Welcome to your 310 Portfolio

A progression of learning for CSC/DSP 310: Programming for Data Science at University of Rhode Island.

About Me

Hello, my name is Brianna MacDonald and I am a Senior at URI with a double major in Computer Engineering and Chinese with a double minor in Cyber Security and Data Science. I've been studying Chinese for about 4 years and I recently had the oppertunity to go to Shanghai and Beijing last Winter right before COVID-19.

Data Science, to me

Data Science is the intersection between computer science, statistics, and domain knowledge. Data Science has many different uses, such as in medical sciences or in machine learning.

There are many different components of Data Science. The four main components of Data Science are Data Strategy, Data Engineering, Data Analysis and Modeling, and Data Visualization/Operationalization. Data Strategy is about determing what data you want to gather and why. It makes a connection between the data you want to gather and the goals for that data. Data Engineering is about the systems and technology that are used to leverage, access, organize, and use the data. Data Analysis/Modeling is about describing or predicting data, creating analysis and assumptions about data, and mathematically modeling the data. Data Visualization/Operationalization is about visualizing data and understanding how different visuals describe the data, as well as making the data operational by making a machine/person make a decision or action based on the computing of the data.

Compute the Grade for CSC/DSP 310

- To run by entering values into function, please run **compute_grade()** with desired values.

```
def compute_grade(num_level1, num_level2, num_level3):
    """
    Computes a grade for CSC/DSP 310 from numbers of achievements earned at each level
    :param num_level1: int, number of level 1 achievements earned
    :param num_level2: int, number of level 2 achievements earned
    :param num_level3: int, number of level 3 achievements earned
    :return: letter_grade: string, letter grade with possible modifier (+/-)
    """
    # Initializing Variables
    letter_grade = ""
    total_grade = num_level1 + num_level2 + num_level3

    # Error Handling
    if total_grade > 45:
        print("Invalid total. Please re-enter values.")

    # Definitions of Grades
    else:
        if 3 <= num_level1 < 5:
            letter_grade = 'D'
        elif 5 <= num_level1 < 10:
            letter_grade = 'D+'
        elif 10 <= num_level1 < 15:
            letter_grade = 'C-'
        elif num_level1 == 15 and 0 <= num_level2 < 5:
            letter_grade = 'C'
        elif num_level1 == 15 and 5 <= num_level2 < 10:
            letter_grade = 'C+'
        elif num_level1 == 15 and 10 <= num_level2 < 15:
            letter_grade = 'B-'
        elif num_level1 == 15 and num_level2 == 15 and 0 <= num_level3 < 5:
            letter_grade = 'B'
        elif num_level1 == 15 and num_level2 == 15 and 5 <= num_level3 < 10:
            letter_grade = 'B+'
        elif num_level1 == 15 and num_level2 == 15 and 10 <= num_level3 < 15:
            letter_grade = 'A-'
        elif num_level1 == 15 and num_level2 == 15 and num_level3 == 15:
            letter_grade = 'A'
        else:
            print("Does not translate to letter grade.")
    print(f'Your grade is {letter_grade}.')
```

The example below will give a grade of a C.

```
# Example 1
compute_grade(15, 2, 0)
```

The example below will give a grade of a B.

```
# Example 2
compute_grade(15, 15, 2)
```

The example below will give a grade of an A-.

```
# Example 3
compute_grade(15, 15, 12)
```

Submission 3

In this submission, I will be attempting to earn **summarize level 3** and **visualize level 3**.

```
### Some code here
```

Submission 4

For this section of my portfolio, I will be attempting to earn **summarize level 3, visualize level 3, and prepare level 2**. In this first section, I will be correcting **Assignment 4** as well as adding additional graphs and plots to summarize and visualize the data.

Prepare Level 2

For Assignment 4, I was a bit wary of cleaning and preparing data as it was a new concept for me. As I showed briefly in *submission_1*, I have gained some knowledge when dealing with cleaning and preparing data. Similarly, I will be using data from the same database as Assignment 4, located here.

This table shows the different percentages of college students (16-24 years old) who were employed, how many hours they worked a week, and their level of institution.

```
# Imports
import pandas as pd
```

```
# First, let's take a look at what we are dealing with
enroll_uc_table = pd.read_excel("tabn503.20.xls")
enroll_uc_table.head(10)
```

```
enroll_uc_table.tail(10)
```

As we can see from just the table above, the data is quite messy. There are multiply NaN values, headers in the wrong place, as well as trailing characters. Through this submission, I will correct these errors.

```
enroll_clean = pd.read_excel('tabn503.20.xls', skipfooter=7, header=list(range(3)), skiprows=2)
enroll_clean.set_index('Control and level of institution and year',inplace=True)
# enroll_clean = enroll_clean.iloc[2:,]
enroll_clean.dropna('index',inplace=True)
```

```
enroll_clean.head(5)
```

```
# Map years by removing trailing periods
year_mapper = {yr: yr[0].replace('.','').strip() for yr in enroll_clean.index}
year_mapper
```

```
enroll_clean.rename(year_mapper, inplace=True)
enroll_clean.head(10)
```

In this table, we also have to take into account the different levels of students. In this table, we have **4 levels of institutions**. The first is the **total**, the second is **public 4-year institutions**, **private 4-year institutions**, and finally **public 2-year institutions**.

To deal with all of these, I will separate these four different levels into four different tables, as well as having different tables for full-time and part-time students.

```
# Total, all institutions table
total_df = enroll_clean.iloc[:32]
```

```
# Public 4-year institutions table
public_four_df = enroll_clean.iloc[32:43]
```

```
# Private 4-year institutions
priv_4_df = enroll_clean.iloc[43:49]
```

```
# Public 2-year institutions
public_2_df = enroll_clean.iloc[49:]
```

Now, I'm going to split the **total** table into two different dataframes. The first one will be for **full-time students**, while the second one will be for **part-time students**. This will make the overall table much cleaner.

```
# Making full-time dataframe
full_cols = [col for col in total_df.columns if ('Unnamed' in col[2]) or ('Part-time students' in
col[0])
            or ('Less than 20 hours .1' in col[2]) or ('Less than 20 hours .2' in col[2])
            or ('20 to 34 hours .1' in col[2]) or ('20 to 34 hours .2' in col[2])
            or ('35 or more hours .1' in col[2]) or ('35 or more hours .2' in col[2])]
full_time = total_df.drop(full_cols, axis=1)
```

```
# Making part-time dataframe
part_cols = [col for col in total_df.columns if ('Unnamed' in col[2]) or ('Full-time students' in
col[0])
            or ('Less than   20 hours .1' in col[2]) or ('Less than   20 hours .2' in col[2])
            or ('20 to 34 hours .1' in col[2]) or ('20 to 34 hours .2' in col[2])
            or ('35 or more hours .1' in col[2]) or ('35 or more hours .2' in col[2])]
part_time = total_df.drop(part_cols, axis=1)
```

```
# Viewing dataframes
full_time.head()
```

```
part_time.head()
```

Now, we will apply more formatting to the above dataframes to make them cleaner and more readable, also following column name guidelines.

```
# Full-time students dataframe
full_time = full_time.unstack().reset_index().drop('level_1',axis=1)
full_time.rename(columns = {'level_0':'enrollment_status',
                            'level_2':'hours_worked',
                            'Control and level of institution and year':'year',
                           0:'total'}, inplace=True)
full_time.head()
```

```
# Part-time students dataframe
part_time = part_time.unstack().reset_index().drop('level_1', axis=1)
part_time.rename(columns = {'level_0':'enrollment_status',
                            'level_2':'hours_worked',
                            'Control and level of institution and year':'year',
                           0:'total'}, inplace=True)
part_time.head()
```

Now the above tables are much cleaner. To visualize them better, I will develop some plots that will visualize the hours worked in total across full-time and part-time students.

```
# First, concat the two tables together to form the total table
enroll_df = pd.concat([full_time, part_time])
enroll_df
```

Visualize Level 3

```
# Plotting full-time students from total amount
import seaborn as sns
full_g = sns.relplot(data=full_time, x='year', y='total', col='enrollment_status',
hue='hours_worked')
full_g.set_xticklabels(rotation=50)
```

```
# Plotting part-time students from total amount
part_g = sns.relplot(data=part_time, x='year', y='total', col='enrollment_status',
hue='hours_worked')
part_g.set_xticklabels(rotation=50)
```

Summarize Level 3

Now, I'll compute some brief summary statistics regarding the data used in this submission.

```
# Calculating brief summary statistics for full-time students, a subset of the total amount of
students
full_time.describe()
```

```
# Calculating various statistics for part-time students, a subset from the total amount of students
part_time.agg({'total' : ['sum', 'min', 'max']})
```

The above summary statistic shows the **sum of the total hours** worked for part-time students, the **minimum number of hours** worked by part-time students, and the **maximum number of hours** worked by a part-time student.

By Brianna MacDonald