worksheet_02

February 3, 2024

1 Worksheet 02

Name: Daniyal Ahmed UID: U11469883

1.0.1 Topics

- Effective Programming

1.0.2 Effective Programming

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

Because you have not wrote the functions, bugs can arise that you have not thought of when you start writing out your functions

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

Because you are not implementing your programs first, you may over do it and make functions you may not actually need.

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

I can stop looking for hours on end for a quick fix, read the error and re-read my code and make it concise enough that the error is easier to spot.

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

[1]:
```
class Board:

    def __init__(self):
        self.board = [["" for _ in range*8] for _  in range(8)]

        def __repr__(self):
            res= ''
            for row in range(8):
                for col in range(8):
                    res+= self.board[row][col]
                    res+=""
                res+="\n"

            return res
```

```
    def set_queen_at(self,row,col):
        self.board[row][col]= "Q"




    def unset_queen_on(self, row):
        self.board[row]= ["-" for _ in range(8)]


    def is_valid_col(self, row, col):
        for i in range(8):
            if i != row and self.board[i][col]== "Q":
                return False

        return True

    def is_valid_row(self, row, col):
        for j in range(8):
            if j != col and self.board[row][j]== "Q":
                return False

        return True

    def is_valid_move(self, row,col):
        if not self.is_valid_row(self,row,col):
            return False

        if not self.is_valid_col(row,col):
            return False


        return True

    def get_queen_on_row(self,row):
        for i in range(8):
            if self.board[row][i] == 'Q':
                return i
        raise ValueError("No queen on this row")

    def find_solution(self):
        row = 0
        col = 0

        while row < 8:
            if self.is_valid_move(row,col):
```

```
                self.set_queen_at(row,col)
                row+=1
                col =0

        else:
            col+=1
            if col >= 8:
                col =self.get_queen_on_row(row-1)
                col+=1
                row -=1
```

1.1 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.

[2]:
```
import pandas as pd

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

'''I got this from Lab'''
```

[2]:
```
       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

[3]:
```
print("there are " +  str(df.isnull().any(axis=1).sum()) + " rows with at least␣
 ↪one empty value")
'''We know from the cheatsheet provided pd.isnull tells us if a row is empty or␣
 ↪now  the .any tells us
That the row/ column can have any NaN value is it to be considered in this␣
 ↪operation finally the axis = 1 tells
the any operation to look at the rows instead of the columns which is default␣
 ↪and sum sums up all the values that isnull.any
returns to be true '''
```

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

[3]:
```
'We know from the cheatsheet provided pd.isnull tells us if a row is empty or
now  the .any tells us \nThat the row/ column can have any NaN value is it to be
considered in this operation finally the axis = 1 tells\nthe any operation to
look at the rows instead of the columns which is default and sum sums up all the
values that isnull.any \nreturns to be true '
```

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

[4]:
```
df = df.drop(columns=df.columns[df.isnull().sum() >=200])

'''Because I couldn't find df.columns in the cheatsheet or the lab solutions I␣
 ↪went ahead and looked
at the documentation, from what I gathered, df.columns returns a sort of list␣
 ↪with all the columns
we put df.isnull().sum() >= 200 inside of it to tell it to return the columns␣
 ↪with 200 or more null values,
and obviously from both the cheatsheat and the lab we know that df.drop drops␣
 ↪the specified columns
link to documentation that helped me:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.columns.
 ↪html#pandas.DataFrame.columns
https://pandas.pydata.org/docs/reference/api/pandas.isnull.html
  '''
```

[4]:
```
"Because I couldn't find df.columns in the cheatsheet or the lab solutions I
went ahead and looked \nat the documentation, from what I gathered, df.columns
returns a sort of list with all the columns \nwe put df.isnull().sum() >= 200
inside of it to tell it to return the columns with 200 or more null values,
\nand obviously from both the cheatsheat and the lab we know that df.drop drops
```

```
the specified columns\nlink to documentation that helped me: \nhttps://pandas.py
data.org/docs/reference/api/pandas.DataFrame.columns.html#pandas.DataFrame.colum
ns\nhttps://pandas.pydata.org/docs/reference/api/pandas.isnull.html\n  "
```

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

[5]:
```
df['Sex']= df['Sex'].str.replace('female','1').str.replace("male","0")
'''I found this from the lab, the only thing I had trouble with was realizing␣
 ↪that we should replace
female first then male, the reason being that female has the word MALE in it '''
```

[5]:
```
'I found this from the lab, the only thing I had trouble with was realizing that
we should replace \nfemale first then male, the reason being that female has the
word MALE in it '
```

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.

[6]:
```
df[['First Name', 'Middle Name', 'Last Name', 'Title']] = df.apply(lambda x: ([
    x['Name'].split(" ")[2] if len(x['Name'].split(" ")) > 2 else None,
    x['Name'].split(" ")[3] if len(x['Name'].split(" ")) > 3 else None,
    x['Name'].split(" ")[0] ,
    x['Name'].split(" ")[1] if len(x['Name'].split(" ")) > 1 else None
]), axis=1, result_type='expand')

'''This one took me quite a awhile to put together but I used the example from␣
 ↪the
lab:
 df['CaloriesPerRadius'] = df.apply(calculate_cal_per_radius, axis = 1)
 This helped me realize that I can make a lambda function, where it returns a␣
 ↪list of names in
  the exact order the question asks for however it took me longer to understand␣
 ↪why I was getting the error :
  Columns must be same length as key, so I looked at the apply function␣
 ↪documentation to see if I was doing something wrong and I tried to turn the␣
 ↪list into columns since I
  was getting an error that had to do with columns and it worked !
  source: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.apply.
 ↪html
```

```
'''
```

[6]:

```
   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  \
0         A/5 21171   7.2500        S       Owen      Harris     Braund,
1          PC 17599  71.2833        C       John     Bradley    Cumings,
2  STON/O2. 3101282   7.9250        S      Laina        None  Heikkinen,
3            113803  53.1000        S    Jacques       Heath   Futrelle,
4            373450   8.0500        S    William       Henry      Allen,

   Title
0    Mr.
1   Mrs.
2  Miss.
3   Mrs.
4    Mr.
```

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

[7]:

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

'''I got fillna from the cheatsheet and the mean function from lab'''


df.head(100)
```

[7]:

```
    PassengerId  Survived  Pclass  \
0             1         0       3
1             2         1       1
2             3         1       3
3             4         1       1
4             5         0       3
..          …       …     …
95           96         0       3
```

```
96           97         0       1
97           98         1       1
98           99         1       2
99          100         0       2

                                                 Name Sex         Age  SibSp  \
0                             Braund, Mr. Owen Harris   0   22.000000      1
1   Cumings, Mrs. John Bradley (Florence Briggs Th…   1   38.000000      1
2                              Heikkinen, Miss. Laina   1   26.000000      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)   1   35.000000      1
4                            Allen, Mr. William Henry   0   35.000000      0
..                                                 …  ..         …    …
95                         Shorney, Mr. Charles Joseph   0   29.699118      0
96                           Goldschmidt, Mr. George B   0   71.000000      0
97                    Greenfield, Mr. William Bertram   0   23.000000      0
98                Doling, Mrs. John T (Ada Julia Bone)   1   34.000000      0
99                                   Kantor, Mr. Sinai   0   34.000000      1

    Parch            Ticket     Fare Embarked First Name Middle Name  \
0       0         A/5 21171   7.2500        S       Owen      Harris
1       0          PC 17599  71.2833        C       John     Bradley
2       0  STON/O2. 3101282   7.9250        S      Laina        None
3       0            113803  53.1000        S    Jacques       Heath
4       0            373450   8.0500        S    William       Henry
..    …               …      …      …         …         …
95      0            374910   8.0500        S    Charles      Joseph
96      0          PC 17754  34.6542        C     George           B
97      1          PC 17759  63.3583        C    William     Bertram
98      1            231919  23.0000        S       John           T
99      0            244367  26.0000        S      Sinai        None

       Last Name  Title
0         Braund,    Mr.
1        Cumings,   Mrs.
2      Heikkinen,  Miss.
3       Futrelle,   Mrs.
4          Allen,    Mr.
..           …    …
95       Shorney,    Mr.
96   Goldschmidt,    Mr.
97    Greenfield,    Mr.
98        Doling,   Mrs.
99        Kantor,    Mr.

[100 rows x 15 columns]
```

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

on what you observe.

[8]:
```
df.groupby('Survived')['Age'].mean().plot(kind='bar')

'''I knew some pandas so I knew I can make a scatter graph but I was unsure␣
 ↪about a bar graph until I looked through the documentation
https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.DataFrame.
 ↪plot.html'''
```

[8]:
```
'I knew some pandas so I knew I can make a scatter graph but I was unsure about
a bar graph until I looked through the
documentation\nhttps://pandas.pydata.org/pandas-
docs/version/0.23.4/generated/pandas.DataFrame.plot.html'
```