Programming Assignment

Introduction to Information Technology 4478

Note 1: Only use IDLE (with **Python 3.6 or above**) to write this assignment! For the purpose of this assignment, you are not allowed to use any other Python IDE and you are not allowed to use any other version of the Python.

Note 2: Familiarise yourself with the Python Style Guide at the end of this document.

Note 3: All cases of plagiarism will be pursued! If in doubt, consult the Student Academic Integrity Policy

Note 4: This work must be your original work, I **may request to meet any student to discuss their submission**.

This assignment will test a student's knowledge of, and skills in writing application software for a task, understanding the business rules of a particular problem, coding these in a computer program, developing a graphical user interface, and reading data from a text file on disk. As such, the assignment requires you to integrate and synthesise what you have learnt so far in this unit, in order to design and create a correct working solution.

The context of this programming assignment is the **Chook Food Company**.

For this assignment, students will use the Python programming language (version 3.x) and development will be on the IDLE platform as practised in the computer lab classes. This assignment consists of three stages.

- **Stage 1**: A simple Python program using an interactive text-based menu (no GUI)
- **Stage 2**: The same (stage 1) but wrapped in a GUI
- **Stage 3**: Input comes from a text file – read only once, then information stored in a suitable data structure. (Advanced GUI)

4478 IIT
Part A – Stage 1 (25 marks)
Part B – Stage 2 (10 marks)
Part C – Stage 3 (15 marks)

Chook Food Company App

Each type of chook food comes in 10 Kg bags and 50 Kg bags. Each chook food has a name, the cost of a 10 Kg bag and the cost of a 50 Kg bag.

Stage 1:

Pellets cost $22.75 for a 10 Kg bag and $100 for a 50 Kg bag. ***Mash*** costs $20.50 for a 10 Kg bag and $90 for a 50 Kg bag. ***Enhanced*** food costs $25.50 for a 10 Kg bag and $125.50 for a 50 Kg bag. These details can be hardcoded in your program (*you may use a nested list to store these values*).

For this stage, your program should know about the three types of food above.

In Stage 1, you will be developing a Python program **without a GUI**. Input and output are via the Python shell. Write a Python program that displays a text based interactive menu to allow the users to:

- Request displaying costs of the chook food for 10Kg or 50 Kg bags.
- Perform chook food shopping as follows:

  1- Allow the user to select the type of food they want to purchase from the menu.

  2- Allow the user to select the price calculation method according to:

  2.1 The Kilograms option, the user enters only the number of 10 kilograms increments, after that the price is calculated by using the cheapest way to buy exactly that weight of the chook food selected. The answer will be the number of 10 Kg bags or the number of 50 Kg bags and the total cost (**see the sample test document**).

  2.2 The Bags options, the user needs to enter the number of 10kg bags and the number of 50kg bags and the price will be calculated according to these two choices (**see the sample test document**)

Notes:

- You should use functions for each task in the program, for example a *printMenu* function to print a menu and *calculcateCost* to calculate the cost of chook food etc.
- Your program should have a *main()* function to appropriately direct the program.

What the tutors will be looking for

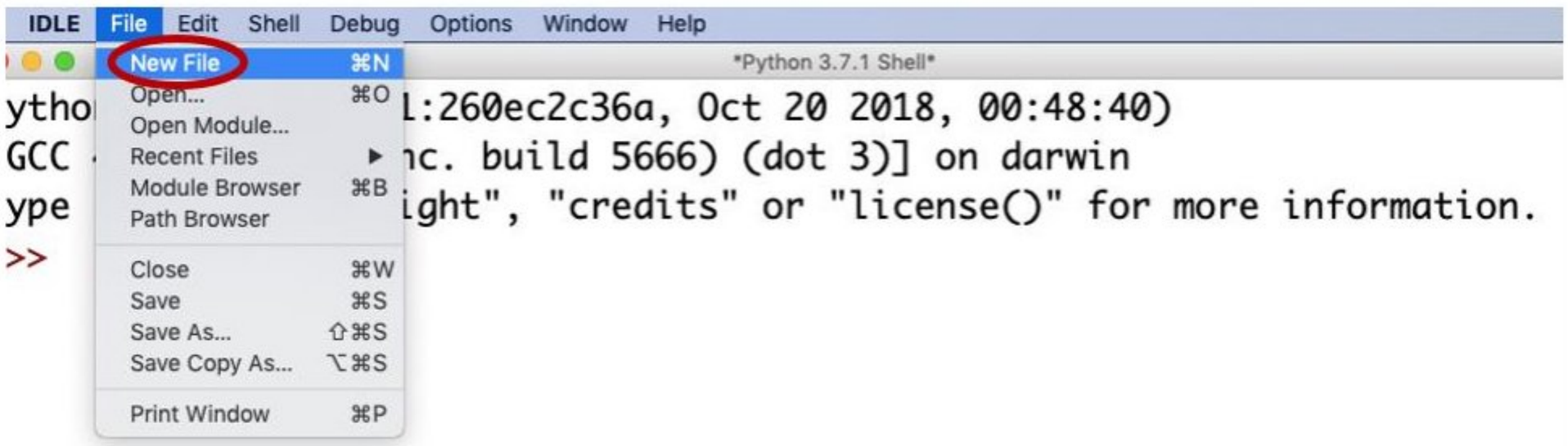
The tutor's instructions include

- Constants vs literals. Using constants is important for the ease of maintenance. Not using constants will result in lower marks. For example, consider constants for the price of chook food for 10Kg bag and 50 Kg bag.
- Program code layout. Separate blocks of code by a blank line. Use comments.
- Program is written using well-defined functions including a main() function to direct the program.
- A comment is not an essay. Comments are important for the maintenance of a program and should contain enough details but keep them concise. Do not comment every single line.
- The program must have a prologue. **Check style against the Python style guide attached below.**
- Good names for your variables and constants. **Check style against the Python style guide attached below.**
- Does the program work correctly?

Please refer to the Programming Assignment Marking Rubric for details on marks.

Step-by-Step Guide for Stage 1

1. Think about your strategy for how you will display the options for user to choose from, how you will calculate the total cost of chook food and how you would calculate the cheapest way to buy the chook food with given weight. Think about writing simple functions for the repetitive calculations.
2. Create a new Python file and call it *ProgAsgStage1*



3. In the Stage1 class (file `ProgAsgStage1.py`), you may put all code for the user interaction and the calculation into the main method. (You might still need to define global variables and constants outside the main method at the top of the editor window). You might like to use nested lists to hold the chook food type and price for 10kg bag and 50 kg bag.

4. You will need to write a function to print a menu to the user, **below a sample code as an example**:

```
##Printing the menu.
def main():
    menu()

##Main navigation tool. It can provide brand info, go to the shopping process or exit program
def menu():
    print("Welcome to Chook Food Selector!")
    print("1. Chicken food brand information")
    print("2. Chicken food shopping")
    print("3. Exit\n")
    sChoice = input("Enter a number from the menu above to proceed: ")
```

5. Now add the code that implements your strategy to display the different type of chook food **TEST**
6. Add the code to calculate the cost of 10 Kg bags and 50kg bags when a chook food is selected **TEST**
7. Add the code to calculate the cheapest way to buy the chook food for the weight entered **TEST**
8. Finally, add print statements to print the output to the user.
9. Test your implementation with the test cases, use the document (sampleTest - stage 1) as a guide. Please note, you need to do your own test with values different from the test cases in the document (**Check the submission section in the last page**).

Stage 2:

In stage1, the user input and output are not very satisfactory from the human computer interaction and usability point of view, your task in this stage is to design and implement a Graphical User interface (GUI) using buttons and labels for the chook food company that provides an easy to use interface. **The tasks are like stage 1, except you need to use GUI to interact with user**.

This GUI application must allow a user to input the various data elements. To this end, the user can

- input the data using an entry widget
- click on a *Calculate* button that starts the calculation of the chook food cost, and
- an *Exit* or *Quit* button to properly close the program.

Use the same code for the calculations from Stage 1. Reusing code is an important part of software development.

You have a great degree of freedom in what GUI elements you choose and how you would like to design the layout of your GUI. What matters is the functionality of the design and that the user can input the required data in a sensible fashion.

What the tutors will be looking for

The tutor's instructions include

- Constants vs literals. Using constants is important for the ease of maintenance. Not using constants will result in lower marks. For example, in addition to Stage 1, consider constants for the default width and height of the labels, buttons and entry widgets to easily achieve a uniform look.
- GUI Design. Is it simple to use and easy to understand? Are all required components there? Grid layout is used to place widgets.
- Program is written using well-defined functions including a *main()* function to direct the program.
- Program code layout. Separate blocks of code by a blank line. Use comments.
- Separate GUI functionality from calculations. Students should use a separate method for the calculations. It is good practice and follows good programming style to separate GUI code from other code.
- A comment is not an essay. Comments are important for the maintenance of a program and should contain enough details but keep them concise. Don't comment every single line.
- The program must have a prologue. **Check style against the Python style guide attached below.**
- Good names for your variables and constants. **Check style against the Python style guide attached below.**
- Does the program work correctly?

Please refer to the Programming Assignment Marking Rubric for details on marks.

Stage 3:

Complete the stage 1 & stage2 plus: The program will determine the list of chook food at run time. Do this by reading this data from the file *chookfood.txt* and data will be stored in a nested list. *chookfood.txt* is available on IIT Canvas site.

The file *chookfood.txt* contains the chook food data for several chook food type. Each line consists of chook food type as the first element, followed by comma and the price of 10Kg bag and price of 50Kg bag. For example: `Mash, 20.50,90.00`

You will need to copy `chookfood.txt` into the same directory as your Python files. When marking, the details in the file may change, but the structure will not.

Your task will be to:

1- Add code to your program that reads the chook food data from a text file.
2- Put these into appropriate storage in memory (I suggest you use a nested list) and display the details to user when requested. Please note that text file contains more food types than given in stage 1 and 2.
3- A List of different chook food types should be displayed using listbox widget to the user.
4- When the user clicks any food from the list, the price of that food must be displayed for 10kg and 50kg.
5- You should also use the advanced Python GUI components to **create a graph to compare prices of different chook food type by different weights (10kg or 50kg) for all foods.**

Notes:

- Food type should be displayed in the ascending order of type to the user.
- The text file should only be read once. You probably want to do that at the start of the program. Consider how you would do that.
- Write a Function called *readFile* to read the data from the text file

What the tutors will be looking for

The tutor's instructions include (apart from the usual such as good variable names, prologue / comments, code layout, …)

- Use of constants. Think about where you could use constants here.
- The text file should only be read once. You probably want to do that at the start of the program. Consider how you would do that.
- Program is written using well-defined functions including a main() function to direct the program.
- Correct display of chook food information based on the details in the text file.
- There are various ways you can store the data when reading the file. You could use a nested list.

Python Style Guide

General

Your programs should be

- Simple
- Easy to read and understand
- Well structured
- Easy to maintain

Simple programs are just that. Avoid convoluted logic, nested if-statements and loops, duplicating execution (such as reading files multiple times), repeated code, being "clever".

Programs can be made easier to read by following the "Layout" and "Comments" guidelines below.

Well-structured code uses functions to help tame complexity.

If programs are simple, easy to understand and well-structured they will be easy to maintain. Easily maintained programs will also use constants rather than literals and use built-in functions and types.

Layout

The IDLE text editor does a good job of automatically laying out your program and it also helps with the indentation of the code inside an/a if-statements, loops and functions. It is very important in Python that you indent your code correctly.

White space

Use white space freely:

- A blank line after declarations
- 2 blank lines before each function declaration

Names

Well-chosen names are one of the most important ways of making programs readable. The name chosen should describe the purpose of the identifier. It should be neither too short nor too long. As a guide, you should rarely need to concatenate more than 3 words.

Letter case

Constants use ALL_CAPITALS separated by underscore if needed
e.g. `PI, MAX_STARS`

Variables use firstWordLowerCaseWithInternalWordsCapitalised
e.g. `cost, numStars`

It is a good idea to use the so-called ***Hungarian notation*** where the first letter (or two) indicates what type the variable has:

- i for int, e.g. `iNum`
- f for float, e.g. `fFloatingPointNum`
- s for String, e.g. `sName`
- bo for Boolean, e.g. `boCar`

Functions use firstWordLowerCaseWithInternalWordsCapitalised
e.g. `cost(), numStars()`

Types of names

1. Names of *GUI widgets* should have a prefix indicating the type of control (Python TK Interface). The prefix should be lower case with the remainder of the name starting with upper case.

Control	prefix	example
Entry Widget	`ent`	`entQuantity`
Label Widget	`lbl`	`lblResult`
List box Widget	`lst`	`lstStudents`
Scrollbar Widget	`scroll`	`yscroll for vertical scroll bar`
Button Widget	`btn`	`btnQuit`
Radiobutton Widget	`rb`	`rbColour`
Checkbutton Widget	`chkb`	`chkbSugar`

2. Names of *functions that does not return a value* should be **nouns** or **noun-phrases**

   Example:

   ```
   newSum = sum(alpha, beta, gamma)
   ```

   or

   ```
   newSum = sumOfStudents(alpha, beta, gamma)
   ```

   rather than

   ```
   newSum = computeSum(alpha, beta, gamma)
   ```

3. Names of *functions that returns a value* should be **imperative verbs**

   Example:

   ```
   calculateCost(quantity, price)
   ```

   rather than

   ~~`results(newSum, alpha, beta, gamma)`~~

4. Names of *Boolean functions* should be **adjectives** or **adjectival phrases**

   Example:

   ```
   if (empty(list))
   ```

   or

   ```
   if (isEmpty(list))
   ```

   rather than

   ~~`if (checkIfEmpty(list))`~~

Comments

Comments should explain as clearly as possible what is happening. They should summarise what the code does rather than translate line by line. Do not over-comment.

In Python, use `##Comment` for a one-line comment.

Comments are used in three different ways in a program:

- Heading comments
- Block comments
- End-of-line comments.

Heading comments: These are generally used as a prologue at the beginning of each program and function. They act as an introduction, also describe any assumptions and limitations. They should show the names of any files, give a short description of the purpose of the program, and must include information about the author and dates written/modified.

Block comments: In general, you should not need to comment blocks of code. When necessary, these are used to describe a small section of following code. They should be indented with the program structure to which they refer. In this way the program structure will not be lost.

End-of-line comments: These need to be quite short. They are generally used with parameters to functions (to explain the purpose and mode of the parameter), and/or with variable declarations (to explain the purpose of the variable), and are not meant to be used for code.

Comment data, not code: Comments about the data – what it is and how it is structured - are much more valuable than comments about the code.

Prologue

Each of your programs should have a prologue. This is a set of "header comments" at the top of editor window. It should include

who wrote it	`## Author: (Your Name & ID)`
when it was written	`## Date created: 21 Feb 2021`
when it was last changed	`## Date last changed: 21 Mar 2021`
what it does	`## This program does...`
what files it reads / writes	`## Input: sample.txt, Output: none`

Named Constants

Program sometimes employs a special constant used several times in program. By convention Python programmers create a global variable and name is written in uppercase letters with words separated by underscore. For instance, SPEEDING_FINE, INTEREST_RATE

The special naming convention reminds the programmer that no reassignments to the variable should be made during the execution of the program. Since Python allows reassignments to any variable, hence the programmer is responsible for not changing the value of the variable.

This makes your program easier to maintain. If the speeding fine or interest rate changes, the change only has to be made in the one place.

Functions

Using functions is one of the simplest and most effective ways of dealing with complexity in a program.

> When you write your own function to perform a calculation, it should not refer to any GUI controls.
> Do not mix IO and calculations in a function.

Function that does not return a value

- encapsulates a task
- name should be a npun. Example**: sum**()
- should only do 1 task.

Function that returns a value

- encapsulates a query
- return type is **int, boolean, double**…;
- if return type is **boolean**, name should be an adjective, otherwise name should be a verb.
  - Examples: **isEmpty(list), SquareRoot(number)**
- should answer only 1 query.

Comments for each function should include (to be placed on the line before the function starts)

name	above rules apply
purpose	what it evaluates or what it does
assumptions	any assumptions made, particularly on the arguments

Duplicated Code

View any duplicated code with suspicion. Look for a way of factoring the duplicated code into a function.

Built-in functions

Unless you have a good reason, use built-in functions rather than writing (or not writing) your own. They will be correct, flexible and familiar to a reader.

Types

Using types is an important way of making your intentions clear. Python does not allow you to be sloppy with your types. In IIT, you are expected to choose appropriate types. You are also expected to use type conversions such as `int (23.26)`

Simplicity

Simplicity has long been recognised as one of the most important characteristics of programming. There are many ways of simplifying programs. These include avoiding nested IF statements, nested loops and complex conditions.

Submission

You need to submit one compressed file that contains **four files**:

1- **Three Python source code files** (ProgAsgStage1.py, ProgAsgStage2.py, ProgAsgStage3.py).

2- **One word document report** that **MUST** be very well organized and **MUST** contain the following parts:

2.1 The student information at the top, here is an example:

```
Student name: John Smith
Student ID: u210236
Assessment: Python Programming Assignment
Date: 22/05/2021
```

2.2 **Screenshots** as a Software testing evidence according to the following scenarios:

- **Stage 1 and Stage 2**:
  - a- Display all chook food names and prices.
  - b- Shopping Mash by bags using (3*50kg and 4*10kg)
  - c- Shopping Pellets by bags using (2*50kg and 4*10kg)
  - d- Shopping Enhanced Food by bags using (3*50kg and 7*10kg)
  - e- Shopping Mash by Kilograms using (80kg, 110kg)
  - f- Shopping Pellets by Kilograms using (30kg, 60kg)
  - g- Shopping Enhanced Food by Kilograms using (10kg, 80kg)
- **Stage 3**:
  - a- Read the food data from the file into the list.
  - b- Click all food types in the list and show their prices for 10kg and 50kg.
  - c- A graph to compare prices of all chook food type by different weights (10kg or 50kg).