

**ANL252**

**Python for Data Analytics**

# **Tutor-Marked Assignment**

**July 2023 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Tan Mei Yi Rachel** | **Q2081315** |

**Tutorial Group: ­­­­­­­­­­­­ TV 03**

**Instructor’s Name: Mr. Munish Kumar**

**Submission Date: 12/09/2023**

**Question 1**

**(a)**

Plagiarism in coding happens because:

* Online solutions are readily available. The availability of online code repositories like GitHub and Stack Overflow allows programmers to store their codes while also providing discussion forums (Calonia, 2023), which may tempt students to use these platforms to blindly copy codes that are not their own directly into their assignments.
* Students have inadequate knowledge of academic policies. Students may be unaware of how to cite and paraphrase sources appropriately and hence may unintentionally commit coding plagiarism (Grammarly, 2022).

Students can avoid this issue by:

* Providing sufficient citation. Once a certain information or idea is taken from a source that is available online, adequate citation should be given to the original author (Calonia, 2023).
* Attempting to write their own code based on knowledge attained from study materials first, which promotes originality. Once an error occurs, students can seek online resources to investigate what is wrong and improve on it. (*All you need to know about plagiarism in coding,* 2021).
* Making use of a broad range of sources. Doing so would enable one to not be limited to codes from a single source which can increase the probability of plagiarism (Hossain, 2021).

**(b)**

L, QualityPoints, TotalQualityPoints, TotalCourseUnits, x, i = [], [], 0, 0, "", 1

try:

while x != "N":

course\_code = input("Course Code(e.g. FSC111)>> ")

course\_mark = int(input("Course Mark/Score(e.g. 70)>> "))

course\_units = int(input("Course Credit Units(e.g. 3)>> "))

if 0 <= course\_mark <= 39:

grade\_point = 0

elif 40 <= course\_mark <= 44:

grade\_point = 1

elif 45 <= course\_mark <= 49:

grade\_point = 2

elif 50 <= course\_mark <= 59:

grade\_point = 3

elif 60 <= course\_mark <= 69:

grade\_point = 4

elif 70 <= course\_mark <= 100:

grade\_point = 5

L += [[course\_code, course\_mark, course\_units, grade\_point]]

TotalCourseUnits += course\_units

TotalQualityPoints += (course\_units \* grade\_point)

QualityPoints.append(f"{course\_units \* grade\_point}")

x = input("Continue (Y or N): ")

if x == 'N':

break

elif x == 'Y':

pass

CGPA = round(TotalQualityPoints/TotalCourseUnits, 2)

print(f"CGPA: {CGPA}")

except ValueError or TypeError as e:

print(f"Error message: {e}")

The above code was adapted from GitHub, by Emmanuel C. Jemeni.

The main objective of this code is to calculate a student’s university Grade Point Average (GPA). It requires students to first key in their course code, course credit units and marks, which will then be assigned a grade point value. After the collection of the student’s information, a formula to calculate the total GPA (Total grade points divided by total credit units) will be used to print the corresponding GPA of the student (Jemeni, 2021).

As a break down, the first line initialises several variables using tuples. The try-except block is then implemented for error-handling. For example, if I key in a string in course\_mark and course\_units, then an error message would occur. A while loop is used to show that as long as the user does not enter “N” when asked to continue, the code in the while loop will continue. If ”N” is entered, the codes within the loop will stop and the corresponding GPA will be calculated and displayed. Within the loops, an if-elif condition is used. This allocates a grade point value to the corresponding mark that the student enters to calculate the GPA.

**(c)**

# Mapping of Letter grades to corresponding grade point values, and degree classifications to GPA ranges based on SUSS’ system through the use of dictionary

grade\_point\_value = {"A+": 5.00, "A": 5.00, "A-": 4.50, "B+": 4.00, "B": 3.50, "B-": 3.00, "C+": 2.50, "C": 2.00, "D+": 1.50, "D": 1.00, "F": 0.00}

degree\_classifications = {"First Class": {"min\_gpa": 4.50, "max\_gpa": 5.00},"Second Upper Class": {"min\_gpa": 4.00, "max\_gpa": 4.49},"Second Lower Class": {"min\_gpa": 3.50, "max\_gpa": 3.99},"Third Class": {"min\_gpa": 3.00, "max\_gpa": 3.49},"Pass": {"min\_gpa": 2.00, "max\_gpa": 2.99},"Fail": {"min\_gpa": 0.00, "max\_gpa": 1.99}}

total\_no\_of\_credits = sum\_of\_individual\_gpv = 0

valid\_credits = (2.5,5)

# While-loops to input module information

while True:

module\_code = input("Enter the module code (or 'END' if finished) : ")

if module\_code == 'END':

break

while True:

no\_of\_credits = float(input(f"Enter the number of credits for {module\_code}: "))

if no\_of\_credits in valid\_credits:

break

else:

print("Invalid. Please indicate either 2.5 or 5 credits")

while True:

grade = input(f"What is your grade for {module\_code}: ")

if grade in grade\_point\_value:

break

else:

print("Invalid grade. Please try again.")

# To calculate the grade point value for a module

total\_gpv = no\_of\_credits \* grade\_point\_value[grade]

# To calculate the total credits for the modules that a student takes in a semester

total\_no\_of\_credits +=no\_of\_credits

sum\_of\_individual\_gpv += total\_gpv

sgpa = sum\_of\_individual\_gpv / total\_no\_of\_credits

print (f"Semester GPA: {sgpa:}")

# To allocate the corresponding value (Semester GPA) to a degree classification from the dictionary

for key, val in degree\_classifications.items():

if val["min\_gpa"] <= sgpa <= val["max\_gpa"]:

print(f"Degree Classification: {key}")

Output:

Enter the module code (or 'END' if that is all): ACC201

Enter the number of credits for ACC201: 5

What is your grade for ACC201: B-

Enter the module code (or 'END' if that is all): BUS100

Enter the number of credits for BUS100: 5

What is your grade for BUS100: B-

Enter the module code (or 'END' if that is all): BUS105

Enter the number of credits for BUS105: 5

What is your grade for BUS105: A-

Enter the module code (or 'END' if that is all): BUS107

Enter the number of credits for BUS107: 5

What is your grade for BUS107: B+

Enter the module code (or 'END' if that is all): SCO101

Enter the number of credits for SCO101: 2.5

What is your grade for SCO101: B+

Enter the module code (or 'END' if that is all): SCO103

Enter the number of credits for SCO103: 2.5

What is your grade for SCO103: B-

Enter the course code (or 'END' if that is all): END

Semester GPA: 3.6

Degree Classification: Second Lower Class

The code was modified and improved with references from (Olumese, 2022) and (Wu & Zhu, 2023).

The rationale behind rewriting the code is that it

* Prevents the likelihood of plagiarism. Quite similar to the concept of paraphrasing, rewriting a code encourages modification such as implementing different variable names and changing the order of the codes which subsequently ensures that the code is no longer entirely identical from the original code that was sourced. For example, I have tried to implement the use of dictionaries which was not done so in the original code.
* Increases understanding of Python concepts. The act of rewriting a code provides us with a learning opportunity by deconstructing codes and it is only through this process that we will be able to better understand the logics behind the use of various codes (Atwood, 2006). In the process of rewriting the code, my aim was to best match them to the ones that have been taught to brush up on my fundamental skills rather than delving straight into complex and sophisticated codes which are widely used online.
* Improves the functionality of the code. The process of rewriting allows us to add new features and enhancements along the way and increase the code’s efficiency (Sharovar, 2022). The process enables us to leverage on opportunities to improve the efficiency through simplifying codes which allows us to save time from dissecting complex codes if a syntax error occurs. It also allows us to incorporate new features or functions that are more applicable to us. For example, I modified the codes that are in line with SUSS’ grading system as well as the credit units (either 2.5 or 5).

**Question 2**

Generally, in terms of maintainability and reliability, the python code could be improved on its error handling aspects.

Maintainability

When I enter a product name that is not in the list, the current code prints an error message and exits the loop, and the entire code just stops there. To improve, if-else conditions can be used to urge the user to re-enter the product if the result is false (the product is not in products).

Reliability

The code would continue even if the price I entered for the product was a string rather than a number. To address this issue, the 'try-except' syntax should be used to handle it. The ValueError exception can be used to ensure that if an input is of not a numerical value,, the user is prompted to re-enter their input (Wu & Zhu, 2023).

Readability

Comments can be used to increase code readability by ensuring that code functionality and intent are explained (Punia, 2023). This is due to the fact that a new user who has only seen the code will not be able to fully understand what it does with a single glance.

Taking above-mentioned points into consideration, the improved code is as shown:

# list down the variables

products = ['laptop', 'mouse', 'webcam', 'keyboard', 'speaker']

query = 'yes'

updated\_items = []

print(f'We have a list of products here: {products}.')

# While-loop for collection of all necessary user inputs

while query == 'yes':

item = str(input("Hello! What do you want to buy?"))

if item in products:

# While-loop to collect input prices until a valid numerical price is provided

while True:

try:

price\_of\_item = float(input(f"How much is the {item} (in SGD)?"))

entered\_input = [item, price\_of\_item]

updated\_items.append(entered\_input)

break

except ValueError:

# Display error message if a valid price is not in numbers

print("Please enter a valid price in numbers.")

else:

# Display error message is product that is not within list is entered

print(f'{item} is not a valid product. Please try again.')

continue

# Prompt user if there is any more products to add to the shopping list

query = str(input("Would you like to continue? (yes/no)"))

print(f'This is our updated shopping list: {updated\_items}')

**References**

Wu, K. Y., & Zhu, S. (2023). ANL252 Python for data analytics. Singapore University of Social Sciences.

A*ll you need to know about plagiarism in coding*. Copyleaks. (2021, July 9). Retrieved September 8, 2023 from https://copyleaks.com/blog/all-you-need-to-know-about-plagiarism-in-coding

Atwood, J. (2006, September 18). *When understanding means rewriting*. Coding Horror. Retrieved September 11, 2023 from https://blog.codinghorror.com/when-understanding-means-rewriting/

Hossain, K. M. E. (2021, June 12). *Plagiarism by students - consequences and how to avoid it*. Plagiarism guide for students: Consequences & how to avoid. Retrieved September 8, 2023 from https://www.totalstudentcare.com/guides/en/plagiarism-by-students-consequences-and-how-to-avoid-it/

Olumese, F. (2022, June 15). *Creating a GPA calculator using Python*. The Coding Process. Retrieved September 12, 2023 from <https://thecodingprocess.hashnode.dev/creating-a-gpa-calculator-using-python>

Grammarly. (2022, September 12). *TIPS to identify and help students avoid unintentional plagiarism*. Retrieved September 8, 2023 from https://www.grammarly.com/blog/unintentional-student-plagiarism/

Sharovar, E. (2022, December 17). *Code rewriting: When and why*. Waverley. Retrieved September 12, 2023 from https://waverleysoftware.com/blog/code-rewriting-when-and-why/

Calonia, J. (2023, April 25). *How to avoid plagiarism*. Grammarly. Retrieved September 8, 2023 from https://www.grammarly.com/blog/5-most-effective-methods-for-avoiding-plagiarism/

Punia, R. (2023, May 27). *8 must-know points to improve code readability*. Medium. Retrieved September 12, 2023 from https://medium.com/@mycodingmantras/8-must-know-points-to-improve-code-readability-824ca4d35e7e#:~:text=Readability%20and%20Code%20Aesthetics,such%20as%20functions%20or%20loops

Jemeni, E. C. (2021), GitHub repository. Retrieved September 11, 2023 from <https://github.com/Jemeni11/GPA-Calculator-CLI/blob/main/main.py>