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 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 06
Week 06

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 [*] Counting Word Program

Learning Objectives: [A] recursive class method; [B] conditional statement [C] base and standard case [D] classes [E] objects [F] input function.

This question relates to the material from Lecture 6, and is about using recursion to count the number of occurrences of a particular word in a given sentence, both of which have to be input from the user as requested by the program.

a) Review the lecture video from Week 6, concerning the program **countWord** that recursively counts all the words in an input sentence. (Your task below will be to adapt this program to recursively count only the occurrences of a specific word given by the user).

b) Execute and test the program with the word: "dog" and the sentence : "*A big dog is bigger than a small dog or a medium-sized dog, but smaller than a huge dog.*"

You should see the following input/output on screen:

Type in a single word to count: dog
Now type in a sentence, ending with a '.': *A big dog is bigger than a small dog or a medium-sized dog, but smaller than a huge dog.*

Output: *The sentence contains 4 instances of the word 'dog'.*

Test the program with some other words and sentences in order to understand how it behaves with different inputs.

remove_punctuation() function

The program should remove all punctuations such as full-stop (.) and commas (,) from the sentence. Write a function called **remove_punctuation()** to remove the punctuation from the sentence and from the target word. You should use or import the ***string*** library to help you with this task if necessarily. Ensure the sentence and the target word are free of any punctuation before checking whether there is any match.

c) ensure you convert both the ***sentence*** and the ***target word*** to lower case.

d) Write and complete a recursive function called **recursiveWordCount()**. This function should have one or two base cases and two recursive cases. The first base case should check whether a sentence exist in the program, if there is no sentence, this should return 0. The first recursive case should return 1 plus the ***recursiveWordCount()*** with it's arguments 'sentence' and 'target word'. Otherwise, return just only the ***recursiveWordCount()*** and the arguments passed into the function call.

Section K

The output of the programme should be:

```
danny$ python3 word_counter_ex01.py

Enter a sentence: A big dog is bigger than a small dog or a medium-sized
dog, but smaller than a huge dog.

Enter the word to count: dog.

The sentence contains 4 instances of the word 'dog'

danny$ python3 word_counter_ex01.py

Enter a sentence: I LIKE programming and I also LIKE Mathematics!

Enter the word to count: LIKE

The sentence contains  2 instances of the word 'like'

danny$ python3 word_counter_ex01.py

Enter a sentence: INST0004 is a very InteresTING module

Enter the word to count: interesting

The sentence contains 1 instance of the word 'interesting'
```

Exercise 2 [*] Factorial Number

Learning Objectives: [A] recursive function; [B] conditional statement [C] base and standard cases [D] input function [E] return statement.

Write a programme that will use a function to compute the Factorial number for a given index. Create a **main()** function in your program to take an integer index via user input (e.g. 3). Create a recursive function called **fact**() that would take a single parameter list ('numb'). Within the base case, write a conditional statement to return 1 if the index is equal to 1. In the standard case compute the recursive function and return the result using the formula below:

$$result\ = numb * fact(numb - 1)$$

Note: In the **main()** function, request user input. Invoke the **fact()** and passed the index into the function. Finally, display the factorial number.

Section L

The output of the program may look as follows:

```
danny$ python3 factorial_number_ex03.py

Enter number: 10

The factorial of 10 is: 3628800
```

Exercise 3[*] Scrabble Scores Game

Learning Objectives: [A] import libraries, input() function [B] dictionary [C] return statement [D] recursive function [E] methods.

The game of Scrabble assigns a numerical score to each word a player puts onto the Scrabble board. The score of a word is the sum of the Scrabble value of the letters in it. In playing the game of Scrabble, the Scrabble value of 'd' and 'g' is 2, the Scrabble value of 'b', 'c', 'm' and 'p' is 3, the Scrabble value of 'f', 'h', 'v', 'w' and 'y' is 4, the Scrabble value of 'k' is 5, the Scrabble value of 'j' and 'x' is 8, the Scrabble value of 'z' and 'q' is 10, and the Scrabble value of all other letters in the alphabet is 1. So, for example, the word "python" scores 3+4+1+4+1+1=14. You are required to use a dictionary container for the letters and values. The dictionary container has been created for you in the code snippet below.

scrabbleScore() method

Define a method called **scrabbleScore()** with a recursive definition that calculates the Scrabble score of its argument correctly.

Hint: You can use the methods *len(), slice()* the string index and substring. **len()** can be used in the base case of the definition to check if the string is of length zero, in which case it will have a Scrabble score of 0. The other two methods can be used in the standard case. Consider, for example, the word "python". The Scrabble score of "python" can be broken down (recursively) into the Scrabble value of 'p' plus the Scrabble score of "ython", the Scrabble score of "ython" can be broken down into the Scrabble value of 'y' plus the Scrabble score of "thon", etc. Note that getting the first character of the word "python" in index 0 returns 'p', and the substring of the remaining character (i.e. index[1:]) returns "ython" (Do read more about this).

Blanks in Scrabble

In Scrabble, a player can also use "blanks", represented by the underscore character '_ ', to represent any letter in a word, although blanks do not contribute to the Scrabble score. So, for example, the word "python" spelt "p_thon" scores 3+1+4+1+1=10. Create a condition in method scrabbleScore so that the program can also correctly deal with words spelt with one or more blanks as input.

Section M

<u>Hint:</u> This will involve adding another simple standard case to your recursive definition. Check whether this is needed for a dictionary containing alphabetical items and their respective values.

<u>numberOfVowels() method</u>

Add a recursively defined method **numberOfVowels**(...) to the **ScrabbleScorer** class that returns (as an ***int***) the number of vowels (‘a’s,‘e’s,‘i’s,‘o’s and ‘u’s) in its single argument of type *String*.

<u>Hint:</u> Counting the vowels can be thought of as a simple version of calculating the Scrabble score, but with vowels having a value of 1, and all other letters not contributing to the score.

<u>Dictionary</u>

Complete the code snippets below. The dictionary items are already created for you, you do not need to change this. You are required to use this dictionary keys and values to create the Scrabble game.

```python
#!/usr/bin/en python
# Author: Danny Onah
# Date: 4 Nov.23
# Time: 00:33AM GMT
""""
A program for a scrabble game using class and object.
""""
class ScrabbleGame:
    # global variable is required to use the dictionary items in methods
    # create a dictionary of letters with values
    points = {
            'd': 2, 'g': 2,
            'b': 3, 'c': 3, 'm': 3, 'p': 3,
            'f': 4, 'h': 4,  'v': 4, 'w': 4, 'y': 4,
            'k': 5,
            'j': 8, 'x': 8,
            'z': 10, 'q': 10,
            'a': 1, 'e': 1, 'i': 1, 'l': 1, 'n': 1, 'o': 1,
            'r': 1, 's': 1, 't': 1, 'u': 1
            #points for the word: python = 3 + 4 + 1 + 4 + 1 + 1 = 14
            }
    # complete the Scrabble game using the program listing in the tutorial sheet 6
```

Section N

You program should display the following statement:

```
danny$ python3 scrabble_scorer_ex03.py

Please type a word: python

The word 'python' is worth  14  points is Scrabble.

The word 'python' has 1  vowel(s)


danny$ python scrabble_scorer_ex03.py

Please type a word: p_thon

The word 'p_thon' is worth  10  points is Scrabble.

The word 'p_thon' has 1  vowel(s)
```

Exercise 4 [*] String Manipulation

Learning Objectives: [A] method [B] conditional statement [C] classes [D] objects [E] input function.

Write a program to manipulate a string word with at least six characters long. The program should be able to display the first character, the third character and the last character in the string or word. For example, if the user type in the word '**helloWorld'** , the program should display the first character as ' **h** ' and the third character as ' **l** ' and the last character as ' **d** '. The program should further display the second half of the word which is '**World**'.

The program should be written using a class called '**StringTester'**. Create an object using Python built-in constructor. Use the object to invoke a **display()** class method with appropriate parameter list.

Hint: Remember to ensure that the word typed in by the user should have at least 6 or more characters.

You program should display the following statement:

```
danny$ python3 string_tester_ex04.py

Type in a word with at least 6 letters: HelloWorld

The first character of the 'HelloWorld'  is:     H

The third character of the 'HelloWorld'  is:     l

The last character of the 'HelloWorld'  is:      d

The second half of the word  'HelloWorld' is:    World
```

```
danny$ python3 string_tester_ex04.py
Type in a word with at least 6 letters: Hello
The required length of word is 6 character or above.
Type in a word with at least 6 letters: HelloEvery
The first character of the 'HelloEvery'  is:     H
The third character of the 'HelloEvery'  is:     l
The last character of the 'HelloEvery'  is:      y
The second half of the word 'HelloEvery' is:    Every
```

Exercise 5 [*] Triangular Number

Learning Objectives: [A] recursive function; [B] conditional statement [C] base and standard case [D] classes [E] objects.

The triangular number is a very similar concept to a factorial. The nth triangular number, T_n, is the sum of all positive numbers equal to or less than n, i.e.

$$T_n = n + (n - 1) + \ldots + 1 \qquad [1]$$

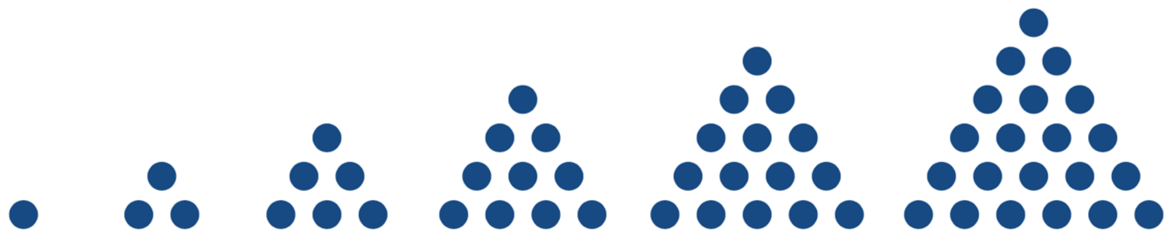


Figure 1: A graphical representation of the first six triangular numbers.

Section P

To see why these numbers are called triangular numbers see Figure 1 and the equation 1 to help you understand this mechanism (Read more about triangular number in mathematics).

Now create a class called **TriangularNumber**. The class should have a recursive method called **triangularGenerator**. This should take a single **int** argument ***number***. The method should return an ***int*** that is the triangular number of the input ***number***.

Note: The program should request an integer input from the user then output the triangular number of that input using the formular above.

Hint: Before thinking about coding, begin by deciding what the base case is, then decide what the standard case is in conceptual terms. Only then should you try to translate that to the code.

You program should display the following statement:

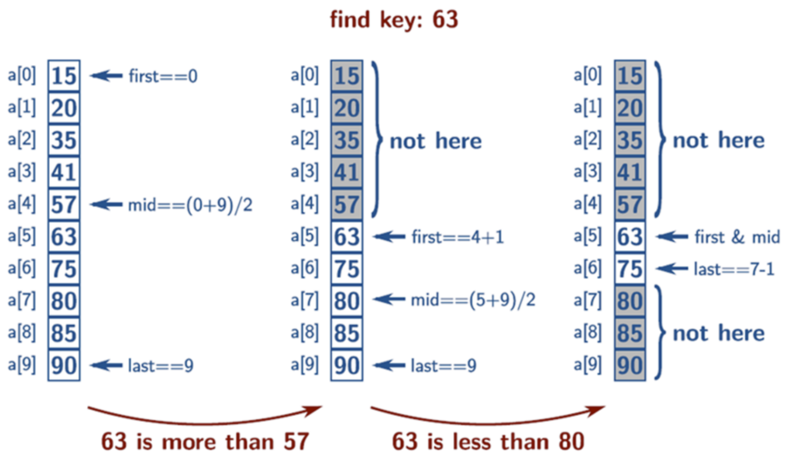
```
danny$ python3 triangular_number_ex05.py

Type in an integer: 1

The triangular number of '1' is: 1


danny$ python3 triangular_number_ex05.py

Type in an integer: 2

The triangular number of '2' is: 3

danny$ python3 triangular_number_ex05.py

Type in an integer: 3

The triangular number of '3' is: 6


danny$ python3 triangular_number_ex05.py

Type in an integer: 4

The triangular number of '4' is: 10


danny$ python3 triangular_number_ex05.py

Type in an integer: 5

The triangular number of '5' is: 15
```

```
danny$ python3 triangular_number_ex05.py
Type in an integer: 6
The triangular number of '6' is: 21
```

Exercise 6 [!] Recursive Binary Search

Learning Objectives: [A] recursive function; [B] conditional statement [C] base and standard case [D] binary search

Using the concept of recursive binary search procedure described in the lecture 6. Write a recursive function to search for the key value of **63** as illustrated in the figure below using the list [15, 20, 35, 41, 57, 63, 75, 80, 85, 90].



<u>Hint:</u> Apply your knowledge of recursive function to perform this task. Think carefully about how to set a proper base case and standard case. Watch the lecture 6 to help you complete the task.

<u>Non-Assessed:</u> Note that this task is part of your homework, and it is not part of the assessment. Therefore, you will NOT be assessed on this question.