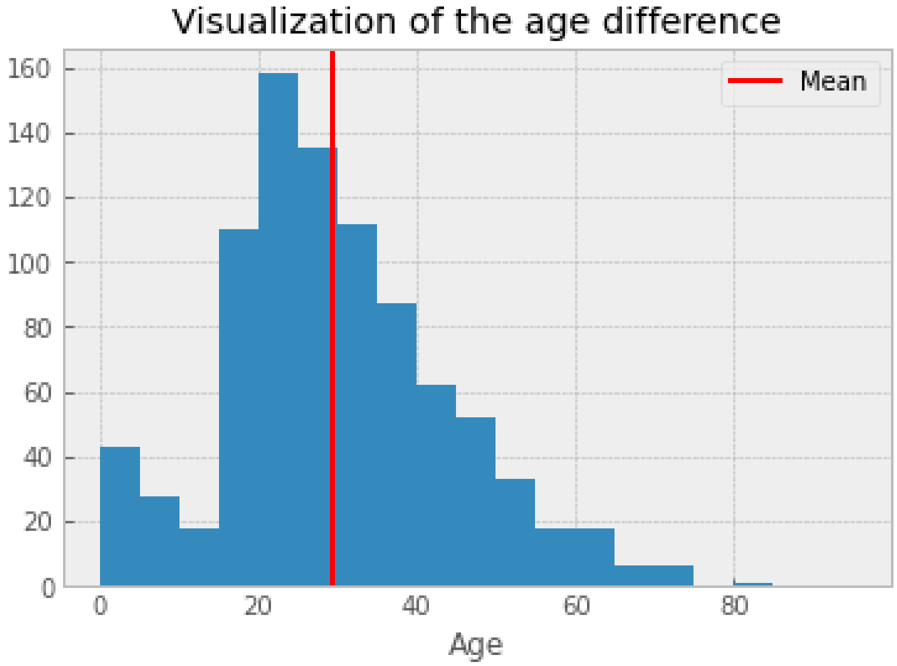
```python
# Plot a histogram over the people who paid less than, or equal to, 150.

fare = df[df['Fare'] <= 150]['Fare']
plt.hist(fare)
plt.xlabel("Fare")
plt.title("Visualizaion of the fare difference for fare <= 150")
plt.show()
```



Assignment i)

```python
# ASSIGNMENT:
# plot a histogram over all the ages with 20 bins. Draw a vertical line at the mean age.

age = df['Age']
plt.hist(age, bins=np.arange(0, 100, 5))
plt.axvline(age.mean(), color='red', label='Mean')
plt.xlabel('Age')
plt.title('Visualization of the age difference')
plt.legend()
plt.show
```

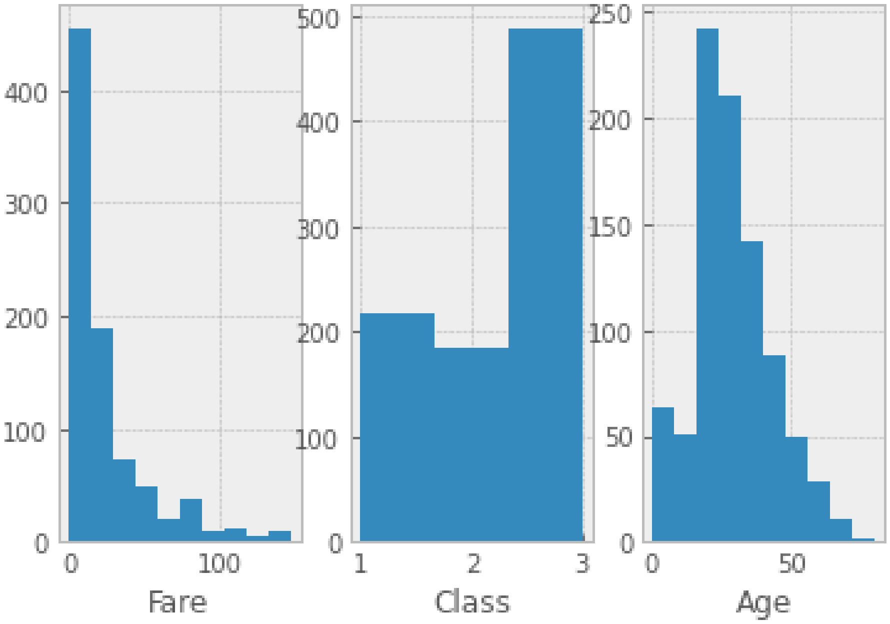
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Assignment j) Sometimes it is better to plot the figures together in one figure instead. This can be done with subplot, as shown in the examples above.

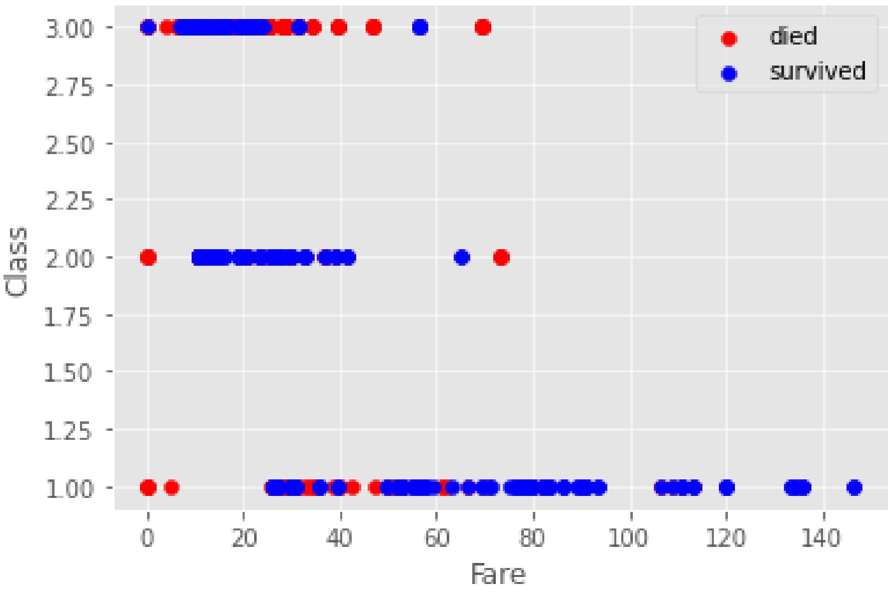
```
# ASSIGNMENT:
# Make a subplot over the Fare, Class, and Age

fig = plt.figure()

ax1 = fig.add_subplot(1, 3, 1)
ax1.hist(fare)
ax1.set_xlabel('Fare')

ax2 = fig.add_subplot(1, 3, 2)
ax2.hist(df['Pclass'], bins=3)
ax2.set_xlabel('Class')

ax3 = fig.add_subplot(1, 3, 3)
ax3.hist(age)
ax3.set_xlabel('Age')
```

```
Text(0.5, 0, 'Age')
```



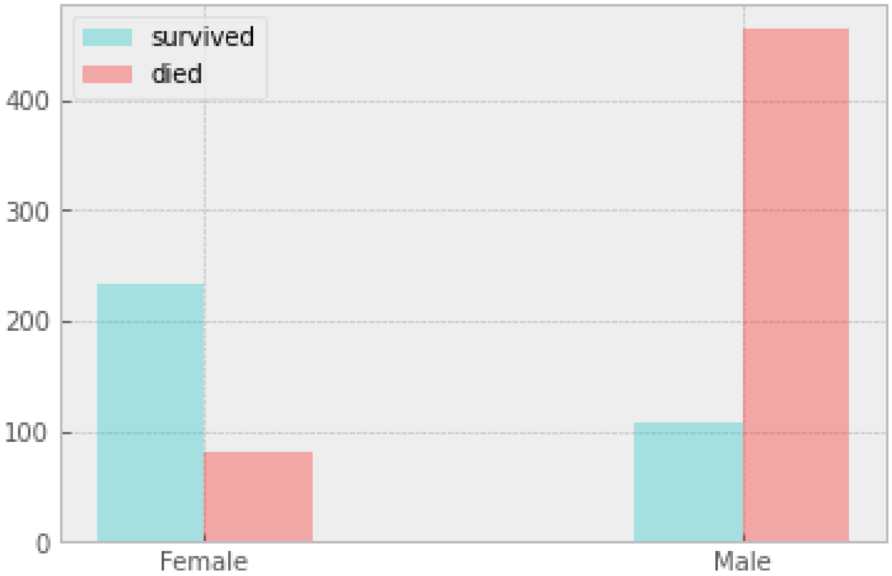
Assignment k) Now we want to compare the fare and class, as we did before, but this time we want to divide them into two colors, depending on if they survived or not.

```
# ASSIGNMENT:
# Make a plot with red dots for all the people who died, and blue dots for the people who s

dead = df[(df['Survived'] == 0) & (df['Fare'] <= 150)]
survived = df[(df['Survived'] == 1) & (df['Fare'] <= 150)]
survived = survived[survived['Fare'] <= 150]
plt.scatter(dead['Fare'], dead['Pclass'], color='red', label='died')
plt.scatter(survived['Fare'], survived['Pclass'], color='blue', label='survived')
plt.xlabel('Fare')
plt.ylabel('Class')
plt.legend()
plt.show()
```



Assignment l) It might also be interesting to visualize how many of the men and women survived. This can be done with the bar function, which will be given to you.

```
# ASSIGNMENT:
# Calculate how many women and men died and survived.

# female_survived, male_survived = df[(df['Sex'] == 'female') & (df['Survived'] == 1)], df[
female_survived, male_survived = len(df[(df['Survived'] == 1) & (df['Sex'] == 'female')]),le
female_died, male_died = len(df[(df['Survived'] == 0) & (df['Sex'] == 'female')]),len(df[(df

plt.bar([0.9,1.9], [female_survived, male_survived] , color='c', label='survived', width=0.2
plt.bar([1.1, 2.1], [female_died, male_died] , color='r', label='died', width=0.2, alpha=0.3
plt.xticks([1,2], ['Female','Male'])
plt.legend()
plt.show()
```



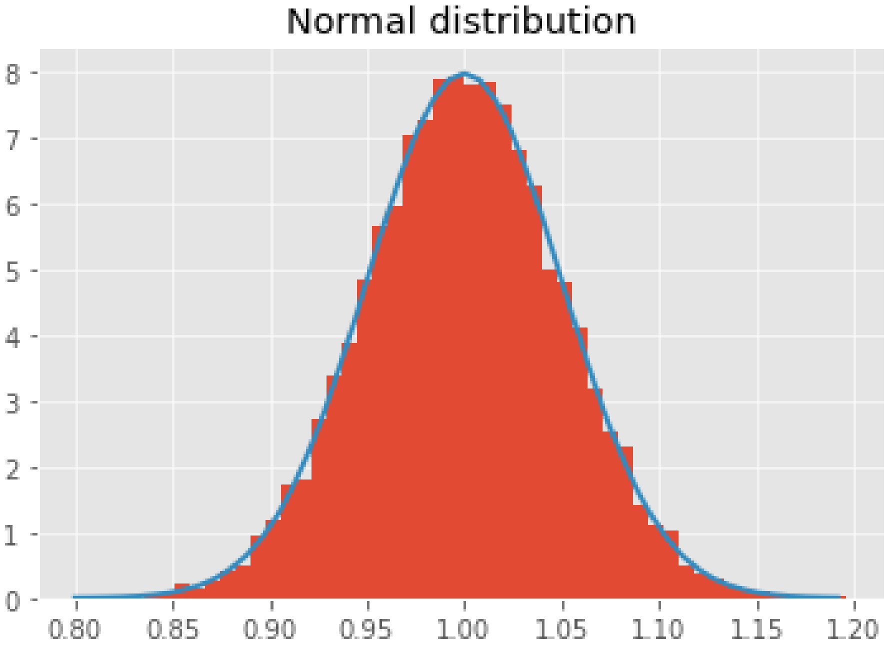
(Optional) Ploting a histogram of a random distribution

OPTIONAL:
Plotting a Histogram of Random values

Your task is to generate 10000 random numbers that follows the normal distribution, with a mean, $\mu = 1$, and variance $\sigma^2 = 0.25$.

Plot the **normalized** histogram with 50 bars and a contour plot.

```
import numpy as np
import matplotlib.pyplot as plt

plt.style.use('ggplot')
np.random.seed(42)

# OPTIONAL ASSIGNMENT:
# Draw 10000 random values from a normal distribution with:
#   mu = 1, sigma2 = 0.25
#
# Plot the histogram and cumulative distribution

mu, sigma = 1, 0.05
normal_distribution = np.random.normal(mu, sigma, 10000)

x = np.arange(0.8, 1.2, 0.4/50)
plt.hist(normal_distribution, bins=50, density=True)
plt.plot(x, 1/(sigma * np.sqrt(2 * np.pi)) * np.exp( - (x - mu)**2 / (2 * sigma**2)))
plt.title('Normal distribution')
plt.show()
```


Normal distribution