Worksheet 02

Name: Shivam Goyal

UID: U35920740

Topics

- Effective Programming

Effective Programming

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

Debugging becomes very difficult as the complexity of the code increases.

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

Building a solution entirely from individual contributions can sometimes result in a final product that lacks unity. Each piece might be well-designed, but they may not integrate seamlessly, leading to a disjointed user experience or system inefficiency.

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

- Add logs and print statements to check where the code is failing
- Divide the issue into smaller, manageable parts. This helps you pinpoint the root cause more efficiently.
- Search for existing solutions and discussions about similar bugs on Community forums and documentation.

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

```
print("This is an optional part")
```

```
This is an optional part
```

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.

```
import pandas as pd

df =
pd.read_csv("/Users/shivam_goyal/Desktop/CS506/titanic/train.csv")
df.describe()
# print(len(df))
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

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

```
rows_with_empty_values = df[df.isnull().any(axis=1)].shape[0]
print("there are " +  str(rows_with_empty_values) + " 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

```
df = df.dropna(thresh=len(df) - 200, axis=1)
df.columns
# df.shape
```

```
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

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

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name  Sex   Age  SibSp
Parch  \
0                            Braund, Mr. Owen Harris    0  22.0      1
0
1  Cumings, Mrs. John Bradley (Florence Briggs Th...    1  38.0      1
0
2                             Heikkinen, Miss. Laina    1  26.0      0
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)    1  35.0      1
0
4                           Allen, Mr. William Henry    0  35.0      0
0

             Ticket     Fare Embarked
0         A/5 21171   7.2500        S
1          PC 17599  71.2833        C
2  STON/O2. 3101282   7.9250        S
3            113803  53.1000        S
4            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.

```
def extract_names(name_str):
    if "(" in name_str:
        name_str=name_str.split(" (")[0]
        parts=name_str.split(', ')
        last_name=parts[0].strip()
        temp1=parts[-1].split('. ')
        title=temp1[0].strip()
        temp2=temp1[-1].split(' ')
```

```
        first_name=temp2[0].strip()
        middle_name=temp2[-1].strip()
    else:
        parts=name_str.split(', ')
        last_name=parts[0].strip()
        temp1=parts[-1].split('. ')
        title=temp1[0].strip()
        temp2=temp1[1].split(' ')
        first_name=temp2[0].strip()
        middle_name=temp2[-1].strip()
    return first_name,middle_name,last_name,title

df[["First Name", "Middle Name", "Last Name", "Title"]] =
df["Name"].apply(extract_names).tolist()
df.head()
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name  Sex   Age  SibSp
Parch  \
0                            Braund, Mr. Owen Harris    0  22.0      1
0
1  Cumings, Mrs. John Bradley (Florence Briggs Th...    1  38.0      1
0
2                             Heikkinen, Miss. Laina    1  26.0      0
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)    1  35.0      1
0
4                           Allen, Mr. William Henry    0  35.0      0
0

             Ticket     Fare Embarked First Name Middle Name  Last
Name Title
0         A/5 21171   7.2500        S       Owen      Harris
Braund    Mr
1          PC 17599  71.2833        C       John     Bradley
Cumings   Mrs
2  STON/O2. 3101282   7.9250        S      Laina       Laina
Heikkinen  Miss
3            113803  53.1000        S    Jacques       Heath
Futrelle   Mrs
4            373450   8.0500        S    William       Henry
Allen    Mr
```

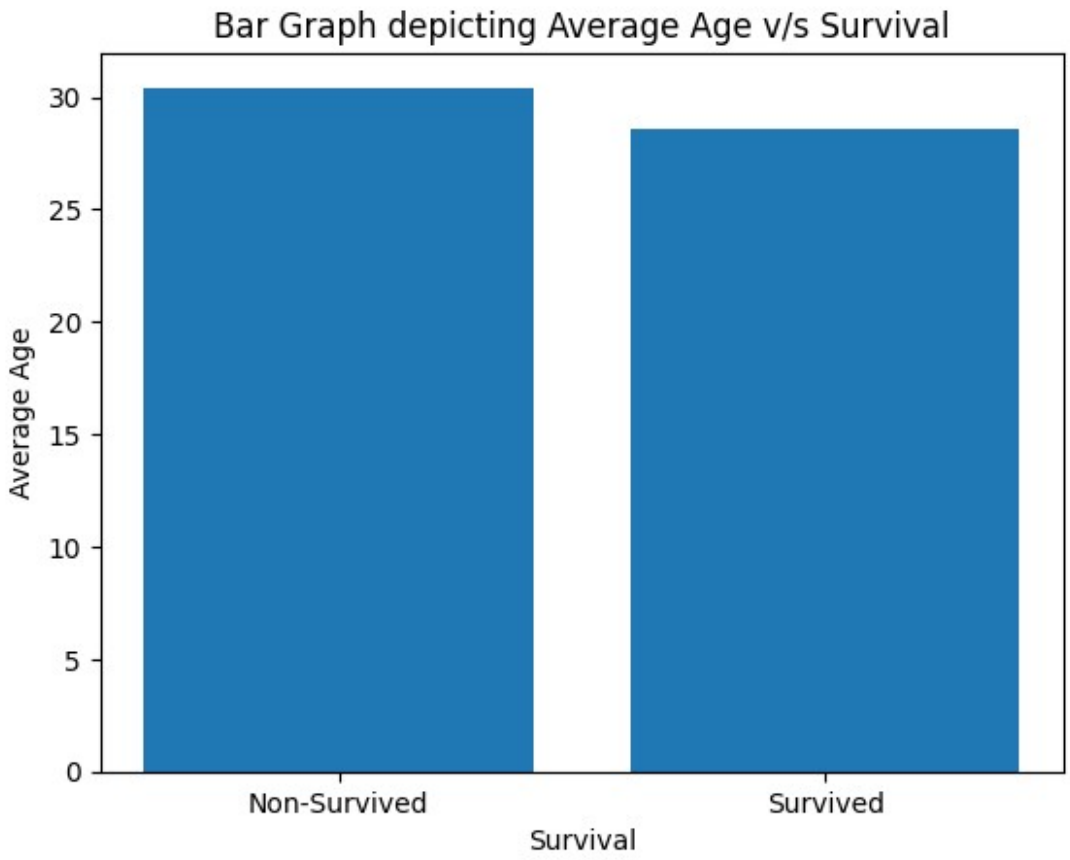
f) Complete the code below to replace all missing ages with the average age

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

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name  Sex   Age  SibSp
Parch  \
0                            Braund, Mr. Owen Harris    0  22.0      1
0
1  Cumings, Mrs. John Bradley (Florence Briggs Th...    1  38.0      1
0
2                             Heikkinen, Miss. Laina    1  26.0      0
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)    1  35.0      1
0
4                           Allen, Mr. William Henry    0  35.0      0
0

             Ticket     Fare Embarked First Name Middle Name  Last
Name Title
0         A/5 21171   7.2500        S       Owen      Harris
Braund    Mr
1          PC 17599  71.2833        C       John     Bradley
Cumings   Mrs
2  STON/O2. 3101282   7.9250        S      Laina       Laina
Heikkinen  Miss
3            113803  53.1000        S    Jacques       Heath
Futrelle   Mrs
4            373450   8.0500        S    William       Henry
Allen    Mr
```

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

```
l = ['Non-Survived','Survived']

import matplotlib.pyplot as plt
plt.bar(l,[df[df['Survived']==0]['Age'].mean(),df[df['Survived']==1]
['Age'].mean()])
plt.xlabel('Survival')
plt.ylabel('Average Age')
plt.title('Bar Graph depicting Average Age v/s Survival')

Text(0.5, 1.0, 'Bar Graph depicting Average Age v/s Survival')
```



Average age of people who survived is less than those who did not survive