# 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. |

# 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 INST0002 Programming 1 /INST0091 Introduction to Programming

# 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 / INST0091 Introduction to Programming
# Tutorial Sheet 02
# Week 02

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.

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[*]

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

Write a programme that will calculate the total number of calories burnt by taking user input of:

- Body weight (`bw`) in kg. (e.g. 71.5)
- Height (`h`) in meters (e.g. 1.73)
- Average speed (`s`) in meters per second (e.g. 1.4)
- Time walked (`t`) in minutes (e.g. 35)

All the values except time must be a `float` data type, while time should be an `int`. The formula for calculating the total calories burnt (`tcb`) is:

$$tcb = t * ((0.035 * bw) + (s^2 / h * 0.029 * bw))$$

Use relevant user prompt messages for requesting and reading user-specified data. Round the total calories burnt to 1 decimal place.

The output of the programme should be:

```
danny$ python3 calorie_alculator.py
Please enter bodyweight (kg): 65
Please enter height (m): 1.69
Please enter average speed (m/s): 1.5
Please enter time walked (min): 20
The number of calories burnt is: 95.7
```

## Exercise 2 [*]

*Learning Objectives: [A] print() function [B] data types and simple arithmetic operations, [C] typecasting data types*

You are required to solve the following simple equations and store the values of x and y in the corresponding variables: `3x - 7 = 14 and 2y + 10 = 16.`

You should then calculate what the remainder of dividing **`x`** by **`y`** should be and store the result in a variable called **`result.`** Finally, the programme should print the following:

```
The value of x is: _____
The value of y is: _____
The value of result is: _____
```

*The blanks in the output should be replaced with the relevant values.*

# Section K

## Exercise 3[*]

*Learning Objectives: [A] import libraries, input() function [B] data types and simple arithmetic operations.*

Write a programme that will take a `float` input from a user for the length of a radius (`r`), and calculate the diameter (`d = 2 * r`), surface area (`a = 4 * π * r²`) and volume (`v = 4/3 * π * r³`) of a sphere.

The programme should output the following:

```
The diameter of the sphere is: d = ______

The surface area of the sphere is: a = ______

The volume of the sphere is: v = _______
```

*The blanks in the output should be replaced with the relevant values.*

To ensure better usability of your programme, please include a prompt message such as "`Please enter the value for the radius: `".

Hint: You would need to import the **math** library package to help with value of Pi (`π`). Read further on how this is done.

## Exercise 4 [*]

*Learning Objectives: [A] using existing Python built-in functions, namely: `round(…)`; [B] input and type conversion.*

**Task 1:**

Write a program to compute the annual gross and net pay for a part-time worker. The application should request the employee to enter their hours and rate. Reflect and consider what data type to use here. The program should be able to calculate the hourly rate pay of the employee. Your program should compute the gross and net pay in two decimal floating points literal. Finally, the program should apply a 20% tax deduction on the employee net pay or take-home salary.

Example: For an employee to earn a monthly salary or pay of $1916.40 before any tax deductions, the employee would need to work for 120 hours in a pay period for that month, with an hourly rate of $15.97/hour.

**Vital information to use in writing this program.**

- The **Gross pay** is the amount an employee could receive prior to tax and other deductions.
- The **Net pay** is the amount employee takes home after all the deductions

# Section L

Hint: In order to arrive at the specified decimal places, you are required to use a Python built-in function known as **round()**. You are also required to use the **format()** function to print the results. You can create constant variables for the month (to store the number of months in the year) and for the tax rate.

The following will be the expected outcome:

```
danny$ python compute_net_pay_ex04.py
Please enter your hours: 120
Please enter your rate: 15.97
Your annual gross pay (gross income) is: $22996.8
Your monthly salary before tax deduction is: $1916.4
Your 20% tax deduction is: $383.28
Your take-home (net-pay) after the tax deduction is: $1533.12
```

### Task 2: [Optional]

Rewrite the same program using a different script (***hourly_rate_pounds_ex04.py***) and print the gross and net pay in pounds £ (GB) using Unicode characters as done above in dollars $ (US Dollars). Unicode characters are typically represented by a hexadecimal number, preceded by the **"\u"** escape sequence. You can read more about different Unicode and ASCII symbols and so on here: https://en.wikipedia.org/wiki/List_of_Unicode_characters

**Note:** This second task (Task 2) is only for your own practice. You **do not** need to submit the file as part of the submission for the exercise 4. You should only submit the first Task 1 for this exercise.

The following will be the expected outcome:

```
danny$ python compute_net_pay_ex04.py
Please enter your hours: 120
Please enter your rate: 15.97
Your annual gross pay (gross income) is: £22996.8
Your monthly salary before tax deduction is: £1916.4
Your 20% tax deduction is: £383.28
Your take-home (net-pay) after the tax deduction is: £1533.12
```

# Section M

## Exercise 5 [*] Car Company

*Learning Objectives: [A] learning to edit and execute Python programmes; [B] practice writing variables and expressions.*

Write a program for a car renter company, save your file as **cars.py**. The car program should contain 100 cars (*cars*). A car should have four spaces (***space_in_a_car***). The company has 30 drivers (***drivers***) and 90 passengers (***passengers***). Your program should calculate the number of cars driven (***cars_driven***) and the number of cars not driven (***cars_not_driven***). Your program should further calculate the capacity for a carpool (***carpool_capacity***) amongst the passengers. Lastly, your program should calculate and display the average number of the passengers per car (***average_passengers_per_car***).

Hint: Use appropriate variable names and comments in your code. The ' _ ' in the variable for example – *space_in_a_car* is called an underscore character. For calculating the carpool you should consider the number of **cars driven** and the number of **space(s) in a car**.

You program should display the following statement:

```
danny$ python cars.py

There are 100 cars available.

There are only 30 drivers available.

There will be 70 empty cars today.

We can transport 120 people today.

We have 90 passengers to carpool today.

We need to put about 3 passengers in each car.
```