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. **LO1** Develop an understanding of data structures and programming techniques in context of a programming language.
2. **LO2** Demonstrate an understanding of how programs are developed i.e. from concept to development and testing.
3. **LO3** 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

An entrepreneurial couple manage a portfolio of 5 rental properties in the city. Each property has been mortgaged, and they aim to cover the mortgages through the rental payments from each property.

The properties need occasional maintenance and repairs. The costs for these repairs are added to the original purchase price of the property for bookkeeping. Table 2 provides typical entries for the rent collected (indicated positive) and expenses for any repairs/maintenance completed (indicated negative) for the 5 properties.

The couple would like a **computer program in Python** to manage the rents and the repairs for the properties and update the amended cost (original purchase price + expenses for repairs/maintenance) for each property.

You are tasked with designing and implementing the majority of features for this program.

Table 1. Property details.

Property#	Original cost	Residual mortgage
B12-3AB	153450	112345
B13-4CD	212130	180234
B14-5GH	120100	85980
B15-6JK	135230	101321
B16-7MO	183230	130234

Table 2. Typical entries for different properties.

Property#	Description	Amount*
B12-3AB	Rent received	760
B13-4CD	Replaced radiator in kitchen	-150
B13-4CD	Rent received	1060
B14-5GH	Repaired water leak in bathroom	-70
B14-5GH	Rent received	600
B12-3AB	Boiler serviced	-80
B15-6JK	Rent received	690
B16-7MO	Repaired washing machine	-120
B16-7MO	Rent received	920

B14-5GH	replaced bathroom flooring	-210
B15-6JK	Boiler serviced	-80
B12-3AB	Rent received	760

*Positive and negative amounts indicate income and expenditure, respectively.

Table 3. Sample summary output based on the entries provided.

Property#	**Original cost**	**Repairs**	**Amended cost**	**Residual mortgage**	**Rents**	**Rent as % of Mortgage**
B12-3AB	153450	80	153530	112345	1520	1.35%
B13-4CD	212130	150	212280	180234	1060	0.59%
B14-5GH	120100	280	120380	85980	600	0.70%
B15-6JK	135230	80	135310	101321	690	0.68%
B16-7MO	183230	120	183350	130234	920	0.71%
Total	804140	710	804850	610114	4790	0.79%

$$Amended\ cost\ =\ Original\ cost\ +\ Repairs$$

e.g. for property B14-5GH, Amended cost = 120100 + (70+210) = 120380

The program should have the following menu:

Rental management menu

1. Enter rental property details
2. Display summary for rentals
3. Exit

Figure 1. Program menu

Assignment Task

Task 1

1) Provide a flowchart to display the menu as 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 program should terminate only when the option for Exit is selected.
   a) Flowchart should include validation for user input (choice selection) to avoid both runtime and logical errors, assuming that the programming language used for implementation is a typed language (like Python).
   b) Standard notations and conventions should be followed in the flowchart.
   c) The algorithm should take the user input for menu options and execute the appropriate subroutine. When option 1 is selected, the subroutine `property_data` should be executed. When option 2 is selected, the subroutine `summary_data` should be executed.
   d) At this stage, each subroutine should only have a print statement to display a message, such as 'Enter property details' or 'Property summary'. No actual processing further than this is required for this task. The complete processing for each subroutine for option 1 and option 2 will need to be written in task 2 and task 3 respectively.

2) Write a Python program which is appropriately commented to implement **the flowchart**.
   a) The program should take the user input for menu options and execute the appropriate subroutine.
   b) The program **should not import any modules**.

3) Test your program by displaying the menu and selecting the various options when the menu is displayed.

4) Provide screenshots for the output of the program showing the correct execution of all menu options and how user input validation works

5) Provide the Python source code in the Appendix **as text** for verification.

(30 marks)
(LOs: 1,2,3,4)

Task 2

In this task you will design the algorithm for the option 1 subroutine `property_data`, implement it with Python, and test the implementation. You must also create suitable data structure(s) to store the details of the properties and their maintenance/rentals. The data structure(s) should be accessible for both Task 2 and Task 3. Refer to Table 1 and Table 2 for **sample** data.

1) In the fully developed application program, the property details such as those shown in Table 1 are to be managed by a dedicated subroutine. Since you are not tasked with developing that subroutine, you will:
   a) hardcode the data in Table 1 in some data structure(s) for testing the implementation you are responsible for.

Note: Although the hardcoded data will only have 5 properties as shown in Table 1, as sample data for testing, there should not be any restrictions or assumptions on the number of properties your algorithm can handle.

2) Provide pseudocode for the option 1 subroutine `property_data` .
   a) The pseudocode should accept user input for the rent collected and expenses for any repairs/maintenance completed for rental properties, similar to data as shown in Table 2. There should not be any restrictions on the number of entries that can be entered.
   b) The menu should then be displayed for further user selection.
   c) Include in your algorithm validation for user input to avoid both runtime and logical errors, assuming that the programming language used for implementation is a typed language (like Python).
   d) Standard conventions and formats such as indentation, capitalisation, descriptive names for variables/subroutines, keywords, operators, etc, should be followed in the pseudocode.
   e) The subroutine should **take user input from the keyboard.** Sample data are shown in Table 2. **The data should not be hardcoded**.
   f) The user input should be stored using some suitable data structure(s).

3) Implement the subroutine for Option 1 - `property_data` in Python **based on the algorithm as presented in the pseudocode**.
   a) The Python code should be appropriately commented.
   b) The subroutine **should not import any modules** in the code.
4) Test your implementation by embedding the complete Python code for option 1 - `property_data` in the program written in Task 1.
   a) You will need to enter at least 6 record values similar to the values from Table 2. Please include **your own values** when providing these entries.
   b) **All this data should be entered using the keyboard as user input.**
   c) Justify the values used for the entries you have provided on how they enable full testing of your full implementation. Your justification will supplement the screenshots mentioned below.
   d) You will also need to enter data which demonstrates input data validation.

5) Provide screenshots for outputs with the user input you have used. Supplement the screenshots with your justification mentioned above.

6) Provide the source code in the Appendix so that it shows ALL your implementation so far including that of `property_data` as accomplished for this task. All source code should be **in text** for verification.

(30 marks)
(LOs: 1,2,3,4)

Task 3

In this task you will design the algorithm of the option 2 subroutine `summary_data`, implement it with Python, and test the implementation. You will need to use the data stored in Task 2 to complete Task 3.

1) Provide a flowchart for the option 2 subroutine `summary_data`
   a) The flowchart should take the data from Task 2
   b) The summary table similar to one shown in Table 3 should be displayed when the option 2 is chosen.
   c) Then the menu should be displayed for further user selection.
   d) Standard notations and conventions should be followed in the flowchart.

2) Implement the option 2 subroutine `summary_data` in Python **based on the algorithm as represented by the flowchart**.
   a) The Python code should be appropriately commented.
   b) The subroutine **should not import any modules** in the code.
3) Test your implementation by embedding the complete code for `summary_data` in the program written so far in Task 1 and then Task 2.
4) You should use the same data entries that you used in Task 2 and take corresponding screenshots supplemented with justification for Task 2.
5) Provide screenshots for the output of this Task (Task 3).
6) Provide the source code in the Appendix so that it shows your FULL implementation, including that of `summary_data` as accomplished for this task.
7) At this stage, the Appendix should include the code **for all tasks** i.e. the complete source code of the whole program **in text** for verification.

(40 marks)
(LOs: 1,2,3,4)

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

Submission Guidance

Assignments submitted late will not be accepted and will be marked as a 0% fail.

Your assessment should be submitted as a **single Word document with the Arden cover sheet** (MS Word with your student number as the file name). Do not zip the file in any way.

The submission for each task should be presented under a relevant heading. Screenshots, algorithm, and inserts should be **legible**, and provided with suitable captions. Flowchart should be presented in an appropriate pictorial format using appropriate page size and orientation to make it legible.

The code should be clearly **presented in text** so that it can be copied and pasted into a suitable coding platform to be executed and verified. For more information, please see the “Submitting an Assignment - Guide” document available on the A-Z key information on iLearn.

The work submitted should cover the **procedural programming paradigm**, concepts covered in the iLearn lessons, and be within the scope of the module.

You must ensure that the submitted assignment **is all your own work** and that all sources used are correctly attributed. Penalties apply to assignments which show evidence of academic unfair practice. (See the Student Handbook which is available on the A-Z key information on iLearn.)

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.

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.