Worksheet 02

Name: Mohit Sai Gutha UID: U48519832

Topics

- **Effective Programming**

Effective Programming

a) What is a drawback of the top down approach?

By prioritizing high-level design and structure during the initial stages of development, details may be overlooked, potentially resulting in costly errors or revisions later in the process.

b) What is a drawback of the bottom up approach?

Building individual parts without considering how they fit together can make it hard to combine them later. Planning everything from the start is difficult.

c) What are 3 things you can do to have a better debugging experience?

1. **Dont panic. Be willing to walk away from the code and get back to it if no progress is made.**
2. **read the error and try to understand what it says.**
3. **Re-read the code and try to understand it's flow. This was also help you to place debugging tools like print statements or breakpoints in appropriate locations.**
4. **Look online for some help if you are unable to figure it out by yourself. If you are taking too long then try to reach out to someone for help.**
5. **Make sure you are implementing regular sanity checks to ensure your code is perfectly functioning till a point.**

d) (Optional) Follow along with the live coding. You can write your code here:

In [ ]:

```
```

Exercise

This exercise will use the Titanic dataset (https://www.kaggle.com/c/titanic/data). Download the file named `train.csv` and place it in the same folder as this notebook.

The goal of this exercise is to practice using pandas methods. If your:

1. **code is taking a long time to run**
2. **code involves for loops or while loops**
3. **code spans multiple lines**

look through the pandas documentation for alternatives. This cheat sheet may come in handy.

a) Complete the code below to read in a filepath to the `train.csv` and returns the DataFrame.

In [65]:

```
import pandas as pd

df = pd.read_csv('train.csv')
df.describe()
```

Out[65]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

b) Complete the code so it returns the number of rows that have at least one empty column value

In [66]:

```
print("there are " +  str(df.isnull().any(axis=1).sum()) + " rows with at least one empty value")
```

```
there are 708 rows with at least one empty value
```

c) Complete the code below to remove all columns with more than 200 NaN values

In [67]:

```
df = df.dropna(thresh=df.shape[0] - 200, axis=1)
```

```
df.columns
```

Out[67]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Embarked'],
      dtype='object')
```

d) Complete the code below to replaces `male` with 0 and `female` with 1

In [68]:

```
df['Sex'] = df['Sex'].replace({'male': 0, 'female': 1}).astype(int)
df.head()
```

Out[68]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.0500	S

e) Complete the code below to add four columns `First Name`, `Middle Name`, `Last Name`, and `Title` corresponding to the value in the `name` column.

For example: `Braund, Mr. Owen Harris` would be:

First Name	Middle Name	Last Name	Title
Owen	Harris	Braund	Mr

Anything not clearly one of the above 4 categories can be ignored.

In [69]:

```
df[['Last Name', 'Title', 'First Name', 'Middle Name']] = df['Name'].str.extract(r'(?P<Last_Name>.*), (?P<Title>[^.]+)\. (?P<Firs
t_Name>[^ ]+)(?: (?P<Middle_Name>[^(.]+))?')
df.head()
```

Out[69]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Last Name	Title	First Name	Middle Name

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Last Name	Title	First Name	Middle Name
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500	S	Braund	Mr	Owen	Harris
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	C	Cumings	Mrs	John	Bradley
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	S	Heikkinen	Miss	Laina	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.1000	S	Futrelle	Mrs	Jacques	Heath
4	5	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.0500	S	Allen	Mr	William	Henry

f) Complete the code below to replace all missing ages with the average age

In [70]:

```
df['Age'] = df['Age'].fillna(df['Age'].mean())
df.head(6)
```

Out[70]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Last Name	Title	First Name	Middle Name
0	1	0	3	Braund, Mr. Owen Harris	0	22.000000	1	0	A/5 21171	7.2500	S	Braund	Mr	Owen	Harris
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.000000	1	0	PC 17599	71.2833	C	Cumings	Mrs	John	Bradley
2	3	1	3	Heikkinen, Miss. Laina	1	26.000000	0	0	STON/O2. 3101282	7.9250	S	Heikkinen	Miss	Laina	NaN
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.000000	1	0	113803	53.1000	S	Futrelle	Mrs	Jacques	Heath
4	5	0	3	Allen, Mr. William Henry	0	35.000000	0	0	373450	8.0500	S	Allen	Mr	William	Henry
5	6	0	3	Moran, Mr. James	0	29.699118	0	0	330877	8.4583	Q	Moran	Mr	James	NaN

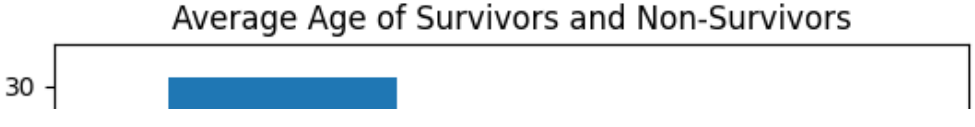
g) Plot a bar chart of the average age of those that survived and did not survive. Briefly comment on what you observe.

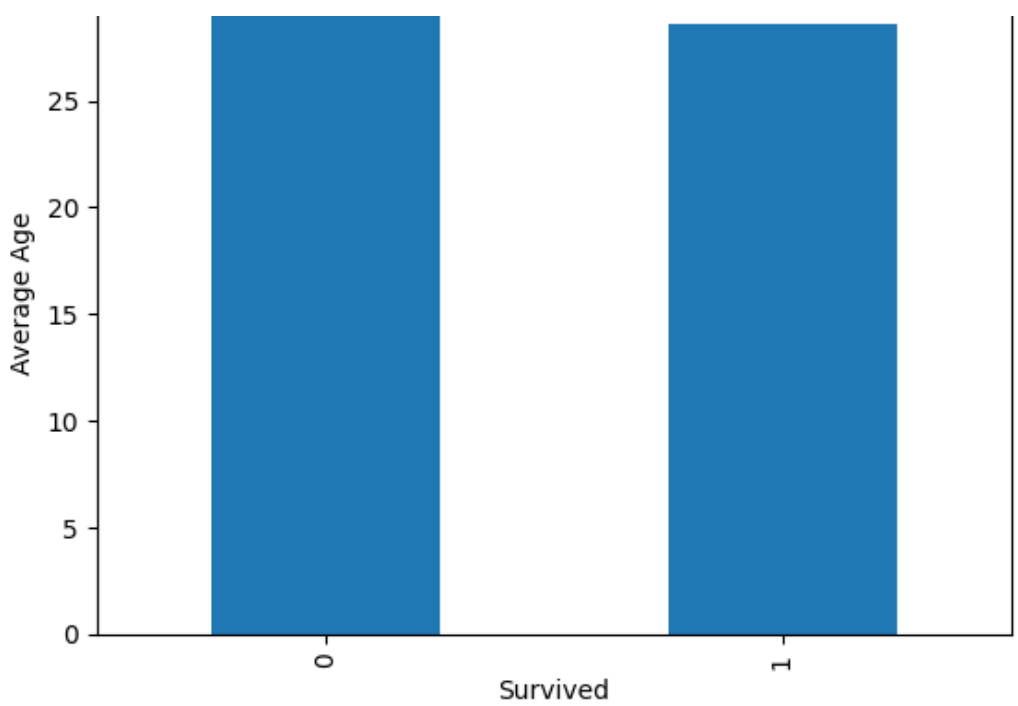
In [72]:

```
df.groupby('Survived')['Age'].mean().plot(kind='bar', xlabel='Survived', ylabel='Average Age', title='Average Age of Survivors and Non-Survivors')
```

Out[72]:

```
<Axes: title={'center': 'Average Age of Survivors and Non-Survivors'}, xlabel='Survived', ylabel='Average Age'>
```





The mean age of survivors appears to be slightly lower than that of the mean age of non-survivors. Though there is a difference, this difference is too small to draw any conclusions on the correlation between the age of the passengers and their survival.