

**ANL252: Python for Data Analytics**

**Tutor-Marked Assignment: July 2023 Presentation**

Submitted by:

|  |  |
| --- | --- |
| Goh Jun Jie | K2210106 |

BSc Logistics and Supply Chain Management, Singapore University of Social Sciences  
  
  
Dr Munish Kumar  
  
  
September 15, 2023

**Question 1a**

According to (Lee, 2020), nature of the coding academic field makes it particularly prone to plagiarism. Two main reasons contributing to occurrence of plagiarism in coding are identified:

Solution Availability

* Easily accessible “open source” codes/softwares e.g. Github, Stack Overflow
* Therefore students are enticed to misuse these resources

Ambiguity of Collaboration

* Culture of widespread cooperation between programmers promoted in software industry
* In contrast, originality of students’ programming academic coursework necessary to assess their coding skills
* Unclear definition regarding what constitutes plagiarism or collusion, and not collaboration
* Collusion refers to multiple parties jointly engaging in academic dishonesty, whereas collaboration refers to working together to comprehend and tackle a logical issue.

Besides defining and communicating to students the difference between collaboration and cheating, according to (Ngo, 2016, p. 3), methods to combat coding plagiarism include designing programming assignments with code skeleton, which confers multiple benefits:

* Provides guidance and scaffolding to students new or weak in coding, hence encouraging students to work individually, and discourage them from seeking to plagiarise.
* Code skeletons have pre-existing data structures and variable names of code skeletons, therefore solutions cannot be readily found online, but students can learn from similar solutions and apply techniques learned to the code skeleton.

(200 words)

**Question 1b**

This piece of python code I have created is aimed at completing the task of reporting the respective scores of each subject that a student in school has failed, with the use of an input function requesting the student to input the subject of a subject score he/she desires to check, which will search the dictionary indexed by keys denoting the subject name, and a combination of if-conditional statements to produce values which are the scores the student has attained in the last semester. However, if the student inputs a different subject name not within the dictionary, an output printed denotes an error or invalidity to be feedbacked to the student so that he/she can try again.

Subsequently, I have used a combination of if-conditional statements, try-blocks and exceptions to complete the task of target setting for the failed subject. Finally, the output is tailored according to the target score value input by the student, where the student is encouraged to pass the subject with a score of more than 50 in the next semester. Meanwhile, negative scores, scores of a non-numerical value or exceeding full marks will also print an error or invalidity to be feedbacked to the student.

(200 words)

print("Unfortunately you have failed 3 subjects in the last semester: Math, Science and Geography")

failed\_subjects = {"Math":47, "Science":34, "Geography":28}

score = input("Which subject score would you like to check?")

if score == "Math":

print(failed\_subjects["Math"])

if score == "Science":

print(failed\_subjects["Science"])

if score == "Geography":

print(failed\_subjects["Geography"])

if score in failed\_subjects:

try:

goal = int(input("How much do you aim to score for this subject in the next semester?"))

if goal>=50 and goal<100:

print(f"Your target score for the next semester has been recorded. All the best! You can do it!")

if goal<50 and goal>0:

print("You can do better!")

if goal>100 or goal<0:

print("Invalid score entered. Please try again.")

except ValueError:

print("Invalid score entered. Please try again.")

if score not in failed\_subjects:

print("Invalid subject entered. Please try again.")

**Question 1c**

In order to avoid coding plagiarism, the code in (b) has been rewritten to:

* Change the name of subjects, scores and scenario to 2 failed subjects, 1 passed subject
* Changed from 1 dictionary list to 2 individual dictionary lists, one consisting of failed subjects and the other consisting of passed subjects.
* The 2 dictionary lists are merged and printed such that all grades are made known to the student at once rather than having to check the grades of each subject individually one by one.
* For target setting, the if-conditional statement “if goal<50 and goal>0:” has been changed to “else:”, to reflect the if-else-block, such that any values that are still a fail grade (outside of the passing score) will print an encouraging message for students to do better and aim for a pass.
* Similarly, the if-conditional statement “if score not in failed\_subjects:” has been changed to “else:”, to reflect the if-else-block such that target\_setting not in the list failed\_subjects will produce an error message.
* In this case, in the event the student has already passed a subject, the target-setting exercise is not necessary and skips to print another encouraging message instead, for the student to seek continuous improvement and keep up the good work.

This is reflective of learning from the solution in (b) and applying the techniques learned to a new form in (c).

print("You passed one subject and failed two subjects in the last semester")

failed\_subjects = {"English":33, "History":38}

passed\_subjects = {"Physics":70}

user\_qn = input("Would you like to check your grades?")

if user\_qn == 'yes':

print(list(failed\_subjects.items()|passed\_subjects.items()))

else:

print("Your grades will be sent by e-mail instead.")

target\_setting = input("Please choose a subject to complete target setting for")

if target\_setting in failed\_subjects:

try:

goal = int(input("How much do you aim to score for this subject in the next semester?"))

if goal>=50 and goal<100:

print(f"Your target score of {goal} for the next semester has been recorded. All the best! You can do it!")

else:

print("You can do better")

if goal>100 or goal<0:

print("Invalid score entered. Please try again.")

except ValueError:

print("Invalid score entered. Please try again.")

else:

print("Invalid subject entered")

if target\_setting in passed\_subjects:

print("You have entered a subject you already passed. Seek continuous improvement and keep up the good work!")

**Question 2**

The code in Appendix 1 can be improved such that there are not so many unnecessary parentheses, particularly in line 11, the parentheses around ‘in SGD’ can be removed because there is no important function delivered by adding the additional parentheses, other than adding to the complexity and difficulty of reading the code. Furthermore, the question remains comprehensible without the parentheses and the string remains working.

Similarly in line 14, the parentheses around ‘yes/no’ can be removed because it serves no purpose other than obstructing ease of reading and there are too many parentheses in the string.

Another way the code can be improved is to add an additional markdown cell on top with an additional header or the question statement so that it is clear what the code is about.

Lastly, instead of a break on line 10, the code would be more functional by replacing it with ‘else’, therefore forming an if-else-block. On line 8, in the event that a wrong product is input, there is a loopback to line 6 item = str(input("Hello! What do you want to buy?")) after print(f'Wrong product! Please try again.'). Instead of having the code end abruptly with an empty updated shopping list, the user is immediately looped back to the initial question to start again.

Improved version of the code in Appendix 1:

#This is a shopping list  
products = ['laptop','mouse','webcam','keyboard','speaker']

query = 'yes'

updated\_items = []

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

while query == 'yes':

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

if item not in products:

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

else:

price\_of\_item = input("How much is it in SGD?")

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

updated\_items.append(entered\_input)

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

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

# References

Lee, C. (28 July, 2020). *What is Programming Plagiarism? Why Is It on the Rise?* Retrieved from Turnitin: https://www.turnitin.com/blog/what-is-programming-plagiarism-why-is-it-on-the-rise

Ngo, M. N. (2016). Eliminating Plagiarism in Programming Courses through Assessment Design. *International Journal of Information and Education Technology*, Vol. 6, No. 11.