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 04
Week 04

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, methods 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[*] Character Counting

Learning Objectives: [A] input() function [B] String [C] conditional statements [D] loops [E] class & object creations [F] Constructor [G] method

Write a program to prompt user to enter a sentence with one or more special characters such as star(*) and dot(.). Your program should be able to count the number of ***stars*** and ***dots*** in the sentence and display the result (as seen in the sample output).

For this program to work, your program task should follow this logic or steps:

- Define a constructor with the required parameter lists and initialised the attributes within the body as **dotCharacter** and **starCharacter** using the concept of data abstraction (covered in lecture 4. Please revise to help you understand how this is done)
- Define a ***display()*** method that takes only a **self** parameter and continuously request user to type in a sentence.
- Add two **int** attributes, one to count the **stars** (**starCount**) and the other to count the **dots** (**dotCount**) within the display () method.
- Use two for-loop constructs, one for the **dotCounter** and the other for a **starCounter** characters. These for-loops should be able to iterate over the sentence and count their respective characters. For example the first for-loop should count how many dots in the sentence and the second for-loop should count how many stars in the sentence.

Creating object

- Use your knowledge of data abstraction to declare two attributes: **starCounter** and **dotCounter.** Initialise the starCounter with a character **(*)** and the dotCounter with a character **(.)**
- Create an object called **charObject**. Remember since the constructor has a few parameter lists, we will need to create the object based on this (covered in lecture 4).
- Finally, display the object (covered in the lecture 4).

Note: The sentence is a string with sequence of characters. The loop should terminate when the user enters the word **finish**.

Hint: Difference between Functions & Methods

In Python, methods and functions have similar purposes but differ in various ways. Functions are independent blocks of code written in our program which can be called from anywhere either within our program or outside. But methods are tied to objects or classes and need an object or class instance to be invoked. We could mostly access methods using the objects created from the class and also the class name can be used to invoke a class method from another class.

Section K

The output of the programme should be:

```
danny$ python3 character_counter_ex1.py
Input a line of characters: Today is a lovely day. Hope you are happy***.
The character '.' appears 2 time(s)
The character '*' appears 3 time(s)
Input a line of characters: ***I'm always very happy*** hope you are too.
The character '.' appears 1 time(s)
The character '*' appears 6 time(s)
Input a line of characters: finish
Goodbye! good to see you!!!
```

Exercise 2 [*] Animal Guesser Game

Learning Objectives: [A] print() function [B] random class, [C] class and object, [D] for loop [E} conditional statements [F] list

Write a program that would be able to guess an animal species and the size. This animal guesser game is similar to the number guesser game you completed in week 1.

Write the program using the following sequence of logic:

Constructor

- Create an animal class called "**Animal**" with two attributes, ***animal***(datatype string) and a ***size***(datatype int).
- Define a constructor that takes two parameter lists: an **animal species** and **size**. The constructor should assign the input to the corresponding attributes within the body/block.

Section L

Show() method

- The *show()* method should create a list of animal species and display these.

Guess() method

- Create a method within the class called **guess().** The method should initialise a count variable to zero, to allow you to count the number of guesses.
- Request user to guess the animal and the size. If the guesses are 8 or more, the program should display this message "*You've ran out of guesses. See you again!*".

compareMessage() method

- You should create a *compareMessage()* method, which takes two input parameter lists **animal** and **size**.
- The method should compare whether the animal guessed is the same as the randomly generated animal. If they are not the same, display "*Wrong animal species!!!*".
- Display "*Too small*" if the guessed animal size is smaller than the randomly generated size.
- Display "*Too large*" if the guessed animal size is larger than the randomly generated size.
- Otherwise display "*Perfect match! Congratulations!! You won!!!*"
- To stop the loop, the user should enter the word **done** for the animal guess and **any number less than or equal to zero** for the size.

Random generation

- Outside the class, generate the random animal species and the size.

Object creation

- Create an object called "animalObj". Display a doc text message "*My animal is size 1 – 10 and is one of these species:*" within the class (this is covered in Lecture 4).
- Use your class object to display the *show()* and *guess()* methods (Watch Lecture 4 to help you understand how to do this).

Hint: You can use the ***self*** keyword to specify the attribute inside the body of your constructor and methods.

Section M

The output of the programme should be:

```
danny$ python3 animal_guesser_ex02.py
My animal is size 1-10 and is one of these species:
Bear
Lion
Tiger
Monkey
Crocodile
Try to guess the animal species: Bear
Try to guess the size: 5
wrong Animal Specie!!!
Too large

Try to guess the animal species: Lion.
Try to guess the size: 4
wrong Animal Specie!!!
Too large

Try to guess the animal species: Tiger
Try to guess the size: 1
wrong Animal Specie!!!

Try to guess the animal species: Monkey
Try to guess the size: 1
perfect Match!
Congratulations! You won!!!
```

Section N

```
Try to guess the animal species: done
Thank you for playing the game!
Goodbye:)

danny$ python3 animal_guesser_ex02.py
My animal is size 1-10 and is one of these species:
Bear
Lion
Tiger
Monkey
Crocodile
Try to guess the animal species: Lion
Try to guess the size: 6
wrong Animal Specie!!!

Try to guess the animal species: Tiger
Try to guess the size: 4
wrong Animal Specie!!!
Too small

Try to guess the animal species: Monkey
Try to guess the size: 9
wrong Animal Specie!!!
Too large

Try to guess the animal species: Bear
Try to guess the size: 5
```

Section O

```
wrong Animal Specie!!!
Too small

Try to guess the animal species: Lion
Try to guess the size: 3
wrong Animal Specie!!!
Too small

Try to guess the animal species: Crocodile
Try to guess the size: 8
wrong Animal Specie!!!
Too large

Try to guess the animal species: Monkey
Try to guess the size: 6
wrong Animal Specie!!!

Try to guess the animal species: Lion
Try to guess the size: 6
wrong Animal Specie!!!
You've ran out of guesses. Please try again!!
```

Section P

Exercise 3[*] Rock, Paper, Scissor Game

Learning Objectives: [A] import libraries, input() function [B] method [C] while loop [D] if else statements [E] class and objects.

Write a program to compute a popular game of rock, paper, and scissors. In this task you will be playing against the computer system random generation. The program should randomly generate the 'rock', 'paper' and 'scissors' and the user would play against the computer to decide the winner.

One of the main important logics of the program is that the program should determine the winner of the game, either *human* or *computer* system.

Description

This program would require a few functionalities:

1) Create a class for the program e.g., **RockPaperScissorGame**..
2) *Human_choice()* method allows the program to get the user's choice and validate this.
3) *Computer_choice()* method generates the random choice to play with the human.
4) *Game_winner()* method compares the choices of the *human* and the *computer* to determine the winner. This method should take as input parameter list the **human choice** and the **computer choice**.
    - If the human choice is the same as that of the computer, the program should return "***it is a tie or draw***".
5) *Play()* method ask the human whether they want to play again against the computer (after the winner of the previous game).

Game Description:

A player who decides to play rock will beat another player who has chosen scissors ("rock crushes scissors" or "breaks scissors" or sometimes "blunts scissors") but will lose to one who has played paper ("paper covers rock"); a play of paper will lose to a play of scissors ("scissors cuts paper"). If both players choose the same shape, the game is tied and is usually replayed to break the tie.

Note: The program should be created in a class called **RockPaperScissorGame**. Ensure you convert the user input to lower case using the ***lower()*** method.

Hint: You can read more about this here: https://en.wikipedia.org/wiki/Rock_paper_scissors

Section Q

The output of the programme should be:

```
danny$ python3 rock_paper_scissors_ex03.py


Enter Rock, Paper or Scissors: rock
Oh yeah! Human win!!
Human player entered: 'rock' and computer chose: 'scissors'
Would you like to play again? (yes/no):yes

Enter Rock, Paper or Scissors: paper
It is a tie/draw! between human and computer :)
Human player entered: 'paper' and computer chose: 'paper'
Would you like to play again? (yes/no):yes

Enter Rock, Paper or Scissors: book
Invalid entry. Please enter either Rock, Paper or Scissors.
Thank you!

Enter Rock, Paper or Scissors: rock
Oh yeah! Human win!!
Human player entered: 'rock' and computer chose: 'scissors'
Would you like to play again? (yes/no):yes

Enter Rock, Paper or Scissors: scissors
It is a tie/draw! between human and computer :)
Human player entered: 'scissors' and computer chose: 'scissors'
Would you like to play again? (yes/no):yes
```

Section R

```
Enter Rock, Paper or Scissors: paper
Oh yeah! Human win!!
Human player entered: 'paper' and computer chose: 'rock'
Would you like to play again? (yes/no):yes

Enter Rock, Paper or Scissors: rock
Oh yeah! Human win!!
Human player entered: 'rock' and computer chose: 'scissors'
Would you like to play again? (yes/no):yes

Enter Rock, Paper or Scissors: scissors
It is a tie/draw! between human and computer :)
Human player entered: 'scissors' and computer chose: 'scissors'
Would you like to play again? (yes/no):yes

Enter Rock, Paper or Scissors: scissors
Oh no! Computer wins!!
Human player entered: 'scissors' and computer chose: 'rock'
Would you like to play again? (yes/no):yes
Enter Rock, Paper or Scissors: rock
Oh yeah! Human win!!
Human player entered: 'rock' and computer chose: 'scissors'

Would you like to play again? (yes/no):no
Thank you for playing the game of Rock, Paper and Scissor!
```

Section S

Exercise 4 [*] Student Record

Learning Objectives: [A] input() function [B] class amd object [C] method [D] constructor

Write a simple class program to display the detail of a student. You should create a constructor with appropriate parameter lists and initialise the attributes within the body. The program should create a student object with the following:

Description

- The program should request for student **identity**, **name,** and **department**.
- Create the student object using the information provided by the user (identity, name, department).
- Finally, display the information from a **student_detail()** method.

The output of the programme should be:

```
danny$ python3 monthly_temperature_ex04.py
Enter studentID: 12393049
Enter studentName: Danny Onah
Enter studentDept: Computer Science
Student ID:  12393049
Student Name:  Danny Onah
Student Department:  Computer Science
```

Section T

Exercise 5 [*] Employee Record and Income Increment

Learning Objectives: [A]class and object[B] print() function [C] return statement [D] class methods.

In this task you are required to write a program to create a class object for an employee record and pay increment. The program should define a constructor that would take some parameter lists (self, name, job, pay) and initialise them in the body of the constructor. You should set the **job** parameter input to **None** and pay to **zero** within the header of the constructor. Declare a class method called **lastName()** to display the last name of all employees. You should also create a class method called **payIncrement()** to increase the annual income by 10%. The pay increment method should take as input a self and a percent parameter list.

Descriptive steps

- *Create a class object 'danny' that holds the first name and last name as (Danny, Onah).*
- *Create another class object 'elaine' that holds the first name, last name, job as Lecturer and income pay as £39000)*
- *Display the last name for both employees.*
- *Display the surname, last name, job title and income of the employee 'elaine'.*
- *Finally, display the 10% increment for 'elaine' annual income.*

Note: For class methods they are created inside of a class. While functions are mostly stand-alone and not embedded inside a class. We can call a function inside a class also. But to call a class method the easiest way is to use the class name or object to invoke the method (read more about this).

The output of the programme should be:

```
danny$ python3 employee_income_ex05.py

Onah Jones

Surname:  Jones, Job Title: Lecturer, Income:  £39000

Pay raised by 10 percent is: £42900
```