Assessment (non-exam) Brief

UCL
SCHOOL OF MANAGEMENT

Module code/name	INST0004 Programming 2
Module leader name	Dr Daniel Onah
Academic year	2023/24
Term	1
Assessment title	Tutorial Sheets 1, 2,3,4,6,8
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.

C: Section 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 INST0004 Programming 2

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

INST0004: Programming 2
Tutorial Sheet 02
Week 02

Academic Year: 2023/24

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 6 in Lab 7, and
- Tutorial sheet 8 in Lab 9

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.

Published September 3rd 2022

Section I

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:
   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[*] Volume of Sphere

Learning Objectives: [A] input() function [B] data types and simple arithmetic operations [3] import library.

Write a program to compute the volume of a sphere given the formula below. If the radius of the sphere is entered by a user as 2.3 and this is stored in a variable called radius (**'r'**) (see output). Using the formula to construct a Python arithmetic expression to calculate the volume of the sphere. You are required to compute the volume of the sphere to two decimal places.

$$V = \frac{4}{3}\pi r^3$$

Hint: Remember to import a math library to use the ***pi*** value. In mathematics the value of ***pi*** is **3.14159265359.**

The output of the programme should be:

```
danny$ python3 volume_sphere_ex01.py
Enter the radius of a sphere: 2.3
The volume of the sphere is:  50.97
```

Exercise 2 [*] Volume of a Bottle/Cylinder

Learning Objectives: [A] print() function [B] data types and simple arithmetic operations, [C] primitive types, [D] import statement

The shape of a bottle is approximated by two cylinders of radius r_1 and r_2 and heights h_1 and h_2, joined by a cone section of height h_3. Using the formula for the volume of a cylinder, $V = \pi r^2 h$, and a cone section, develop a program to compute the volume of the bottle. The program should request for user input for the two radiuses (radii) $\mathbf{r}_1$, $\mathbf{r}_2$ and **height**. Let's assume these are taking in meters (metres). Using an actual bottle with known volume as a sample, calculate the volume of the bottle to two decimal places using the formula below.

$$V = \pi \frac{(r_1^2 + r_1 r_2 + r_2^2)h}{3}$$

Section K

Hint: Try solving this equation by hand first. Then translate this to your Python program. Remember to print the volume measurement as **m^3** as shown in the output. One way is to use Unicode or character mapping code equivalent to the exponential construct in your Python version (this works for Python3 above). Use the concept of **BODMAS** in the lecture 2.

The output of the programme should be:

```
danny$ python3 volume_bottle_ex02.py
Enter the first radius (r1): 5
Enter the second radius (r2): 7
Enter the height of the bottle: 10
The volume of the bottle is: 1141.45 cubic metre (m³)
```

Exercise 3[*] A Tourist Direction Program

Learning Objectives: [A] input() function [B] conditional statements [C] loops [E] equality operator.

Write a program to provide direction for a tourist or visitor from a train station to a residential apartment. You are required to create a menu-options for which the tourist can select to get the appropriate direction to the location.

The menu:

- *0: Exit*
- *1: Harrow-on-The-Hill Station.*
- *2: St. Ann's Shopping Centre.*
- *3: St. George's Shopping Centre*
- *4: Bradstowe House, Headstone Road.*

The program should allow the user to enter a number to select one of the options for the direction. This will then provide the route direction of walk from the current location point that was selected or chosen by the tourist. You should use a while-loop to allow for continuous entering of the number input for each of the options. The program should terminate or stop once zero (0) is entered as input. In order to provide the correct direction, use **if – elif - else** conditional statements.

Section L

Program implementation: This program should be implemented using both **while-loop** and **if - elif & else** conditional statements. The program should display the current location and present the next route as the options are selected. Write the program within a ***main()*** function.

Python Pictography

You are advised to study the given Python Pictography and understand it fully before attempting the exercise. The sample output provided immediately after the listing of the given program and pictography may help you complete the exercise.

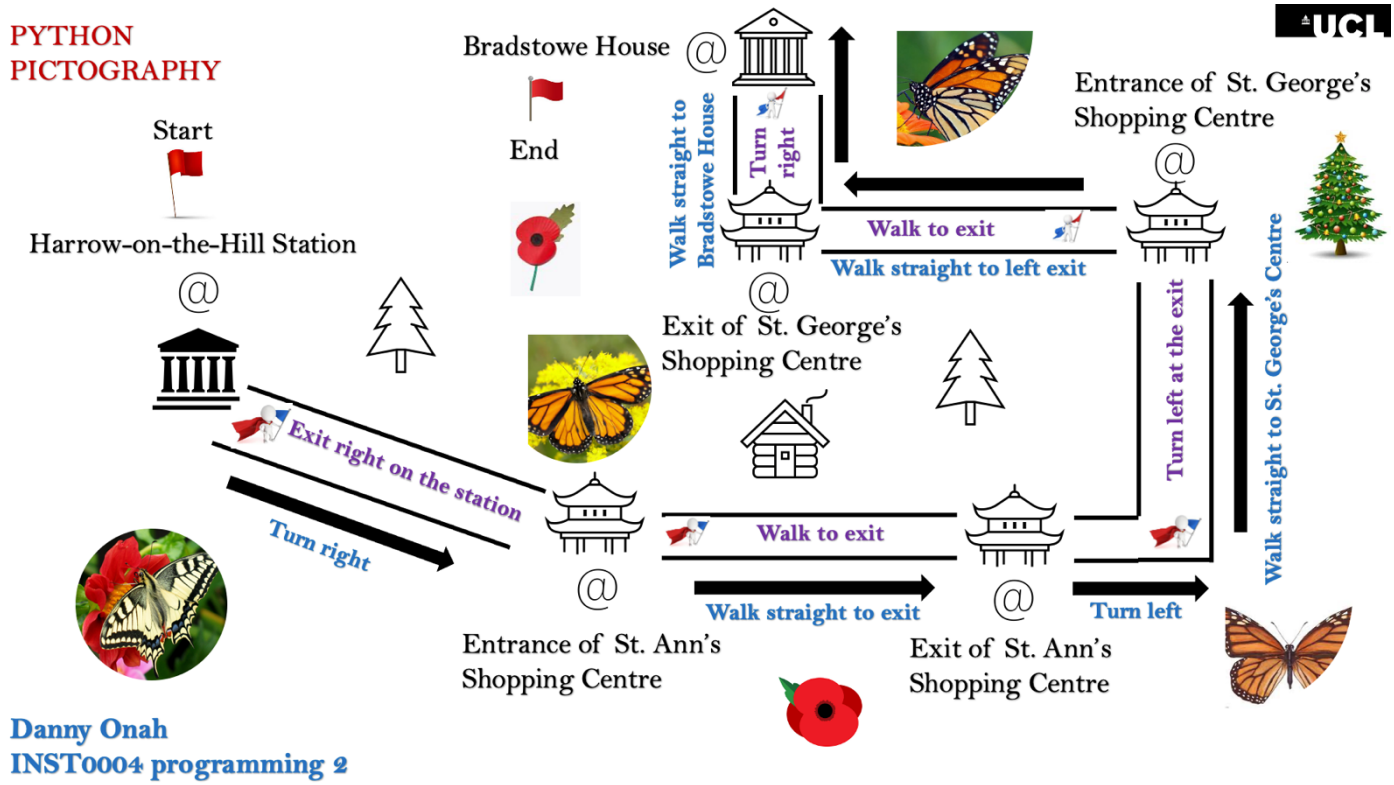

Image-designed: Python Pictography (by Danny)

Section M

Additional Task (Bonus) [Optional]:
Improve the program and rewrite this using match and cases construct in Python. Write this program without using ***if , elif*** and ***else***, but with only the **match** & **case** construct in Python. In other language such as *Java* this is known as switches and cases. If the user enters zero (0) the selection case number is triggered, and the program should end the execution. **Note:** Follow the same logic as the original description. You can save this new program as ***tourist_direction2_ex3.py***

The output of the programme should be:

```
danny$ python3 tourist_direction_ex3.py


*** Please select an option ***
0: Exit
1: Harrow-on-The-Hill Station.
2: St. Ann's Shopping Centre.
3: St. George's Shopping Centre
4: Bradstowe House, Headstone Road.
Enter the number (1 - 4):1
**************************************************
You are at the Harrow-on-The-Hill Station.
Turn right towards St. Ann's Shopping Centre.
**************************************************
*** Please select an option ***
0: Exit
1: Harrow-on-The-Hill Station.
2: St. Ann's Shopping Centre.
3: St. George's Shopping Centre
4: Bradstowe House, Headstone Road.
Enter the number (1 - 4):2
**************************************************
You are at St. Ann's shopping centre
```

Section N

```
Walk towards the exit at the end.
Turn left towards St. George's Shopping Centre.
**************************************************
*** Please select an option ***
0: Exit
1: Harrow-on-The-Hill Station.
2: St. Ann's Shopping Centre.
3: St. George's Shopping Centre
4: Bradstowe House, Headstone Road.
Enter the number (1 - 4):3
**************************************************
You are at St. George's shopping centre
Walk towards the exit on the left.
Turn right at the exit.
Walk straight to Bradstowe House, Headstone Road ahead.
**************************************************
*** Please select an option ***
0: Exit
1: Harrow-on-The-Hill Station.
2: St. Ann's Shopping Centre.
3: St. George's Shopping Centre
4: Bradstowe House, Headstone Road.
Enter the number (1 - 4):4
**************************************************
You are at Bradstowe House, Headstone Road.
You have reached your destination.
**************************************************
*** Please select an option ***
```

Section O

```
0: Exit
1: Harrow-on-The-Hill Station.
2: St. Ann's Shopping Centre.
3: St. George's Shopping Centre
4: Bradstowe House, Headstone Road.
Enter the number (1 - 4):0
**************************************************
You have exited the program.
End of direction search.
Goodbye!:)
**************************************************
```

Exercise 4 [*] Lifts Program – Apartments & Buildings

Learning Objectives: [A] create user-defined function [B] input and type conversion [C] conditional statement [D] list container and return statement [E] import module [F] logical, equality and relational operators [G] list construct.

Write a program that would mimic the working functionality of a residential or office building lifts. Suppose a user is at a particular floor and they wish to take the lift out of the building or apartment. The user should be able to press the button of the lift at their floor to call the service of the lift. You should write the program to request the user to enter the number of lifts in the building or apartment (as input). For example the user can enter at least **2** or more integer input to indicate the number of lifts in the building or apartment.

Note: *Remember the main aim of the program is to send the lift nearest to the user's floor once this is called (or as soon as the user presses the lift button at the current floor location for which they are)*.

Programming Description

In this program, you should apply your knowledge of list container. Create an empty list called ***liftsFloor*** to store the number of lifts from the user input (e.g. 2). Create an arbitrary list of numbers called ***liftsNumbers*** in ascending order from 0 – 9, representing the number of floors in the building or apartment. Your program should generate random numbers from the list with respect to the number of lifts in the building apartment. You should then iterate over the list and add (***append***) the randomly generated numbers into the empty list "*liftsFloor*". Create two functions: (1) the first called *minimumFloor()* , should compute the lift in the minimum floor location, (2) the second called

Section P

maximumFloor() should compute the lift in the maximum floor location. Note, you have to use **relational**, **equality** and **logical** operators to decide the lift to call (*be careful as this is the tricky part*).

Hint: Write the entire program embedded in a ***main()*** function. *Please revise and study more about **LIST** covered in Programming 1.*

Pseudocode:

- If the user floor number is the same as the lift number, display a message to call the lift at the floor of the user.
- If all lifts are at the same floor number, create a construct to call one of the lifts (one way you can do this is by using a timer input) or you can simply call both lifts (this is also acceptable).
- Both functions (***minimumFloor()*** and ***maximumFloor()***) should take as parameter list the list items stored in the empty list (i.e. ***liftsFloo**r*)
- Then call the functions within the ***main()*** function

Additional task [optional]: *Use while loop to continuously requesting the user for input (i.e. by entering the floor number).*

The output of the programme should be:

```
danny$ python3 lifts_program_ex04.py
Please enter the number of lifts in the apartment: 2
Enter the floor you are (0 - 9): 8
Lift_in_floor: 3
Lift_in_floor: 5
*** Lift closer to the user's floor is called! ***
Lift in higher floor 5 is called!

danny$ python3 lifts_program_ex04.py
Please enter the number of lifts in the apartment: 2
Enter the floor you are (0 - 9): 3
Lift_in_floor:  9
Lift_in_floor:  2
*** Lift closer to the user's floor is called ***
Lift in lower floor 2 is called!
```

Section Q

Exercise 5 [*] Arithmetic Computation

Learning Objectives: [A] create user-defined function [B] input and type conversion [C] conditional statement [D] list container[E] user defined function [F]relational operators [G] list construct.

In this task you are required to define your own user functions to compute the maximum, minimum and the sum of elements in a list.

Use the following to construct your program:

- Define a user function called *maximumNumber()* that takes as a parameter list a list of floating-point numbers.
  - ***This function should compute the maximum number in the list.***
- The second function should be called *minimumNumber()* that takes also as a parameter list a list of floating-point numbers.
  - ***The function should compute the minimum number in the list***
- The final function called *summation()* should also take as a parameter list the list of floating-point numbers.
  - ***This function should compute the total sum of the elements(numbers) in the list.***

Program construct:

- Create an empty list to store the list items.
- Request user to enter the number of items or the length of the elements to be stored in the list.
- User should enter the list items using floating-point numbers (e.g. 3.6, 12.348 etc).
- Display all entries in 2-decimal places.
- Use a for-loop to iterate and take the user input and store this in the empty list.
- Embed the whole program within a **main()** function. In the *main()* function do the following:
  - Call the maximum number function with appropriate arguments
  - Call the minimum number function with appropriate arguments
  - Call the summation function with appropriate arguments

Note: ensure you know how to add the items to the empty list (e.g. append()). Also, decide the argument to pass into the functions.

Hint: You can use the argument you pass as the parameter list also for the function call. *Please revise and study more about **LIST** covered in Programming 1.*

Section R

The output of the programme should be:

```
danny$ python3 arithmetic_computation_ex05.py
Enter the number of Items: 4
Enter number in index[0]: 23.45
Enter number in index[1]: 30.12
Enter number in index[2]: 2.90
Enter number in index[3]: 76.39
The total sum is:  132.86
The maximum number is: 76.39
The minimum number is: 2.90

danny$ python3 arithmetic_computation_ex05.py
Enter the number of Items: 4
Enter number in index[0]: 23.345
Enter number in index[1]: 3.4567
Enter number in index[2]: 6.778
Enter number in index[3]: 3.900
The total sum is:  37.48
The maximum number is: 23.34
The minimum number is: 3.46
```