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 07
Week 07

Academic Year: 2023/24

Set by Module Leader: 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 [] Dice Program

Learning Objectives: [A] class method; [B] classes [C] objects [D] importing a class [E] print function [F] constructor

Write a dice program to generate the outcome of a die. The **Die** class should have a constructor method that allows you to create the die with the default sides value of 6. The **RollDie** class should generate a random number between 1 and the sides value to stimulate the rolling of the die.

Description

The program should create instances of the Die class to generate different type of dice value such as four-sided die, six-sided die. The constructor is used to create the number of sides, this should be hardcoded in the parameter when creating the object.

The output of the programme should be:

```
danny$ python3 Die.py

The outcome of the dice roll is:  2
```

Exercise 2 [] Counting Coin Heads

Learning Objectives: [A] method; [B] conditional statement [C] import a class [D] input function [E] return statement.

Look at the Coin class covered in the lecture. Use the same Coin class exactly as appeared in the lecture 7. There is a simple function in the Coin class called flip(), that flips the coin and display whether it is heads or tails. Create a new class called **CountingHeads**. This class should be able to request for the number of tosses from the user and flips the coin. Then count how many heads identified after the flip is completed.

numberOfHeads

The program should define a class method called **numberOfHeads**. The class method should take as parameter the number of tosses requested from the user. Initialise a class variable called **toss** to zero. Within the class method, initialise a **headCount** variable to zero and create an instance of the **Coin** class that was imported into the **CountingHeads** class. Use a loop construct to iterate over the number of tosses and increment the **toss** variable. Declare a conditional statement to check how many HEADS identified when the coin flip is completed.

Section K

Program Requirements - pseudocode

a) request a number of tosses from the user and assign to variable `numTosses`

b) initalise a variable called `headCount` to zero

c) create a new object of type `Coin`

d) toss this coin `numTosses` and increment `headCount` by one each time the coin shows heads

e) finally, output the number of tosses and the number of heads.

You will need to implement all these steps and so on, for the program to work.

Hint: *You would need to create an object for the **CountingHead** class using a default constructor. You will also need to call the **numberOfHeads** class methods on the coin object passing the number of tosses entered by the user as the argument.*

The output of the program may look as follows:

```
danny$ python3 counting_heads_ex02.py
Enter the number coin toss needed: 9
The coin tossed  9  times with  5  heads.
```

Exercise 3[] Rolling Dice Game

Learning Objectives: [A] import random, input() function [B] classes [C] return statement [D] objects [E] methods [F] conditional statement[G] method call.

Using your knowledge from exercise 1, write a simple program about rolling dice game. The program should allow a player to enter a digit and compare the number with the generated numbers. If the number entered by the user is equal to the **sum** of the two randomly generated number, then display **"You win!"** otherwise, display **"You lose!"**.

You are required to use two classes **Dice** and **RollingDice** for this program.

Section L

Dice class

This class should declare a constructor method that takes one input parameter list called **generator** including the **self** parameter. The program should be able to initialise the generator attribute within the constructor with a randomly generated numbers inclusive of digit 6. You should pay special attention and consideration to the physical features of a typical dice. Define a method called **roll()** that takes a parameter list called **top** and also including the **self** parameter. The method should initialise the *top attribute* with the random number generated in the constructor. Create a getter or accessor method called ***getTop()*** that returns the *top attribute*.

RollingDice class

The **RollingDice** is the test class for the **Dice** class. The **RollingDice** class should define a method called **rollDice()** that should take a user input to guess the dice and create two dice objects '**diceX**' and '**diceY**'. Each of the dice object should take as argument the guess input from the user. In order to roll the dice, you would need to use the instances of the Dice class to invoke the ***roll()*** method. That is, your program should be able to display two randomly generated numbers and return these from the **getTop()** method. Finally, your program should be able to display the numbers and compare whether the sum of these numbers is **equal** to the user input.

Hint: You should use the objects created by the **Dice class** to generate these numbers. Think carefully about how you could do this or how this can be done.

Additional Task: Ensure the user input is between 1 and 12. Use a sentinel control loop to display a message "**Thank you for playing!**" and end the program if the user enters the number 0.

You program should display the following statement:

```
danny$ python3 RollingDice.py
Enter dice to guess:20
Please enter numbers between 1 and 12:

Enter dice to guess:5
*** dice rolling ***
Rolled:  6 and  3
You lose!
```

Section M

```
Enter dice to guess:6
*** dice rolling ***
Rolled:  6 and  3
You lose!

Enter dice to guess:10
*** dice rolling ***
Rolled:  4 and  6
You win!

Enter dice to guess:0
Thank you for playing!
```