Project 4 Report

Andrew Becker

becke198

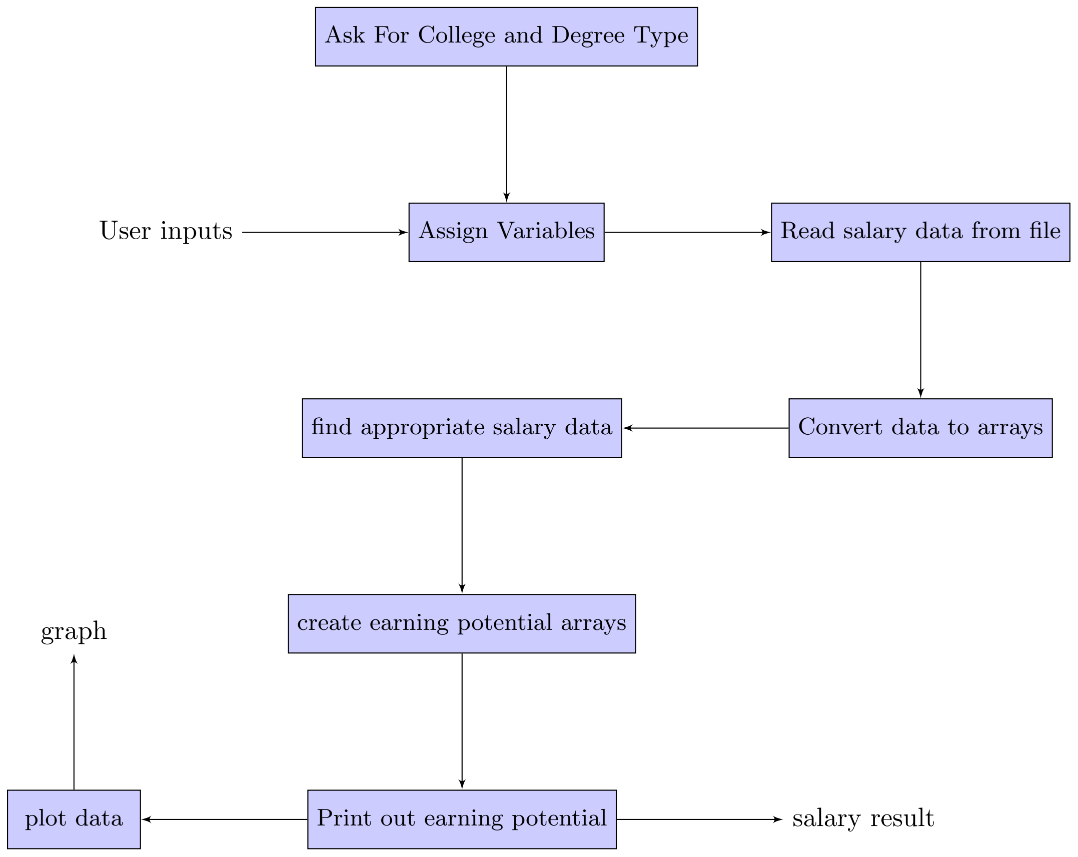
Section 001

Team 005

1 Introduction

I am creating a program that will calculate an engineer's median lifetime earnings. It will analyze data calculated from numerous sources such as College Scorecard, Glassdoor, etc., to provide the user with their college and median salary. I selected this theme, since it allows engineers or engineering graduates to gauge their earnings over a career to make better guided financial decisions. Many graduates have no idea about their major's predicted earnings, and lose valuable money paying off student debt, purchasing unnecessary items, and overspending on living expenses. I want to make their earnings clear so that they don't run into these mistakes.

1.1 Functional Block Diagram


Ask For College and Degree Type
User inputs
Assign Variables
Read salary data from file
find appropriate salary data
Convert data to arrays
graph
create earning potential arrays
plot data
Print out earning potential
salary result

2 Overview

The program first inputs the user's college from the top 100 engineering schools on US News. Then the program looks through the file of colleges and prints out each major, asking the user to select one. The program inputs the user's major and looks through the file again for the 4th and 5th columns representing the median wage 1 year after graduation and 5 years after graduation. With this data, the program calculates a vector of the potential earnings over the career. It outputs the data for the median wage 1 year and 5 years after graduation, along with a predicted career earnings plot.

3 User-defined Functions

3.1 `get_colleges(keyword)`

This function inputs a keyword the user enters and returns the colleges that match. If the user enters an indistinguishable keyword or nothing, all colleges are returned.

3.2 `get_majors_for_college(college)`

This function inputs the existing college from the user. It locates the college and returns all of the engineering majors that are located within that college, by indexing into the data array.

3.3 `get_salary_data(college, major)`

This function inputs the user's engineering college and major. It searches the data array and returns a tuple with the 1-year and 5-year median wages. If salary info is not found, the tuple `(None, None)` is returned.

3.4 `predict_lifetime_earnings(salary_1yr, salary_5yr, years=40)`

This function calculates the predicted lifetime earnings over a 40-year career using:

$$\text{salary}(y) = \begin{cases} S_1, & \text{if } y \leq 1 \\ S_1 + \dfrac{(S_5 - S_1)(y-1)}{4}, & \text{if } 1 < y \leq 5 \\ S_5 \cdot (1.03)^{y-5}, & \text{if } y > 5 \end{cases}$$

It begins with a fixed salary after 1 year, linearly increases to year 5, then compounds 3% annually for the remainder. [1]

4 User Manual

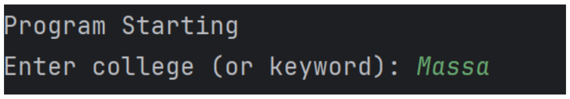
4.1 Start

To start the code, press the run button. The code will display: `Program Starting`

4.2 Inputs

An input prompt will appear:

```
Enter college (or keyword):
```

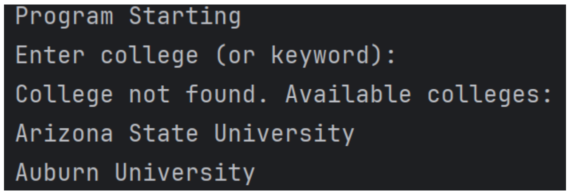


```
Program Starting
Enter college (or keyword): Massa
```

Figure 1: Enter College Prompt

If the keyword is invalid or empty, the entire list is displayed:

```
College not found.  Available colleges:
Arizona State University
...
```

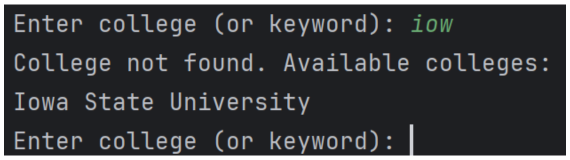


```
Program Starting
Enter college (or keyword):
College not found. Available colleges:
Arizona State University
Auburn University
```

Figure 2: No Colleges Entered

For example, entering `iow` results in:

```
College not found.  Available colleges:
Iowa State University
```

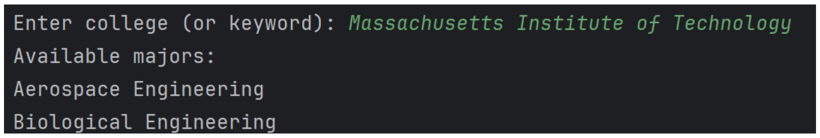


```
Enter college (or keyword): iow
College not found. Available colleges:
Iowa State University
Enter college (or keyword):
```

Figure 3: ’iow’ Entry

After selecting a college, users are asked to select a major:

```
Available majors:
Aerospace Engineering
Biomedical Engineering
...
Enter major:
```

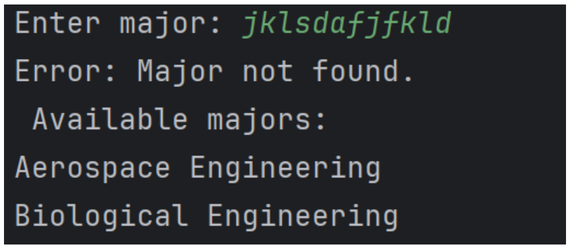


```
Enter college (or keyword): Massachusetts Institute of Technology
Available majors:
Aerospace Engineering
Biological Engineering
```

Figure 4: ’MIT’ Entry

Incorrect inputs causes the program to repeat the prompt until valid. For instance:

```
Enter major:  [jklsdafjfkld]
Error:  Major not found.
Available majors:
Aerospace Engineering
...
Enter major:
```



```
Enter major: jklsdafjfkld
Error: Major not found.
 Available majors:
Aerospace Engineering
Biological Engineering
```

Figure 5: 'jklsdafjfkld' Entry

4.3 Outputs

Example for *Chemical Engineering* at *Massachusetts Institute of Technology*:

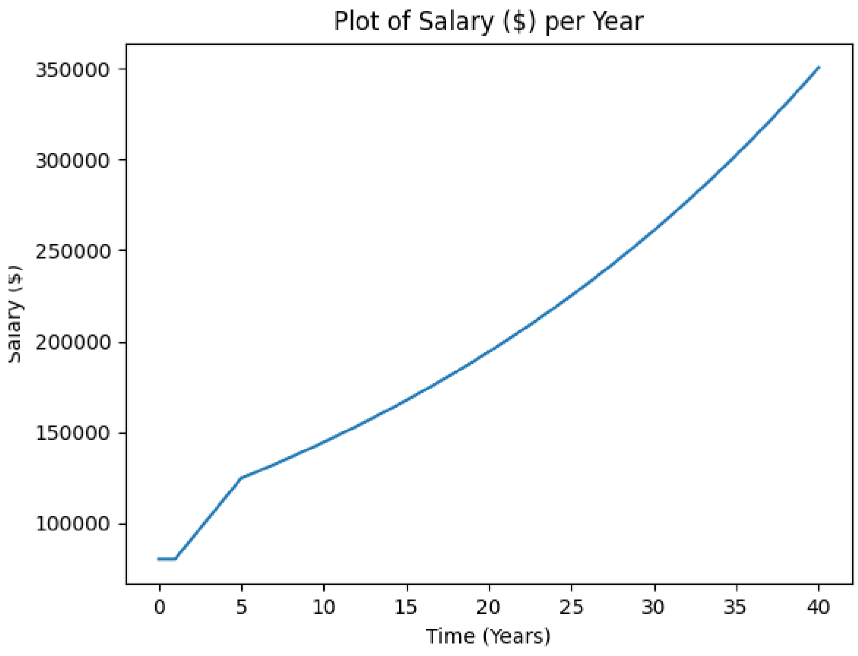


Figure 6: Predicted Salary Per Year Plot

```
The median salary 1 year after graduation for a Chemical Engineering major at
Massachusetts Institute of Technology is $80,139
```

```
The median salary 5 years after graduation for a Chemical Engineering major at
Massachusetts Institute of Technology is $124,650
```

If the file has missing or invalid data, the following error message is displayed:

```
Error:  Data is not available
```

5 An appendix (with code)

5.1 Main File

```
"""
==============================================================================
ENGR 130 Spring 2024 - Project 4


    I am creating a program that will calculate an engineer’s median lifetime earnings.
    It will analyze data calculated from numerous sources such as College Scorecard, Glassdoor, etc.,
    to provide the user with their college and median salary. I selected this theme, since it allows
    engineers or engineering graduates to gauge their earnings over a career to make better guided finan
     Many graduates have no idea about their major’s predicted earnings, and lose valuable money paying
      purchasing unnecessary items, and overspending on living expenses. I want to make their earnings c
      don’t run into these mistakes.


    Assignment:     Project 4
    Author:         Andrew Becker, becke198@purdue.edu

    Citations
    I found the try-except and .tolist() methods online on W3 schools.
    Used Chat-GPT to generate the file data from many online sources including glassdoor and college sco

==============================================================================
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import Project_4_SecondFunction

print("Program Starting")

# Load data (for grading purposes, file is engineering_salaries.csv)
data = pd.read_csv(r’engineering_salaries.csv’)

# Convert DataFrame to a 2D NumPy array
matrix = data.to_numpy()

#Creates the colleges
colleges = matrix[:,0].tolist()
majors = matrix[:, 1].tolist()
sals_1 = matrix[:, 2].tolist()
sals_5 = matrix[:, 3].tolist()


def get_colleges(keyword):
    """Returns the colleges for what is entered in the search bar"""
    keyword = keyword.lower().strip()
    new_colleges = []
    seen = []
    for college in colleges:
        college_str = str(college).lower()
        if keyword in college_str and college not in seen:
            new_colleges.append(college)
            seen.append(college)
    if new_colleges == []:
        for college in colleges:
            if college not in seen:
                new_colleges.append(college)
                seen.append(college)
        return sorted(new_colleges)
    return sorted(new_colleges)

def get_majors_for_college(college):
    """Returns a list of engineering majors for the given college."""
    matching_rows = data[data['University'] == college]
    majors_list = []
    for index in range(len(colleges)):
        if colleges[index] == college:
            majors_list.append(majors[index])

    #majors = sorted(list(set(matching_rows['Engineering Major'].astype(str))))
    return majors_list

def get_salary_data(college, major):
    """Returns 1-year and 5-year salaries for the given college and major."""
    for index in range(len(colleges)):
        if colleges[index] == college and majors[index] == major:
            try:
                sal_1 = float(sals_1[index])
                sal_5 = float(sals_5[index])
                return sal_1, sal_5
            except:
                return None, None
    return None, None
    #Error checking algorithm 1 returns None if salary info is not found from file




def main():

    #Get college input
    college = ""
    college = input("Enter college (or keyword): ")
    while college not in colleges:

        print(f"College not found. Available colleges:")
        for c in get_colleges(college):
            print(c)
        college = input("Enter college (or keyword): ")

    #get major input
    major = ""
    print(f"Available majors:")
    for m in get_majors_for_college(college):
        print(m)
    major = input("Enter major: ")


    #error checking algorithm
    while major not in get_majors_for_college(college):
        print(f"Error: Major not found.\n Available majors:")
        for m in get_majors_for_college(college):
            print(m)

        major = input("Enter major: ")

    sal_1,sal_5 = get_salary_data(college, major)


    x,y = Project_4_SecondFunction.predict_lifetime_earnings(sal_1, sal_5)

    #prints the output to console
    if len(x) == 0 and len(y) == 0:
        print(f"\n\nError: Data is not available")
    else:
        #makes sure the program does not crash. If it does, an error message is displayed.
        try:
            print(f" The median salary 1 year after graduation for a {major} major at {college} is ${in
            print(f" The median salary 5 years after graduation for a {major} major at {college} is ${in

            #plot starts
            plt.plot(x,y)
            plt.xlabel("Time (Years)")
            plt.ylabel("Salary ($)")
            plt.title("Plot of Salary ($) per Year")
            plt.show()
        except:
            print(f"\n\nError: Data is not available")




if __name__ == "__main__":
    main()
```

5.2 Second File

```
import numpy as np

def predict_lifetime_earnings(salary_1yr, salary_5yr, years=40):
    #predicts the salaries over a career, returns the range of years and the salary (for plotting)

    #If the salary_1yr or salary_5yr is empty, an empty array is returned.

    if salary_1yr == None or salary_5yr == None:
        return [], []
    #Creates the total amount of years for the x-axis
    years_range = np.arange(0, years + 1)
    # Creates the salaries for the y-axis
    salaries = []
    for year in years_range:
        if year <= 1:
            salary = salary_1yr
        elif year <= 5:
            salary = salary_1yr + (salary_5yr - salary_1yr) * (year - 1) / 4
        else:
            #A 3% increase in salaries is applied
            years_after_5 = year - 5
            salary = salary_5yr * (1 + 0.03) ** years_after_5
        salaries.append(salary)

    return years_range, salaries
```

References

1 https://www.oysterhr.com/library/how-to-calculate-raise-percentage?