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.

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 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 03
Week 03

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.

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[*] Income Tax

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

Write a program to calculate the gross annual income and tax deduction for an employee. The program should request employee to enter the income. Then should be able to calculate the **income tax**, display the **annual gross pay** (or annual gross income) entered by the user and the **monthly salary** (net pay) for the employee using the following pseudocode and conditional constructs.

Considering the various payment threshold and percentage tax deductions:

- if income tax threshold less that non-taxable income of £12,570 no tax deduction
- if income tax threshold is less than £50,000 the tax should be taken at 20% (basic rate)
- if income tax threshold is greater than or equal to £50,000 the tax should be taken at 40% (Higher rate)
- if income tax threshold is greater than or equal to £125,140 the tax should be deducted at 45% (additional rate)

Note: Let's assume we can calculate the monthly salary from the difference between the annual income and the annual tax.

Hint: The program should output the ***annual gross income***, ***monthly salary,*** and the ***annual tax deduction*** for the employee. You are required to display the output in 2 decimal places.

Additional Task [Optional]

Additional task for you. How would you write the program to compute the **total net salary** (take home salary or net pay) of the employee for that year after the **tax** deduction.

The output of the programme should be:

```
danny$ python3 income_tax_ex1.py
Please enter your income: 12276
Your annual income of £12276.0 is not taxable.
Your take home monthly salary is: £1023.0


danny$ python3 income_tax_ex1.py
Please enter your income: 48976
```

```
Your gross annual tax is calculate based on a **Basic Rate** at income of
£48976.0 as: £9795.2

Your take home monthly salary is: £3265.07


danny$ python3 income_tax_ex1.py


Please enter your income: 51283

Your gross annual tax is calculate based on a **Higher Rate** at income of
£51283.0 as: £20513.2

Your take home monthly salary is: £2564.15


danny$ python3 income_tax_ex1.py


Please enter your income: 126423

Your gross annual tax is calculate based on a **Additional Rate** at income
of £126423.0 as: £56890.35

Your take home monthly salary is: £5794.39
```

Exercise 2 [*] Exam Grade Average

Learning Objectives: [A] print() function [B] data types and simple arithmetic operations, [C] while loop, [D] for loop [E} conditional statements

Write a program that would compute the average grade of a number of students. The program should be able to allow the user to enter the number of exams grade each student would have. The program should continuously prompt for user input such as student's ID, grades and compute the average of the grades. At the end of the computation, the program should be able to ask the user whether they wish to enter the same number of exam grade for another student.

Pseudocode:

- The program should request user whether they want to add another student
- If response is 'Y' continue to compute the student's average

Section L

- If response is ‘y’ convert to upper case and continue to compute the average
- If the user enter ‘N’ or ‘n’ the program should terminate

Note: You should use control flow structure such as ***for-loop*** for the program (revise and use Lecture 3 as a guide)

The output of the programme should be:

```
danny$ python3 exam_grade_ex02.py
How many exam grades does each student have? 5
Enter student ID: STU20123098
Enter the exam grades.
Exam Grade 1: 82
Exam Grade 2: 78
Exam Grade 3: 90
Exam Grade 4: 68
Exam Grade 5: 70
The student with ID: STU20123098 has an average grade of 77.60
Enter exam grades for another student (Y/N)? y
Enter student ID: STU26589761
Enter the exam grades.
Exam Grade 1: 60
Exam Grade 2: 54
Exam Grade 3: 67
Exam Grade 4: 71
Exam Grade 5: 53
The student with ID: STU26589761 has an average grade of 61.00
Enter exam grades for another student (Y/N)? n
```

Section M

Exercise 3[*] Grade Computation

Learning Objectives: [A] import libraries, input() function [B] data types and simple arithmetic operations [C] while loop [D] multiple if else statements.

Write a program to compute information related to a sequence of grades obtained from the user. The program should be able to compute the number of passing and failing grades, computes the average grade and find the highest and lowest grade. Use your knowledge of decision or conditional statement covered in the lecture 3.

One of the main important logics of the program is that the program should be able to print the number of students that passed and those that failed. If the number of passing is more than the number of failing, the program should display "*Congratulations! Proceed to the next level*", otherwise display "*You will need to perform better*". But, if the number passing and the failing are the same, display "*You are in-between passing and failing!\nYou would need to study more!*"

Description

- The user should be able to enter grades continuously.
- Any score less than 40 should be failed grade.
- The program should display the number of failed, passed, average, lowest and highest grades.
- The program should continuously request for user input, and you should use a sentinel value to end the loop and request (review **lecture 3** for more about sentinel and so on).

Hint:

1. Initialise the variables e.g. passing & failing etc.
2. Take user input.
3. Initialise the pass mark to 40%.
4. Count how many passed and failed.
5. Print summary of the exam result

The output of the programme should be:

```
danny$ python3 grades_compute_ex03.py
```

```
Enter a grade or -1 to finish: 40
Enter a grade or -1 to finish: 67
Enter a grade or -1 to finish: 38
Enter a grade or -1 to finish: 90
Enter a grade or -1 to finish: 76
Enter a grade or -1 to finish: 89
Enter a grade or -1 to finish: 30
Enter a grade or -1 to finish: -1
The average grade is: 61.43
Number of passing grades is: 5
Number of failing grades is: 2
The maximum grade is: 90.00
The minimum grade is: 30.00
Remark: Congratulations! Proceed to the next level.

danny$ python3 grades_compute_ex03.py
Enter a grade or -1 to finish: 23
Enter a grade or -1 to finish: 90
Enter a grade or -1 to finish: 30
Enter a grade or -1 to finish: 38
Enter a grade or -1 to finish: 40
Enter a grade or -1 to finish: 37
Enter a grade or -1 to finish: 25
Enter a grade or -1 to finish: 50
Enter a grade or -1 to finish: -1
The average grade is: 41.62
Number of passing grades is: 3
Number of failing grades is: 5
```

```
The maximum grade is: 90.00
The minimum grade is: 23.00
Remark: You will need to perform better.

danny$ python3 grades_compute_ex03.py
Enter a grade or -1 to finish: 35
Enter a grade or -1 to finish: 80
Enter a grade or -1 to finish: 45
Enter a grade or -1 to finish: 30
Enter a grade or -1 to finish: 76
Enter a grade or -1 to finish: 25
Enter a grade or -1 to finish: 39
Enter a grade or -1 to finish: 50
Enter a grade or -1 to finish: -1
The average grade is: 47.50
Number of passing grades is: 4
Number of failing grades is: 4
The maximum grade is: 80.00
The minimum grade is: 25.00
Remark: You are in-between passing and failing!
You would need to study more!
```

Section P

Exercise 4 [*] Monthly Temperature

Learning Objectives: [A] for loop [B] input and type conversion [C] conditional statement [D] main function.

In this task you are required to write a simple program that would compute the month with the highest temperature reading in a year. Ideally, you should enter for each month the highest temperature for that month (e.g., Jan. 31^{0}C, Feb. 21^{0}C etc). The program should prompt user to enter the year and the subsequent months from January to December (as seen in the sample output).

Description

- The program should initialise the highest month by 1.
- Use a for-loop construct to decide the range (review lecture 3 for guide)
- Display the month with the highest temperature for that year.

The output of the programme should be:

```
danny$ python3 monthly_temperature_ex04.py
Enter year: 2023
Enter a temperature for Month[1]: 31
Enter a temperature for Month[2]: 34
Enter a temperature for Month[3]: 29
Enter a temperature for Month[4]: 39
Enter a temperature for Month[5]: 21
Enter a temperature for Month[6]: 12
Enter a temperature for Month[7]: 39
Enter a temperature for Month[8]: 40
Enter a temperature for Month[9]: 10
Enter a temperature for Month[10]: 20
Enter a temperature for Month[11]: 23
Enter a temperature for Month[12]: 21
Month 8 has the highest temperature for the year 2023¡
```

Section Q

Exercise 5 [*] Strictly Judges Program

Learning Objectives: [A] for loop [B] input and type conversion [C] conditional statement [D] main function.

In this task you are required to write a program to store the names and scores of four judges in a popular TV dance competition programme. You program should take the judges names, scores and store them in a dictionary container (***please revise and study more about dictionary covered in programming 1***). You should be able to test your program by printing the name of all the judges and their scores in a dictionary format (see sample output). Display the number of judges, the name and score of the first judge, the final judge and score. To test the functionality of your program.

Descriptive steps

- Initialise an empty dictionary called judges.
- Initialise an empty list called name.
- Initialise an empty list called score.
- Use a nested for loop to iterate over the judges' names and score (watch the lecture 3).
- Carefully consider the range that would be equivalent to the length of the elements to store in the lists.
- Populate the dictionary container with the names and scores of the judges from the two lists.
- The program should be able to retrieve all judges and scores from the lists.
- Print all the names and scores in the dictionary container (see output section).
- Restrict only scores between 1 and 10.
- Display the total score.

Functions

You are required to write this program embedded in a main function. You should create two user defined function called **lowScore()** and **highScore().** The ***lowScore()*** function should compute the lowest score and the ***highScore()*** function should compute the highest score. You should use this functions to help you in displaying the judges with the lowest score and those with the highest scores. If you wish you can print the name of the judge and the score by the side or you can display this separately (as see in the output section).

Note: You are not allowed to use Python built-in functions such as ***min()*** and ***max()*** for this task. Use the idea of the *minimumNumber*() and *maximumNumber*() functions from tutorial sheet 2 tasks (ex. 4 & 5) to help you solve this.

Hint: You can use multiple for loop for this program. Also, one way you could write this program is to create two lists, one for storing the names of the judges and the other for the scores. Thinking carefully, how you can add or update the lists (i.e. names and scores) and save this into the dictionary container (this is the tricky part). Study more on how to use the ***zip()*** method with lists and dictionary.

Additional Task [Optional]

Suppose you wish to restrict any score less than 5 to be stored in the dictionary container. How would you modify your program codes to take this into consideration during the program implementation. For example if a judge enters a score less than 5 such as 1, 2, 3 & 4. Your program should be able to set the value of the score given by the judge to **5 (e.g., score 3 should be 5 etc)**. *Try modifying your code to do this. Save the file as:* **judges_scores2_ex05.py**

The output of the programme should be:

```
danny$ python3 judges_scores_ex05.py

Welcome to strictly come dancing :)

   *** The results are in ***

Enter name of judge: Craig Revel Horwood

Enter score: 3

Enter name of judge: Motsi Mabuse

Enter score: 7

Enter name of judge: Shirley Ballas

Enter score: 7

Enter name of judge: Anton Du Beke

Enter score: 8

{'Craig Revel Horwood': 3, 'Motsi Mabuse': 7, 'Shirley Ballas': 7, 'Anton Du
Beke': 8}

There are  4  judges.

The first judge, Craig Revel Horwood ,  gave:  3

The final judge,  Anton Du Beke , gave:  8

The lowest score is: 3

The highest score is: 8

The judge with the lowest score is:  Craig Revel Horwood
The judge with the highest score is:  Anton Du Beke

The total score is: 25

danny$ python3 judges_scores_ex05.py

Welcome to strictly come dancing :)

   *** The results are in ***

Enter name of judge [1]: Craig Revel Horwood

Enter score [1]: 5

Enter name of judge [2]: Motsi Mabuse

Enter score [2]: 8

Enter name of judge [3]: Shirley Ballas

Enter score [3]: 7

Enter name of judge [4]: Anton Du Beke

Enter score [4]: 8

{'Craig Revel Horwood': 5, 'Motsi Mabuse': 8, 'Shirley Ballas': 7, 'Anton Du
Beke': 8}

There are 4 judges.

The first judge, Craig Revel Horwood , gave: 5

The final judge,  Anton Du Beke , gave: 8

The lowest score is: 5

The highest score is: 8

The judge with the lowest score is:  Craig Revel Horwood

The judge with the highest score is:  Motsi Mabuse

The judge with the highest score is:  Anton Du Beke

The total score is: 28
```

Section T

```
danny$ python3 judges_scores_ex05.py
Welcome to strictly come dancing :)
   *** The results are in ***
Enter name of judge [1]: Craig Revel Horwood
Enter score [1]: 11
The allowed score is between (1 – 10).

danny$ python3 judges_scores_ex05.py
Welcome to strictly come dancing :)
   *** The results are in ***
Enter name of judge [1]: Craig Revel Horwood
Enter score [1]: 0
The allowed score is between (1 – 10).
```