programming language (version 3.x) and development must be on IDLE

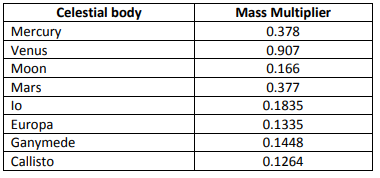
Planetary Exploration App Our astronauts are about to leave for a trip to outer space. They may be visiting the Moon, Mercury, Venus, Mars or one of the larger moons of Jupiter (IO, Europa, Ganymede or Callisto). Each astronaut has a mass allowance in kg for items that they can take along on the trip, including tools and personal items.

The amount they’re allowed depends on their job description.

Flight crew are allowed a total of 100kg; Mission specialists are allowed a total of 150kg.

Each of these solar system bodies (celestial bodies) has less gravity than the earth. For example, this means that a mass of 100kg on earth will weigh 100kg, but on the Moon, a mass of 100kg will weigh the equivalent of 16.6kg. You will be calculating how much mass each astronaut has left over for personal items, the total available mass and the average available personal mass allowance across all six astronauts. You will also calculate the weight of the average available personal mass allowance on the celestial body the astronauts are travelling to.

Stage 1: needs a list that contains the names of the celestial bodies that our astronauts may travel to and their Mass multipliers:



These details can be hardcoded in your program.

In Stage 1,

develop 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:

• Display list of celestial bodies that are possible destinations along with their mass multipliers

• Display the weight allowances for the two different types of astronaut

• Calculate personal mass allowances along with the total mass available

• Calculate the average available mass available for personal items

A nested list to store the name of the celestial body and its mass multiplier should be used. You can

assume there are three crew astronauts and three mission specialists. You may wish to store the mass of their tools in a list as well.

Notes:

• You should use functions for each task in the program, for example a printMenu function to print

a menu and calculcateAverageMass to calculate the average available mass etc.

• Your program should have a main() function to appropriately direct the program.

Necessary points within the code

• Constants vs literals. Using constants is important for the ease of maintenance. For example, consider constants for the mass multipliers for each celestial body and the upper limit for each type of astronaut.

• 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.

• 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?

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 available personal item mass for each astronaut and how to calculate the average

available mass and average available weight. Think about writing simple functions for the

repetitive calculations.

2. Create a new Python file, you may call it ProgAsgStage1

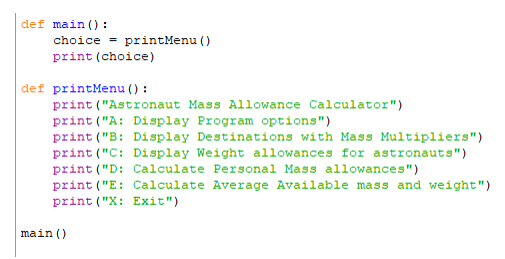
3. In the Stage1 class (file ProgAsgStage1.py), you may put all code for the user interaction and

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 name and mass multiplier of the celestial bodies.

4. You will need to write a function to print a menu to the user. One possible example is:



5. Now add the code that implements your strategy to display the different destinations with their

mass multipliers TEST

6. Add the code to display the weight allowances for the two different types of astronaut TEST

7. Add the code to calculate the personal mass allowances of each of the six astronauts along with

the total available mass TEST

8. Add the code to calculate the average available mass allowance and weight on destination TEST

9. Finally, add print statements to print the output to the user.

10. Test your implementation with the test cases mentioned below

# Stage 2:

In stage1, the user input and output is 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 Astronaut Allowance Calculator that provides an easy to use interface.

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 available personal weight allowances,

total and average values and

• an Exit or Quit button to properly close the program.

Use the same code for the calculations from Stage 1.

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 should be includeded:

• Constants vs literals. Using constants is important for the ease of maintenance.

. 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.

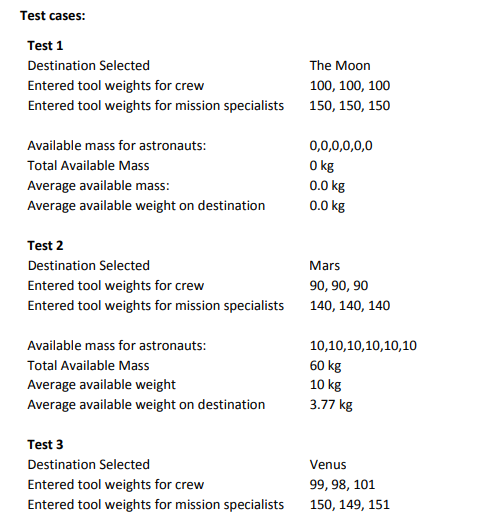
• Separate GUI functionality from calculations. Students should use a separate method for the

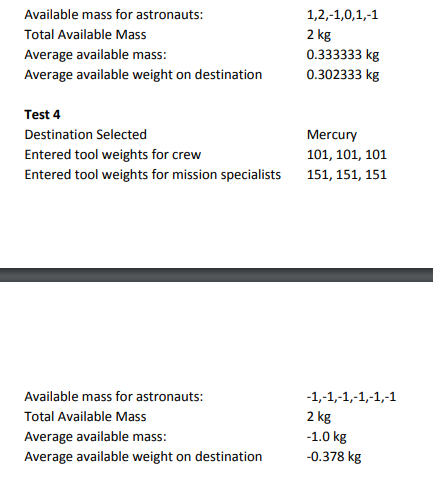
calculations. It is good practice and follows good programming style to separate GUI code from

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.

• Good names for your variables and constants.





Stage 2:

In stage1, the user input and output is 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 Astronaut Allowance Calculator that provides an easy to use interface.

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 available personal weight allowances,

total and average values and

• an Exit or Quit button to properly close the program.

Use the same code for the calculations from Stage 1

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.