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 05
Week 05

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 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[] Bank Account

Learning Objectives: [A] input() function [B] data types [C] classes and objects [D] insertion sort algorithm [E] control structure [F[data abstraction

Write a bank account program that would allow a customer to deposit and withdraw from their account. The program should restrict the customer from withdrawing more than their account balance. The program should prompt user to perform a transaction by entering their *account number, account balance*, *deposit* and *withdraw* ***amounts***. You should write this program with the knowledge and concept of **data abstraction**.

Constructor

Define a class constructor that takes three parameter lists: **self, account_number** and **balanc**e. The constructor should initialise the attributes.

my_deposit

Define a class method called ***my_deposit,*** that takes two parameter lists including the self-keyword and the *amount* to deposit. This method should check whether the amount deposited is more than zero and increment the current account balance. You should display the current account balance after the deposit.

my_withdraw

Define a method called *my_withdraw* that takes two parameter lists, one of which is the *amount* and the other *self*. Write a condition to check whether the amount is more than zero and less than or equal to the balance. If this is the case, decrement the balance. Otherwise, if the amount entered is more than the current account balance, display a message "*Withdraw amount more than balance*". ***Ask the customer whether they would like to make another transaction*** *if the response is yes*, ***proceed with the transaction***. Otherwise, terminate the transaction with a message "*Thank you for visiting the bank."*.

get_balance

The get_balance method should return the balance of the account.

get_transaction

The get_transaction method should return the account number of the customer performing the transaction.

Creating object

Create an account object with account number and balance. Use the account object to invoke the *my_deposit* method by passing the deposit amount input as argument. Invoke the *my_withdraw* method with the account object by passing as argument the withdraw amount.

Hint: At the end, the program should be able to display the initial account balance, customer number and current account balance after the withdraw transaction is completed.

Section K

The output of the programme should be:

```
danny$ python3 bank_account_ex01.py
Enter account number: 45555444
Enter account balance: 2000
Enter deposit: 500
Enter amount to withdraw: 1500
Account Balance: 2500.0
Customer with account number:  45555444 has a current bank account balance
of:  1000.0


danny$ python3 bank_account_ex01.py
Enter account number: 45555444
Enter account balance: 2000
Enter deposit: 500
Enter amount to withdraw: 3000
Withdraw amount more than balance.
Would you like to perform another transaction (Yes/No): yes
Enter account number: 45555444
Enter account balance: 2000
Enter deposit: 500
Enter amount to withdraw: 1200
Initial account balance:  2500.0
Customer with account number:  45665444 has a current bank account balance
of:  1300.0
```

Section L

```
danny$ python3 bank_account_ex01.py
Enter account number: 4587672
Enter account balance: 3000
Enter deposit: 200
Enter amount to withdraw: 4000
Initial account balance:  3200.0
Withdraw amount more than balance.
Would you like to perform another transaction (Yes/No): no
Thank you for visiting the bank!
```

Exercise 2 [] Insertion Sort Algorithm - Strings

Learning Objectives: [A] input() function [B] data types [C] classes and objects [D] insertion sort algorithm [E] control structure

In this task you are required to perform an insertion sort algorithm on a list of names ["Danny", "Elaine", "Adam", "Jane", "Peter", "Zack", "James", "Oliver"]. Your program should be able to sort the list of names in descending order from the highest character to the lowest in alphabetical order according to the names (covered in lecture 5). Create a class called ***InsertionSort*** with a method called ***insertion*** that would be used for computing the algorithm.

Descriptive step

- The program should use a for-loop to iterate over the list of names, taking into consideration the length of the list.
- Use a while loop to iterate over the list to create the condition for deciding how the insertion sort would be performed (covered in lecture 5).
- Finally display the sorted name list in descending order.

Creating object

- Create the list of names.
- Then create the object called ***insertionObj***
- Use the object to invoke the method.

Note: *This program should be written without any built-in functions such as sort() or sorted().*

Section M

Additional Task

Re-write the program to sort a list of names in a class in ascending order. Your program should request for user input and create the class instance object (using the knowledge from lecture 5).

The output of the programme should be:

```
danny$ python python3 implementating_algorithm_class.py
['Zack', 'Peter', 'Oliver', 'Jane', 'James', 'Elaine', 'Danny', 'Adam']
```

Exercise 3[] Insertion Sort Algorithm - Numbers

Learning Objectives: [A] input() function [B] data types [C] classes and objects [D] insertion sort algorithm [E] control structure

In this task you are required to perform an insertion sort on a list of numbers (1, 90, -12, 56, 99, 75, -20, 3, 39, 6, 71, 0). Your program should be able to sort the list of numbers in descending order from the maximum number to the minimum (covered in lecture 5). Create a class called ***InsertionSort*** with a method called i***nsertion***. Similar to the previous task but with number data type.

Descriptive step

- The program should use a for-loop to iterate over the list of numbers, taking into consideration the length of the list.
- Use a while loop to iterate over the list to create the condition for deciding how the insertion sort would be performed (covered in lecture 5).
- Finally display the sorted list in descending order.

Creating object

- Create the list of numbers.
- Then create the object called ***insertionObj***
- Use the object to invoke the method.

Note: *This program should be written without any built-in functions such as sort() or sorted().*

The output of the programme should be:

```
danny$ python python3 implementating_algorithm_class.py
[99, 90, 75, 71, 56, 39, 6, 3, 1, 0, -12, -20]
```