Assessment (non-exam) Brief

UCL
SCHOOL OF MANAGEMENT

Module code/name	INST0002 Programming 1/INST0091 Introduction to Programming
Module leader name	Dr Daniel Onah
Academic year	2024/25
Term	2
Assessment title	Tutorial Sheets 1, 2,3,4,5,6
Individual/group assessment	Individual

Submission deadlines: Students should submit all work by the published deadline date and time. Students experiencing sudden or unexpected events beyond your control which impact your ability to complete assessed work by the set deadlines may request mitigation via the extenuating circumstances procedure. Students with disabilities or ongoing, long-term conditions should explore a Summary of Reasonable Adjustments.

Return and status of marked assessments: Students should expect to receive feedback within one calendar month of the submission deadline, as per UCL guidelines. The module team will update you if there are delays through unforeseen circumstances (e.g. ill health). All results when first published are provisional until confirmed by the Examination Board.

Academic Misconduct: Academic Misconduct is defined as any action or attempted action that may result in a student obtaining an unfair academic advantage. **Academic misconduct includes plagiarism, obtaining help from/sharing work with others be they individuals and/or organisations or any other form of cheating.** Refer to Academic Manual Chapter 6, Section 9: Student Academic Misconduct Procedure - 9.2 Definitions.

Referencing: You must reference and provide full citation for ALL sources used, including articles, textbooks, lecture slides and module materials. This includes any direct quotes and paraphrased text. If in doubt, reference it. If you need further guidance on referencing please see UCL's referencing tutorial for students. Failure to cite references correctly may result in your work being referred to the Academic Misconduct Panel.

Use of Artificial Intelligence (AI) Tools in your Assessment: Your module leader will explain to you if and how AI tools can be used to support your assessment. In some assessments, the use of generative AI is **not permitted** at all. In others, AI may be used in an **assistive** role which means students are permitted to use AI tools to support the development of specific skills required for the assessment as specified by the module leader. In others, the use of AI tools may be an **integral** component of the assessment; in these cases the assessment will provide an opportunity to demonstrate effective and responsible use of AI. See page 3 of this brief to check which category use of AI falls into for this assessment. Students should refer to the UCL guidance on acknowledging use of AI and referencing AI. Failure to correctly reference use of AI in assessments may result in students being reported via the Academic Misconduct procedure. Refer to the section of the UCL Assessment success guide on Engaging with AI in your education and assessment.

:

Content of this assessment brief

Section	Content
A	Core information
B	Coursework brief and requirements
C	Module learning outcomes covered in this assessment
D	Groupwork instructions (if applicable)
E	How your work is assessed
F	Additional information

Section A Core information

Submission date	On the module page
Submission time	On the module page
Assessment is marked out of:	100
% weighting of this assessment within total module mark	100%
Maximum word count/page length/duration	NA
Footnotes, appendices, tables, figures, diagrams, charts included in/excluded from word count/page length?	NA
Bibliographies, reference lists included in/excluded from word count/page length?	NA
Penalty for exceeding word count/page length	No specified maximum (and thus no penalty for exceeding)
Penalty for late submission	Standard UCL penalties apply. Students should refer to Refer to https://www.ucl.ac.uk/academic-manual/chapters/chapter-4assessment-framework-taught-programmes/section-3-moduleassessment#3.12
Submitting your assessment	Weekly at the scheduled deadline/time

:
:

Anonymity of identity. Normally, <u>all</u> submissions are anonymous unless the nature of the submission is such that anonymity is not appropriate, illustratively as in presentations or where minutes of group meetings are required as part of a group work submission	The nature of this assessment is such that anonymity is not required.

Section B Assessment Brief and Requirements

On the Tutorial sheets.

Section C: Module Learning Outcomes covered in this Assessment

This assessment contributes towards the achievement of the following stated module Learning Outcomes as highlighted below:

Weekly lectures for INST0002 Programming 1

Section D : Groupwork Instructions (where relevant/appropriate)

NA

Section E

: How your work is assessed

Within each section of this assessment you may be assessed on the following aspects, as applicable and appropriate to this assessment, and should thus consider these aspects when fulfilling the requirements of each section:

- The accuracy of any calculations required.
- The strengths and quality of your overall analysis and evaluation;
- Appropriate use of relevant theoretical models, concepts and frameworks;
- The rationale and evidence that you provide in support of your arguments;
- The credibility and viability of the evidenced conclusions/recommendations/plans of action you put forward;
- Structure and coherence of your considerations and reports;
- Appropriate and relevant use of, as and where relevant and appropriate, real world examples, academic materials and referenced sources. Any references should use either the Harvard OR Vancouver referencing system (see References, Citations and Avoiding Plagiarism)
- Academic judgement regarding the blend of scope, thrust and communication of ideas, contentions, evidence, knowledge, arguments, conclusions.
- Each assessment requirement(s) has allocated marks/weightings.

Student submissions are reviewed/scrutinised by an internal assessor and are available to an External Examiner for further review/scrutiny before consideration by the relevant Examination Board.
It is not uncommon for some students to feel that their submissions deserve higher marks (irrespective of whether they actually deserve higher marks). To help you assess the relative strengths and weaknesses of your submission please refer to UCL Assessment Criteria Guidelines, located at https://www.ucl.ac.uk/teaching-learning/sites/teaching-learning/files/migratedfiles/UCL_Assessment_Criteria_Guide.pdf

The above is an important link as it specifies the criteria for attaining 85% +, 70% to 84%, 60% to 69%, 50% to 59%, 40% to 49%, below 40%.

You are strongly advised to **not** compare your mark with marks of other submissions from your student colleagues. Each submission has its own range of characteristics which differ from others in terms of breadth, scope, depth, insights, and subtleties and nuances. On the surface one submission may appear to be similar to another but invariably, digging beneath the surface reveals a range of differing characteristics.

Students who wish to request a review of a decision made by the Board of Examiners should refer to the UCL Academic Appeals Procedure, taking note of the acceptable grounds for such appeals.

Section F

Note that the purpose of this procedure is not to dispute academic judgement – it is to ensure correct application of UCL's regulations and procedures. The appeals process is evidence-based and circumstances must be supported by independent evidence.

Section G

: Additional information from module leader (as appropriate)

Work in accordance with the weekly assessment brief and the tutorial sheet questions herein.

Section H

INST0002: Programming 1
Tutorial Sheet 03
Week 03

Academic Year: 2024/25

Set by: Daniel Onah

The aim of the practical sheet is to provide you with a set of exercises to practice developing your programming skills. Writing programmes will help you develop programming skills and prepare for the Programming Test and the Moodle Quiz.

Assessed Tutorial Questions:

Some questions in the labs are marked with a [*] symbol. These questions are compulsory and you will be assessed on one or more of these in the following week's lab. For most of you, this means you will be assessed on:

- Tutorial sheet 1 in Lab 2
- Tutorial sheet 2 in Lab 3
- Tutorial sheet 3 in Lab 4
- Tutorial sheet 4 in Lab 5
- Tutorial sheet 5 in Lab 7, and
- Tutorial sheet 6 in Lab 8

Please look at all the questions spend some time thinking carefully about them, before asking for help. If you are still stuck:

- attend the live practical & booster sessions and ask me, or the TAs, for help
- or post a question on the Moodle discussion board.

Once you have completed the assessed questions you should upload the specified files to the Assessment tab on Moodle under the appropriate submission link. You can submit at the specified submission time on Moodle. In preparation for your 1-to-1 viva, you should be able to clearly indicate which constructs such as classes, functions, variables used in your program is of your own work and which classes or features where from other sources (e.g., from textbooks, online sources etc.).

Please refer to Read about scheduling and live sessions and Read about tutorial based assessment, submissions and vivas for additional information on the procedures of being assessed.

A reminder on how to compile a programme

This is just a quick reminder of how to compile a simple Python program.

1) Make sure you organise your files into folders for each of the weeks:

Published September 3rd 2022

Section I

a. You should keep your files on the `N:` drive. I suggest creating a folder called `INST0002` with subfolders `week01, week02` etc. for each of the tutorial files.

2) You should open the Command Prompt Shell and navigate to the current directory where your Python files are located, for example, `N:/INST0002/week01`. To do this, use the cd command. For example:

```
cd  N:/INST0002/week01
```

To change the drive from C: to N: simply type in `N:` in the command prompt shell. Speak to a TA or a peer student if you need help with using the N drive or the Command Prompt shell.

3) Once in the working directory, you can execute/run the files without providing the path to the file:

```
python hello_world.py
```

(where python is the name of the interpreter that will trigger the Runtime Environment and execute specific file)

Section J

Exercise 1 [*]
Learning Objectives: [A] conditional statement; [B] input() & print() function.

You are applying for a mortgage from UCL bank. Before your mortgage is approved you will need to meet certain income requirements. In order for you to borrow a loan from the bank, your income need to be calculated and the threshold amount allocated to your income range will be printed out. The program should ask the user to enter how much they earn in a year in pounds.
Using the following print statement:

- If the income is greater than or equal to £100,000. The following print statement should be displayed " You can borrow any amount."
- If the income is greater than or equal to £50,000. The following print statement should be displayed " You can borrow up to 500,000 pounds."
- If the income is greater than or equal to £30,000. The following print statement should be displayed " You can borrow up to 400,000 pounds."
- If the income is greater than or equal to £20,000. The following print statement should be displayed " You can borrow up to 200,000 pounds."
- If the income is greater than or equal to £10,000. The following print statement should be displayed " You can borrow up to 100,000 pounds."
- Otherwise, your program should print "You do not qualify for a mortgage" for any other input lower than £10, 000.

Exercise 2 [*]
Learning Objectives: [A] use a for loop; [B] use of break statement; [C] relational operators [D] conditional statements – if, elif, else.

In this task you are required to write a program to calculate the grades of a student and compute the cumulative grade point average (GPA). The program should prompt the student to enter how many modules they have. The input should be captured or stored as integer values. You should use for-loop construct for the task. Create a for-loop with the range between 1 and the number of modules inclusive (study the Lecture 3 resources for more about how to do this). Use a try and except error handler to check whether the grade entered by the student is less than zero (i.e. the program should check for invalid entries including negative grades). This means the program should check for negative values using appropriate comparison/relational operator. The program should be able to calculate the average grade for the student and the degree classification (e.g. if a student score 90 , the program should display the ***grade*** and *"First class"* classification).

Grade	**Classification**
80 - 100	First class
70 – 79	Second Class Upper
60 – 69	Second Class Lower
50 – 59	Third Class

Section K

40 – 49	Pass
0 – 39	Fail

Example:

Let's say the pass mark for the programme is between 40% to 49%. If the student's computed average was 44 as their GPA. The program condition should check whether the average grade (GPA) is *greater than or equal to 40 and less than or equal to 49*. If the condition is true, this should then display the average of 44 and "Pass" classification.

Hint: Use the conditional statements of *if, elif* and *else* to check these various conditions. You should use for-loop for continuous iteration.

```
danny$ python3 exam_grade.py
How many modules do you have: 3
Please enter the grade for module number 1: 67
Please enter the grade for module number 2: 78
Please enter the grade for module number 3: 98
Your cumulative GPA is:  81.0
First Class

danny$ python3 exam_grade.py
How many modules do you have: 4
Please enter the module number 1: -10
You have entered a negative number -10
Please enter only positive integer grades. Thank you!
Your average grade is:  0
Fail

danny$ python3 exam_grade.py
How many modules do you have: 5
Please enter the module number 1: k
Please enter only positive integer grades. Thank you!
Your average grade is:  0
Fail
```

Section L

Exercise 3 [*]
Learning Objectives: [A] use a for & while loop; [B] use of Python random; [C] Primitive types and relational operators [D] conditional statements – if, elif, else.

Write a program for a Roller coaster company that wishes to provide authorisation or access to only eligible adults. Let's say the company can only accommodate only about 100 customers at a business day. Design the program to generate random numbers between 1 and 100 for these customers.

Description:
The main aim of this program is to provide accessibility to only adult customers and restrict others. The program should consider a certain age limit and check the condition against this age. The age limit required for this program is **18** years. So for a customer less than 18, they should not be allowed to join in the roller coaster ride.

Assuming we declared a constant **SENTINEL** value of **-999**. Write an if statement to check or determine whether the input entered by the user is this sentinel value. If the input is the value, then the program should print a statement *" Thank you for coming. See you again!"* and terminate. The program should take the username together with the age. Also, we need to provide the user with the opportunity to terminate the program at the first prompt i.e. "Please enter your name". If the user enters nothing (blank entry) or enters the character **'n'** (which stands for no), then the program should display *"Thank you for coming. See you again!"* and terminate. Since the main aim of the program is to use the **age** to authorise eligibility to take the ride, we have to define and write the code inside a try and except block that would handle any mismatch entry and allows only integers or whole numbers to be entered as the **age** value.

Note: Try to run your program using *python* or *python3* (see the expected output).

Hint: The program should request for continuous input and only when the sentinel value is entered for the ***age*** and a blank for the ***name*** (or 'n') before the program can end. Use the try and except construct from the lecture 3 to check whether the input entered is appropriate as required by the task.

```
"""
A Roller coaster program.
"""
# Author: Danny Onah
# Date: 20 Jan.2023
# Time: 15:12PM GMT
```

Section M

```
import random
# define a sentinel value to use
SENTINEL = -999
# this will produce random numbers from 1 - 100 limit
secretNumber = random.randint(1, 100)

# use a while loop


# define the try and except block here

    try:
    # declare the conditional statement to check for the first condition

       # declare the conditional statement to check whether the customer is under-age

    except ValueError:
     # display the results of the exception
```

Program expected outcome:

```
danny$ python3 roller_coaster_ride.py
Please enter your name: Danny
Pease enter your age: 20
Welcome  Danny , Ride number is:  87

Please enter your name: Onah
Pease enter your age: 16
Welcome  Onah , your age is:  16
You are a junior, therefore you are not authorised to book the roller coaster ride!

Please enter your name: n
Thank you for coming. See you again!

danny$ python3 roller_coaster_ride.py
Please enter your name: Dan
Pease enter your age: -999
Thank you for coming. See you again!

danny$ python3 roller_coaster_ride.py
Please enter your name: DanOnah
```

Section N

```
Pease enter your age: hello
Please enter age as integers-whole numbers only...

Please enter your name: (this name here is blank or empty)
Thank you for coming. See you again!
```

Exercise 4[*]

Learning Objectives: [A] double-selection using `if…else`*; [B] use of input function; [C] Primitive types and simple arithmetic operations.*

Case study: A market stall owner is planning to resell 55kg of apples. She needs to box the apples before taking these to the market, but does not know how many boxes to buy. She knows that each of the boxes takes 10kg of apples. How many boxes will the stall owner need?

Write a programme that will help the stall owner in the above case study to calculate how many boxes they need. The programme should ask the user for the weight of apples they have (this must be a ***floating-point*** number of kilos) and the program should be able to calculate the number of boxes they (stall owner) need for the given weight as shown below.

The programme should prompt the user to specify the weight the box can carry, followed by the weight of produce, before calculating the number of boxes required for the given weight of produce. Make sure that the user is able to use floating-point numbers (e.g. `decimal numbers`) for both weight of produce as well as the weight that the boxes can carry.

Program expected outcome:

```
danny$ python apples_boxes_calculator.py
Enter the maximum weight the box could take: 72.67
Enter the weight of apples to be boxed: 189.98
The number of boxes you need is: 3
```

Exercise 5[*]

Learning Objectives: [A] double-selection using `if…else`*; while loop [B] use of input function; [C] Strings, break statement.*

Case study: A real world program to present meal specials to customers from a restaurant and with a note describing each meal. If the meal entered by the customer is not on the list, then the program should print an error message. The program should continuously prompt the customer for input. The restaurant decides to write the program to ask the customer to enter the meal-time that they are interested in.

Section O

Your task is to write the program using conditional statements ***if***, ***elif*** and ***else*** to display only the details of the meal-time selected by the customer.

Note: The meal-time special for this program specification are: **breakfast**, **lunch** and **dinner**. Remember, we are not calculating the time within this program using any primitive datatype (int, float etc). We are only required to use strings throughout.

Hint: Getting input and formatting output is often very important to use Python scrips to write such a program. Use a while-loop for continuous entry and control of the meal-time special input (ensure you handle any infinite loop). Define a condition that would stop the infinite loop if the customer finishes or end their order. You should use any ***print()*** function that you are familiar with.

You are advised to study the given code and understand it fully before attempting the exercise. The sample output provided immediately after the listing of the given ***meal_time.py*** *file may help you complete the exercise.*

```
"""
A real world program to present meal specials to customers from a restaurant and with a note for
each meal. If the meal enter is not on the list then the program will print an error message.
The program should continuously request the customers for input.

Getting input and formatting output is often very important to use scripts to write such a
program. The restaurant decides to write the program to ask the customer to
enter the meal-time that they are interested in. Write your if-statement to print out only the
meal-time selected by the customer.
"""

# Author: Danny Onah
# Date; 7 January 2023
# Time: 23:02PM GMT


# Declare the meal special (You don't need to change the variables in this script)
# Use these variables to write the program

breakfast = "Texas Omelet"
breakfast_description = " Contains brisket, horseradish cheddar"
lunch = "Donburi"
lunch_description = " A soul warming bowl of steaming rice, packed with protein + crunchy
vegetables"
```

Section P

```
dinner = "Teppanyaki"
dinner_description = " Noodles sizzling from the grill, turned quickly so the noodles are soft
and the vegetables stay crunchy"



# declare your while loop here


# Request the user to enter their meal-time (using input() function)

meal_time = # complete this construct

# write the condition for ending the loop


# using if, elif and else conditional statement to check and print the meal and description
```

Program expected outcome:

```
danny$ python meal_time.py

Which meal-time do you want? [breakfast, lunch, dinner](or enter “n” to end-loop): breakfast
 *** Specials for breakfast ***
Texas Omelet:
 Contains brisket, horseradish cheddar

Which meal-time do you want? [breakfast, lunch, dinner] (or enter “n” to end-loop): dinner
 *** Specials for dinner ***
Teppanyaki:
 Noodles sizzling from the grill, turned quickly so the noodles are soft and the vegetables stay
crunchy

Which meal-time do you want? [breakfast, lunch, dinner] (or enter “n” to end-loop): Rice pudding
 *** Specials for Rice pudding ***
Sorry, " Rice pudding " isn't on the list of the meal-special menu in the restaurant.

Which meal-time do you want? [breakfast, lunch, dinner] (or enter “n” to end-loop): n
Thank you for your patronage. Hope to see you next time!!!
```

Section Q

Exercise 6 [*]
Learning Objectives: [A] use a for & while loop; [B] use of Python random; [C] Primitive types and relational operators [D] conditional statements – if, elif, else, break & continue.

Case Study:
A common game for kids on a boring bus journeys, is for one person to randomly pick a number between an upper and lower limit and for the other person to try to guess that number in as few guesses as possible.

Write a program which randomly picks a number between 1 and 20 (both inclusive) and repeatedly asks the user to enter a guess ("Guess:"). Depending on the number entered, do one of the following things as described in this task.

The completed programme should represent a simple game where users are given 20 points to start with and asked to guess a number that was randomly generated using a `random library (import random)`. You do not have to implement this functionality as it is already added and explained in the code below.

Game Description
Write a program to use a loop that would allow a player only **6 guesses** of a randomly generated machine number. Note that you would need to create a loop that should allow the player only 6 guesses. Begin the program to start the game by offering the user an initial *20 points*. The **points** should be reduced by 1 after each unsuccessful attempt (or guess). This means each wrong guess will reduce the score by 1 point. If the player correctly guesses the number on or before the 6 attempts, the program should display *'Good Job! You are a Winner. You have successfully guessed the number in _ guesses* (replace the **'_'** with the number of guesses entered in order to arrive at the correct guess). Otherwise, if the player's allowed attempt reaches 6, the program should terminate the loop and display *'You have loss the Game. The number was _'* (replace the '_' with the random number or correct number).

Program interactive display:

- The program should be able to capture if a number greater than **20** is entered, this should display *"Invalid entry! The allowed number should fall between 1 and 20"*. The program should not terminate but proceed to request for the next entry. Remember, this is also considered as a guess, therefore points should be reduced or deducted here as well by 1 and the program should display the remaining points.
- If the player's guess is higher than the random number generated, the program should display '*Your guess is too high.'* and reduce the points by 1. This should display the current points remaining.

Section R

- If the player's guess is lower than the random number generated, the program should display *'Your guess is too low.'* and reduce the points by 1. This should display the current points remaining.

You are advised to study the given code and understand it fully before attempting the exercise. The sample output provided immediately after the listing of the given ***number_guesser.py*** *file may help you complete the exercise.*

```
# Author: Danny Onah
# Date: 20 January 2023
# Time: 02:20AM GMT
"""
This is a guess the number game.
"""
import random # import the random package or library to help with the random number generation
currentPoints = 20 # Initial score or points allocated to the user
guess = 0 # initialise the guess to 0
secretNumber = random.randint(1, 20) # generate random numbers between 1 and 20
print('You are given 20 points for this game. \nYou are allowed 6 guesses between 1 and 20.')

# Ask the player to guess 6 times (using a for-loop here)
# See the lecture 3 resources to understand how to do this.


# Request for player guess

# Additional or Bonus Task (try and except handler [OPTIONAL])

# define your conditional statements here.
# Using only if, elif and else


# Check whether the input entered is valid, between 1 and 20


# Define a condition to check whether the guess is low

# Define a condition to check whether the guess is high


# Check using an if conditional statement whether the guess is the correct guess and the number
of guess is 1!
```

Section S

```
# Check if the condition is met and the player guess is correct
# Display the WINNER -- game statement


# Otherwise, display the LOST -- game statement
```

Additional or Bonus Mark [optional]:
You can assume the user will only type in whole numbers (i.e., conversion from string to integer won't fail or terminate the program). Write the program using the try and except block to check whether the input entered is an integer or whole number. For example, if a character or string is entered as input, the program should be able to handle this and display *'Please enter a valid number between 1 and 20'* and proceed to request for the next entry.

Adjust the program to allow negative scores, and this can be reached when the number of failed attempts at guessing the number go beyond 20 (This means if the failed guesses decrease from 20 to 19 to 18 …0 to -1 to -2 etc). Use a while loop for this, so the program can run infinitely and request for input from the player.

Note: You can save this bonus task as ***guess_number_bonus.py***.

The output of the program may look as follows:

```
danny$ python guess_number.py
You are given 20 points for this game.
You are allowed 6 guesses between 1 and 20.
Guess the number: 1
Your guess is too low.
Current score: 19

Guess the number: 0
Invalid entry! The number should be between 1 and 20.
Current score: 18

Guess the number: -1
Invalid entry! The number should be between 1 and 20.
Current score: 17

Guess the number: 3
Good job! You are a Winner. You have successfully guessed the number in 4 valid guesses!
Congrats! Your final score is: 17
```

Section T

```
danny$ python guess_number.py
You are allowed 6 guesses between 1 and 20.
Guess the number: 3
Good job! You are a Winner. You have successfully guessed the number in 1 valid guess!
Congrats! Your final score is: 20


danny$ python guess_number.py

You are given 20 points for this game.
You are allowed 6 guesses between 1 and 20.
Guess the number: 6
Your guess is too low.
Current point: 19

Guess the number: 10
Your guess is too low.
Current point: 18

Guess the number: 16
Your guess is too low.
Current point: 17

Guess the number: 20
Your guess is too high.
Current point: 16

Guess the number: 19
Good job! You are a Winner. You have successfully guessed the number in 5 valid guesses!
Congrats! Your final score is: 16


danny$ python guess_number.py
You are given 20 points for this game.
You are allowed 6 guesses between 1 and 20.
Guess the number: 2
Your guess is too low.
Current point: 19

Guess the number: 4
Your guess is too low.
Current point: 18
```

Section U

```
Guess the number: 7
Your guess is too low.
Current point: 17

Guess the number: 9
Your guess is too low.
Current point: 16

Guess the number: 18
Your guess is too high.
Current point: 15

Guess the number: 16
Your guess is too low.
Current point: 14
You have lost the Game. The number was:  17
```