

**ANL252 (Online)**

**Python for Data Analytics**

# **Tutor-Marked Assignment**

**July 2023 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Villanueva Van Hally Borines** | **H2072822** |

**Tutorial Group: ­­­­­­­­­­­­ T 03**

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

**Submission Date: 15/09/2023**

Question 1a:  
  
Reasons as to why plagiarism in coding happens:

1. Unfamiliarity with the code - If a problem requires a user to input a code, unfamiliarity or lack of knowledge with regards to coding may result in the user having to plagiarise instead.
2. Time Management - Students are always packed with schoolwork such as quizzes, essays, assignments, and projects. If due to a poorly planned schedule, a student might have no time to do proper research/readout of the topic. Thus, resulting in copying codes from the internet or from their friends.

How to avoid plagiarism:

1. Cite sources - Basic rules to avoid plagiarism as emphasised by the school is to source out any codes or snippets that are taken from other sources.
2. Improve knowledge - To tackle unfamiliarity with coding, users should improve their knowledge of the subject to be able to better code. Thus decreasing the chances of plagiarising
3. Better time management - Emphasis on having better time management would ensure that students can better plan their schedule and allocate more time to subjects such as coding so that they can clarify with their peers/instructors if they have concepts that they do not understand.

Question 1b:

Code (Wu et al., 2023):

i=0

while i<3:

    i=i+1

    name=input("What is your Name?")

    score=int(input(f"Hello {name}. What is your score?"))

    if score<40:

        grade="F"

    elif score>=40 and score<50:

            grade="D"

    elif score>=50 and score<60:

            grade="C"

    elif score>=60 and score<80:

            grade="B"

    elif score>80 and score<100:

            grade="A"

    print(f"{name}, your grade is{grade}.")

Output:

What is your Name? Van

Hello Van. What is your score? 77

Van, your grade is B.

What is your Name? Jason

Hello Jason. What is your score? 65

Jason, your grade is B.

What is your Name? James

Hello James. What is your score? 55

James, your grade is C.

The code above was taken from the ANL252 study guide (Wu et al., 2023). The code prompts the user to enter their name and state their score. Their score will then be checked based on the set range. The code will then prompt the user for a name and his/her grade. The steps above are achieved with the help of a while-loop. We first input a counter variable i with a value of 0. The condition for the loop was that while the counter variable is less than 3, the counter variable i would then increase by 1 with reference to “i=i+1” in the code.

The code will proceed to ask the first user, Van to input his name followed by his score. When reaching James’s turn, the counter variable i would have already been in its 2nd reiteration. Therefore, the code would stop prompting other users as the condition of i<3 will no longer be true after James has keyed in the input.

Question 1c:

Code:

i = 0

while i < 3:

    i += 1

    name = input("What is your Name? ")

    score = int(input(f"Hello {name}. What is your score? "))

    if score < 40:

        grade = "F"

    elif 40 <= score < 50:

        grade = "D"

    elif 50 <= score < 60:

        grade = "C"

    elif 60 <= score < 80:

        grade = "B"

    elif 80 <= score <= 100:

        grade = "A"

    else:

        grade = "Invalid input, please enter a score between 0 and 100."

    print(f"{name}, your grade is {grade}.")

Output:  
  
What is your Name? Bruce

Hello Bruce. What is your score? 75

Bruce, your grade is B.

What is your Name? Bobby

Hello Bobby. What is your score? 95

Bobby, your grade is A.

What is your Name? Charlotte

Hello Charlotte. What is your score? 33

Charlie, your grade is F.

Reasons for rewriting the code:

* Simplified conditional statements: Originally, the elif statements were coded “elif score>=40 and score<50:”. This code has been simplified to “elif 40 <= score < 50:” instead. Doing so minimizes the risk of accidentally having an overlap or conflicts in conditions stated which can lead to minor syntax errors. The simplification also makes it easier for developers to read and understand the conditions set. If one were to make changes or updates to the conditions for the grading system, it would be much easier as compared to the previous code.
* Mitigating errors: In a scenario that a user keyed in 0.5 or 150 in the score box, the addition of “else” statement prompts the user to provide a score within the specified range of 1 to 100. This prevents the code from potential crashes when processing invalid inputs without having the “else” statement to check through.
* Code readability: The new code is much more readable as compared to the old code with the consistent indentation and additional spaces around operators. This helps the developers visually as the codes become neater, it would be easier for them to debug and will be able to easily spot syntax errors or trace the flow of execution.

**Question 2:**

Modified Code:

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

updated\_items = []

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

while True:

    item = input("Hello! What do you want to buy? ").lower()  # Convert input to lowercase for case-insensitive comparison

    if item not in products:

        print('Wrong product! Please try again.')

    else:

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

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

        updated\_items.append(entered\_input)

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

    if query != 'yes':

        break

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

After running through the Appendix 1 code, it could be changed to the code above. When an invalid product was entered to the code, the code breaks out of the loop and results in the statement below:

We have a list of products here: ['laptop', 'mouse', 'webcam', 'keyboard', 'speaker'].

Hello! What do you want to buy? phone

Wrong product! Please try again.

This is our updated shopping list: []

Although it informs the user that it has keyed in a wrong product, it does not allow the user to go and key in again. Thus, adding an input validation is important as it helps to handle incorrect inputs. Instead of breaking through the loop, displaying an invalid message and allowing the user to re-enter the product name improves the code’s reliability. This also ensures that invalid input are handled correctly.

As long as ‘query’ is equal to yes, the user would be able to add as many items in the product list, making the code more reliable.

The function ‘.lower()’ also helps in making the code readable as when the keyed input is compared with the product list, it would be case-insensitive. Therefore treatment for either ‘Laptop’ or ‘laptop would be the same.

Reference:

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