COM4018

Introduction to Programming

Date for Submission: Please refer to the timetable on ilearn

(The submission portal on ilearn will close at 14:00 UK time on the date of submission)

Assignment Brief

As part of the formal assessment for the programme, you are required to submit an **Introduction to Programming** assignment. Please refer to your Student Handbook for full details of the programme assessment scheme and general information on preparing and submitting assignments. The assignment brief will specifically give details and instructions for the assignment.

Module grade: Coursework 100%

Description: The assignment is given as three tasks, which are to be completed in sequence.

Note: Your work for this assessment needs to be provided in different ways –screenshots, flowcharts, and code in text**. Please ensure that you include your fully functional code in text in the appendix so it can be verified**.

Learning outcomes:

After completing the module you should be able to:

1. Develop an understanding of data structures and programming techniques in context of a programming language.
2. Demonstrate an understanding of how programs are developed i.e. from concept to development and testing.
3. Demonstrate an ability to write programs using appropriate structure and language rules.

Graduate attributes

4. Discipline Expertise: Knowledge and understanding of chosen field. Possess a range of skills to operate within this sector, have a keen awareness of current developments in working practice being well positioned to respond to change.

All learning outcomes must be met to pass the module.

Guidance

Your assignment should include a title page containing your student number, the module name, the submission deadline and the exact word count of your submitted document; the appendices if relevant; and a reference list in AU Harvard system(s). You should address all the elements of the assignment task listed below. Please note that tutors will use the assessment criteria set out below in assessing your work.

You must not include your name in your submission because Arden University operates anonymous marking, which means that markers should not be aware of the identity of the student. However, please do not forget to include your STU number.

Maximum word count: 3000 words

Please refer to the full word count policy which can be found in the Student Policies section here: Arden University | Regulatory Framework

The word count includes everything in the main body of the assessment (including in text citations and references). The word count excludes ***numerical data in tables, figures, diagrams, footnotes, reference list and appendices. All other printed words ARE included in the word count.***

Students who exceed the wordcount up to a 10% margin will not be penalised. Students should note that no marks will be assigned to work exceeding the specified limit once the maximum assessment size limit has been reached.

Scenario

You are required to create and test a new software system for a small shop loaning fishing and camping equipment for hire to customers.

The fishing and camping equipment available for hire and the cost is shown in figure 2. Customers can rent it from 9am to 6pm each day. The equipment can be hired from 9am of day 1 but should be returned by 2pm the next day irrespective of the time of collection on day 1. For each additional night a 50% discount is applied for each piece of equipment hired. If the equipment is returned after 2pm, it will still be counted as an additional night and extra 50% payment for each piece of equipment hired will need to be paid.

Your new software system should be able to record the details of the customers and equipment hired as shown in figure 3. It should also provide a report for the operator when required as shown in figure 4. For this, the new software system should open with a menu as shown below.

1. Enter details of customer and equipment hired
2. Create report
3. Exit

Figure 1

Tackle hire (9am day 1 - 2pm day 2)

Equipment	Hire Cost (each)
Day chairs	£15.00
Bed chairs	£25.00
Bite Alarm (set of 3)	£20.00
Bite Alarm (single)	£5.00
Bait Boat	£60.00
Camping tent	£20.00
Sleeping bag	£20.00
Rods (3lb TC)	£10.00
Rods (Bait runners)	£5.00
Reels (Bait runners)	£10.00
Camping Gas stove (Double burner)	£10.00

Figure 2

Customer and equipment details

Customer ID	Customer Name	Phone Number	House Number	Postcode	Credit/Debit Card	Equipment type with number
101	ABC	1234	1	ab123d	9876	Bed chairs –2, camping tent - 1

Figure 3

Earnings Report

Customer ID	Equipment	Number of nights	Total C	Returned on time (y/n)	Extra charge for delayed return
101	Bed chairs – 2 camping tent – 1	1	70	y	0

Figure 4

Assignment requirements:

- You must produce ONE single MS Word (.docx) document (the Document) for all tasks and TWO Python source (.py) files, one for Task 1 only and the other for both Tasks 2 and 3.
- Use your student number as the filename of the Document. Name your Python source files with your student number e.g. 24987654 as `24987654_T1.py` (for Task 1) and `24987654.py` (for both Tasks 2 and 3).
- You must therefore submit 3 separate files. DO NOT zip or compress your files in any way. Submission not in the correct file formats will result in ZERO (0) marks given.
- All flowcharts and screenshots must be legible without zooming. Screenshots should be annotated or supplemented otherwise with concise explanation.
- All source code should be appropriately commented.

- Input validation should be incorporated in all implementations wherever applicable.
- Importing any python modules is not allowed

Assignment Task

Task 1 – the main user menu

A. Provide a flowchart for an algorithm to display the program main menu as shown in figure 1 and process the corresponding options inputted by the user. Once an option is selected and the code for the option executed, the menu should be displayed again. The algorithm comes to an end only when the option for Exit is selected. More detailed requirements are as follows:
   1. The algorithm should include user input validation for different valid inputs as mentioned in figure 1.
   2. Common notations and conventions should be followed.
   3. At this stage, the algorithm should take the user input for menu options and just print a message depending on the option. When option 1 is selected, print `'Customer and hire details selected'`. When option 2 is selected, print `'Earnings report selected'`.

B. Write a Python program to implement the flowchart.

C. Provide sufficient screenshots of sample runs to demonstrate that your program works as expected.

(900 words equivalent)
(30 marks)
LOs:

Task 2 – hiring equipment

You will design the algorithm for option 1, implement it with Python and integrate it with the main user menu in Task 1. You must use suitable data structures to store data necessary as shown in figure 3 to accomplish this task. A minimum of 10 hires should be shown. **You should include records for each equipment type, different number of days, late return of equipment etc.** The data structures should be accessible to both Task 2 and Task 3. To better modularise the program, the functionality of this task will be put in a subroutine which will be invoked when the corresponding option is chosen from the main menu.

A. Provide pseudocode for the algorithm of the subroutine subject to the following requirements:
   1. The pseudocode should accept user inputs for hiring different equipment types and number of items of each type as mentioned in the Task summary.
   2. Ignore the tabular formatting details in the pseudocode. Just output the data required.
   3. Data should be entered using the keyboard as user input. These data should be stored in data structure(s) separate from those for read-only data.
   4. You are NOT required to include input validation in the pseudocode.
   5. Standard conventions and formats such as indentation, descriptive names for variables/subroutines, control structure constructs, operators, pseudo-keywords etc, should be followed.

B. Write a new Python program beginning with the menu such that when option 1 is chosen, the subroutine of this task is called. Implement the subroutine according to the pseudocode but remember to **add input validation** and cater for the **output format** specified. **Note: Copy the code you wrote in task 1 and alter this for when option 1 is chosen. Do not make any changes to the python program in task 1 py file.**

C. Provide sufficient screenshots of sample runs to demonstrate that your program works as expected.

(1200 words equivalent)
(40 marks)

Task 3 – Earnings report

You will design the algorithm for option 2, Earnings report, implement it with Python and integrate it into the code of Task 2. You will need to use the data stored in Task 2 to complete Task 3. Similar to Task 2, the functionality of this task will be put in a subroutine which will be invoked when the corresponding option is chosen from the main menu.

A. Provide a flowchart for the algorithm of the subroutine subject to the following requirements:
   1. The algorithm will make use of the data from Task 2.
   2. A report similar to figure 4 should be displayed, but the columnar formatting details can be ignored in the flowchart.
   3. The exact data to be displayed depends on the data entered so far with the subroutine of Task 2. If more orders are placed, subsequent report printing should show the increased earnings amounts.
   4. Standard notations and conventions should be followed

B. **Modify** the program code that you created in Task 2 so that when option 2 is chosen, the subroutine of this task is called. Implement the subroutine according to the flowchart, but make sure to cater to the **output format**.

C. Provide sufficient screenshots of sample runs to demonstrate that your program works as expected.

(900 words equivalent)
(30 marks)

End of questions

As technology and platforms may change, your module tutor will provide you with up-to-date details.

Formative Feedback

You have the opportunity to submit a draft report to receive formative feedback. You are encouraged to submit your assignment for feedback **once** and it is 30% of your entire submission. You, the student, are to choose 30%, **not the tutor**. The last day for hand-in is by Friday during Week 8. The Feedback is designed to help you develop areas of your work, encouraging academic skills and independent learning.

If you are a Distance Learning student, then you are encouraged to send 30% of your assignment for feedback by email to your tutor, no later than two weeks before your final submission week. Dates will be given to you by your tutor on a module-by-module basis.

No formative feedback will be given after the time specified above, either blended, or distance learning.

Referencing Guidelines

You **MUST** underpin your analysis and evaluation of the key issues with appropriate and wide ranging academic research, ensuring all cited literature is referenced using the AU Harvard system(s).

Follow this link to find the referencing guides for your subject: Arden Library

Assessment Criteria (Learning objectives covered - all)

Level 4 is the first stage on the student journey into undergraduate study. At Level 4 students will be developing their knowledge and understanding of the discipline and will be expected to demonstrate some of those skills and competences. Student are expected to express their ideas clearly and to structure and develop academic arguments in their work. Students will begin to apply the theory which underpins the subject and will start to explore how this relates to other areas of their learning and any ethical considerations as appropriate. Students will begin to develop self-awareness of their own academic and professional development.		
Grade	**Mark Bands**	**Generic Assessment Criteria**
First (1)	80%+	Outstanding performance which demonstrates the ability to analyse the subject area and to confidently apply theory whilst showing awareness of any relevant ethical considerations. The work shows an outstanding level of competence and confidence in managing appropriate sources and materials, initiative and excellent academic writing skills and professional skills (where appropriate). The work shows originality of thought.
	70-79%	Excellent performance which demonstrates the ability to analyse the subject and apply theory whilst showing some awareness of any relevant ethical considerations. The work shows a high level of competence in managing sources and materials, initiative and excellent academic writing skills and professional skills (where appropriate). The work shows originality of thought.
Upper second (2:1)	60-69%	Very good performance which demonstrates the ability to analyse the subject and apply some theory. The work shows a very good level of competence in managing sources and materials and some initiative. Academic writing skills are very good, and expression remains accurate overall. Very good professional skills (where appropriate). The work shows some original thought.
Lower second (2:2)	50-59%	A good performance which begins to analyse the subject and apply some underpinning theory. The work shows a sound level of competence in managing basic sources and materials. Academic writing skills are good, and expression remains accurate overall although the piece may lack structure. Good professional skills (where appropriate). The work lacks some original thought.
Third (3)	40-49%	Satisfactory level of performance in which there are some omissions in understanding the subject, its underpinning theory, and ethical considerations. The work shows a satisfactory use of sources and materials. Academic writing skills are limited and there are some errors in expression and the work may lack structure overall. There are some difficulties in developing professional skills (where appropriate). The work lacks original thought and is largely imitative.
Marginal Fail	30-39%	Limited performance in which there are omissions in understanding the subject, its underpinning theory, and ethical considerations. The work shows a limited use of sources and materials. Academic writing skills are weak and there are errors in expression and the work may

		lack structure overall. There are difficulties in developing professional skills (where appropriate). The work lacks original thought and is largely imitative.
Clear fail	29% and Below	A poor performance in which there are substantial gaps in knowledge and understanding, underpinning theory and ethical considerations. The work shows little evidence in the use of appropriate sources and materials. Academic writing skills are very weak and there are numerous errors in expression. The work lacks structure overall. Professional skills (where appropriate) are not developed. The work is imitative.

Marking Rubric:

Criteria and weighting	Outstanding 80% - 100%	Excellent 70% - 79%	Very Good 60% - 69%	Good 50% - 59%	Pass 40% - 49%	Poor 30 – 39%	Fail 0 – 29%
Task 1 Algorithm (flowchart) (40%) Max marks 12	Functional, comprehensive, **efficient** algorithm that is justified with all standard conventions that meets all requirements and **adds appropriate value** toward meeting the requirements.	Functional, **comprehensive** algorithm without logical errors and with error-proofing, justified, with all standard conventions that meets all the requirements.	Functional algorithm justified with all standard conventions and **with error-proofing** and without logical errors/flaws that meets all the requirements.	Functional algorithm **justified** with **all** standard conventions, **without logical errors** that meets **all** the requirements.	Functional algorithm **elaborated,** with **most** standard conventions that meet **most** of the minimum requirements.	**Functional** and/or **reverse-engineered** algorithm **presented** with **some** standard conventions that meet **some of the minimum** requirements.	**Non-functional** algorithm **provided** with **little or no standard conventions** that meets **few or none** of the task requirements. Significant **omission, or logical errors** in algorithm.
Task 1 Code in text (30%) Max marks 9	Effectively commented, succinct, and **efficient** code provided in text that **uses resources providently**, with error-proofing and without logical errors that implements the algorithm exactly and meets all the requirements	**Effectively** commented, **succinct** code provided in text with error-proofing, without logical errors, that implements the algorithm exactly and meets all requirements. Code makes **full** use of language features **productively**.	Fully commented code provided in text that implements the algorithm exactly **with error-proofing** and no logical errors and meets all the requirements. The code makes **some use of language features for productivity**.	**Fully** commented code provided in text that implements the algorithm **exactly** and meets all the requirements. The code has **little or no redundancy** and **no logical errors**.	**Mostly** commented fully functional code, in text, implements the algorithm **closely** and meets **most** of the minimum requirements. The code may have **some** redundancy.	**Functional** code with **some** comments**, in text,** implements the algorithm to **some extent,** meets **some of the minimum** requirements. There may be **significant redundancy** in code.	**Nonfunctional** Code provided **is without comments,** and/or **not in text**; has **errors** (syntax, logical), **does not** implement the algorithm and/or meets **few or none** of the requirements.
Task 1 Sample output screenshots (30%) Max marks 9	**Continuous** and comprehensive sample output illustrates the user interactivity and tests including fool-proofed inputs, while meeting all the requirements, and **providing testing quality assurance**.	**Comprehensive** sample output illustrates the user interactivity and tests invalid user inputs including error messages, while meeting all the requirements and without logical errors indicated through display prompts.	Sample output illustrates user interactivity and tests for all values **including invalid user inputs**, while meeting all requirements. All display prompts **including error messages** are appropriate, clear, unambiguous without logical errors.	Sample output illustrates user interactivity and tests **ALL** relevant values and meets **all** requirements. **All** display prompts included **appropriately**, are clear, unambiguous and **without logical errors**.	Sample output illustrates the user interactivity and tests for **most** relevant values, while meeting **most** of the minimum requirements. **Most** display prompts are included, are **clear** and/or **unambiguous**.	Sample output is **limited** to illustrate user interactivity, tests for **few** input values, while meeting **some** of the **minimum** requirements. **Some** display prompts are included and are **unclear** and/or **ambiguous**.	**Insufficient** sample output provided, which **does not** illustrate the user interactivity or functionality of code**. Limited or no** display prompts in the sample output.

Criteria and weighting	Outstanding 80% - 100%	Excellent 70% - 79%	Very Good 60% - 69%	Good 50% - 59%	Pass 40% - 49%	Poor 30 – 39%	Fail 0 – 29%
Task 2 Algorithm (pseudo code) (40%) Max marks 12	Functional, comprehensive, **efficient** algorithm that is justified with all standard conventions that meets all requirements and **adds appropriate value** toward meeting the requirements.	Functional, **comprehensive** algorithm without logical errors and with error-proofing, justified, with all standard conventions that meets all the requirements.	Functional algorithm justified with all standard conventions and **with error-proofing** and without logical errors/flaws that meets all the requirements.	Functional algorithm **justified** with **all** standard conventions, **without logical errors** that meets **all** the requirements.	Functional algorithm **elaborated,** with **most** standard conventions that meet **most** of the minimum requirements.	**Functional** and/or **reverse-engineered** algorithm **presented** with **some** standard conventions that meet **some of the minimum** requirements.	**Non-functional** algorithm **provided** with **little or no standard conventions** that meets **few or none** of the task requirements. Significant **omission, or logical errors** in algorithm.
Task 2 Code in text (30%) Max marks 9	Effectively commented, succinct, and **efficient** code provided in text that **uses resources providently**, with error-proofing and without logical errors that implements the algorithm exactly and meets all the requirements	**Effectively** commented, **succinct** code provided in text with error-proofing, without logical errors, that implements the algorithm exactly and meets all requirements. Code makes **full** use of language features **productively**.	Fully commented code provided in text that implements the algorithm exactly **with error-proofing** and no logical errors and meets all the requirements. The code makes **some use of language features for productivity**.	**Fully** commented code provided in text that implements the algorithm **exactly** and meets all the requirements. The code has **little or no redundancy** and **no logical errors**.	**Mostly** commented fully functional code, in text, implements the algorithm **closely** and meets **most** of the minimum requirements. The code may have **some** redundancy.	**Functional** code with **some** comments**, in text,** implements the algorithm to **some extent,** meets **some of the minimum** requirements. There may be **significant redundancy** in code.	**Nonfunctional** Code provided **is without comments,** and/or **not in text**; has **errors** (syntax, logical), **does not** implement the algorithm and/or meets **few or none** of the requirements.
Task 2 Sample output screenshots (30%) Max marks 9	**Continuous** and comprehensive sample output illustrates the user interactivity and tests including fool-proofed inputs, while meeting all the requirements, and **providing testing quality assurance**.	**Comprehensive** sample output illustrates the user interactivity and tests invalid user inputs including error messages, while meeting all the requirements and without logical errors indicated through display prompts.	Sample output illustrates user interactivity and tests for all values **including invalid user inputs**, while meeting all requirements. All display prompts **including error messages** are appropriate, clear, unambiguous without logical errors.	Sample output illustrates user interactivity and tests **ALL** relevant values and meets **all** requirements. **All** display prompts included **appropriately**, are clear, unambiguous and **without logical errors**.	Sample output illustrates the user interactivity and tests for **most** relevant values, while meeting **most** of the minimum requirements. **Most** display prompts are included, are **clear** and/or **unambiguous**.	Sample output is **limited** to illustrate user interactivity, tests for **few** input values, while meeting **some** of the **minimum** requirements. **Some** display prompts are included and are **unclear** and/or **ambiguous**.	**Insufficient** sample output provided, which **does not** illustrate the user interactivity or functionality of code**. Limited or no** display prompts in the sample output.

Criteria and weighting	Outstanding 80% - 100%	Excellent 70% - 79%	Very Good 60% - 69%	Good 50% - 59%	Pass 40% - 49%	Poor 30 – 39%	Fail 0 – 29%
Task 3 Algorithm (flowchart) (40%) Max marks 16	Functional, comprehensive, **efficient** algorithm that is justified with all standard conventions that meets all requirements and **adds appropriate value** toward meeting the requirements.	Functional, **comprehensive** algorithm without logical errors and with error-proofing, justified, with all standard conventions that meets all the requirements.	Functional algorithm justified with all standard conventions and **with error-proofing** and without logical errors/flaws that meets all the requirements.	Functional algorithm **justified** with **all** standard conventions, **without logical errors** that meets **all** the requirements.	Functional algorithm **elaborated,** with **most** standard conventions that meet **most** of the minimum requirements.	**Functional** and/or **reverse-engineered** algorithm **presented** with **some** standard conventions that meet **some of the minimum** requirements.	**Non-functional** algorithm **provided** with **little or no standard conventions** that meets **few or none** of the task requirements. Significant **omission, or logical errors** in algorithm.
Task 3 Code in text (30%) Max marks 12	Effectively commented, succinct, and **efficient** code provided in text that **uses resources providently**, with error-proofing and without logical errors that implements the algorithm exactly and meets all the requirements	**Effectively** commented, **succinct** code provided in text with error-proofing, without logical errors, that implements the algorithm exactly and meets all requirements. Code makes **full** use of language features **productively**.	Fully commented code provided in text that implements the algorithm exactly **with error-proofing** and no logical errors and meets all the requirements. The code makes **some use of language features for productivity**.	**Fully** commented code provided in text that implements the algorithm **exactly** and meets all the requirements. The code has **little or no redundancy** and **no logical errors**.	**Mostly** commented fully functional code, in text, implements the algorithm **closely** and meets **most** of the minimum requirements. The code may have **some** redundancy.	**Functional** code with **some** comments**, in text,** implements the algorithm to **some extent,** meets **some of the minimum** requirements. There may be **significant redundancy** in code.	**Nonfunctional** Code provided **is without comments,** and/or **not in text**; has **errors** (syntax, logical), **does not** implement the algorithm and/or meets **few or none** of the requirements.
Task 3 Sample output screenshots (30%) Max marks 12	**Continuous** and comprehensive sample output illustrates the user interactivity and tests including fool-proofed inputs, while meeting all the requirements, and **providing testing quality assurance**.	**Comprehensive** sample output illustrates the user interactivity and tests invalid user inputs including error messages, while meeting all the requirements and without logical errors indicated through display prompts.	Sample output illustrates user interactivity and tests for all values **including invalid user inputs**, while meeting all requirements. All display prompts **including error messages** are appropriate, clear, unambiguous without logical errors.	Sample output illustrates user interactivity and tests **ALL** relevant values and meets **all** requirements. **All** display prompts included **appropriately**, are clear, unambiguous and **without logical errors**.	Sample output illustrates the user interactivity and tests for **most** relevant values, while meeting **most** of the minimum requirements. **Most** display prompts are included, are **clear** and/or **unambiguous**.	Sample output is **limited** to illustrate user interactivity, tests for **few** input values, while meeting **some** of the **minimum** requirements. **Some** display prompts are included and are **unclear** and/or **ambiguous**.	**Insufficient** sample output provided, which **does not** illustrate the user interactivity or functionality of code**. Limited or no** display prompts in the sample output.