The Brief

Your task is to create a Vending Machine program using the Python programming language. The program should demonstrate your knowledge of programming and make use of the techniques introduced over the course of the module. Your application should be accompanied by a development document (1,000 - 1,500 words).

The Application

The Vending Machine must have the following features as a minimum requirement:

- A menu of drinks and snacks presented via the console. The number and range of items is up to you.
- A set of numbers or codes that the user can input to select a particular drink or snack.
- A way of managing money so the user can input any amount of money and have the correct change returned.
- A message that tells the user that a particular drink or snack has been dispensed.
- A message that tells the user how much change they have received.
- Comments in the code to explain key operations.

You may wish to add additional features to your Vending Machine to achieve higher marks. Below is an indication of some of these features, however you may also wish to come up with your own:

- A method of categorising items in the vending machine to improve the user experience (e.g. ‘Chocolate’ or ‘Hot Drinks’).
- A way of allowing users to buy additional items.
- Appropriate error checking to validate inputs and ensure the user has enough money
- An intelligence system for suggesting purchases. For example, if you buy a coffee, the vending machine may suggest that you buy biscuits.

- A stock system meaning the machine may run out of products

To achieve marks in the higher mark boundaries you should be aiming to implement the more advanced techniques covered in the module including use of functions and object oriented programming.

The Development Document

Your Vending Machine Python program must be accompanied by a Development Document of 1,000 - 1,500 words.

Please include the following elements (note the suggested word counts):

- <u>*Specification:*</u> A short explanation of what you have been asked to build and a list of features that your vending machine includes. This section should also include a link to your GitHub repository. *(100 words)*
- <u>*System Flowchart:*</u> A visual depiction of the logical operation of your Vending Machine. This should be accompanied by a short explanation (*50 words).*
- <u>*Technical Description & Walkthrough:*</u> For this section you should provide a video which includes a walkthrough of your program running as well as a technical breakdown of your code. This technical breakdown should explain how the key features of your program have been implemented via code. The video technical description & walkthrough contributes to the overall word count at approx. 100 words per minute. The anticipated length of the video should be between 5 - 8 minutes, anything longer than 10minutes will not be reviewed and may result in a mark penalty. *(500 - 800 words)*
- <u>*Critical Reflection:*</u> This should describe what aspects of your Vending Machine you find compelling, what could be improved, and what programming skills you need to learn to make such improvements. *(250 words)*
- <u>*Appendix:*</u> A copy of your code should be included in an appendix at the end of your documentation. (*Not included in word count)*

You may wish to use MS Word of other word processor when drafting sections of your development document. However the final version must be written in the assignment submission portal. <u>Please do not submit a word document it will not be marked.</u>

Deadline

The deadline for the Utility App is **10th January, 2025, 11:59am**

Mark penalties may be applied to late submissions without prior approval of an extension. Please ensure that you prepare and submit your work in good time to allow for any issues that may arise.

Assessment feedback will be returned within 15 working days from the deadline date for submission.

Deliverables

The deliverables for this assignment are as follows:

- The Application: The Python code (e.g. main.py file) and any supplementary assets required to run your vending machine. This should be submitted to Utility App folder in your Github repository.
- The Development Document including walkthrough video (1,000 - 1,500 words). This should be written and submitted via the submission portal.

Submission

Please follow the submission instructions below. Work that is submitted incorrectly may not be accepted or could incur a points penalty.

Before submitting have you…

- Checked that any digital work is functioning as expected?
- Spell-checked and grammar-checked any written work that accompanies your digital work? Please make an appointment with the Writing and Learning Centre or speak to your tutor if you are experiencing challenges in this area.
- Formatted your written work to the specification below?
- Referenced all sources of information accurately? Please refer to www.citethemrightonline.com (Harvard) for guidance.

Your development document must be submitted via the submission portal and code submitted via your GitHub repository. Please adhere to the following method:

- Check your vending machine is working as expected.
- Commit and **push** the final version of your code to the Utility App folder in your repository assignment (ensure you push all the files required to run your vending machine. In many cases this may just be a single .py file).
- Locate the Utility App submission portal in the assessment folder and write your development document with the submission box ensuring you include all the elements specified on the brief.
- Submit your work.

You can save and return to work in progress as you write the development document, however please note once you click submit you cannot edit the work. Should you need to make any alterations to work that has been 'submitted' you will need to start a new attempt. Only your final attempt submitted before the deadline will be marked.

You may wish to use MS Word of other word processor when drafting sections of your development document. However the final version must be written in the assignment submission portal. Please do not submit a word document it will not be marked.

Only code pushed to your Github repository before the assessment deadline will be marked. Ensure you give yourself enough time before your final push.

Format

All written work must conform to university styling and submission guidelines. They must:

- Contain appropriate in-text citation that supplies an accurate list of references.
- Be accurate in referencing. See Bath Spa guidelines and the Harvard system described at www.citethemrightonline.com.
- Be accurate in spelling and paragraphing.

Use of online sources

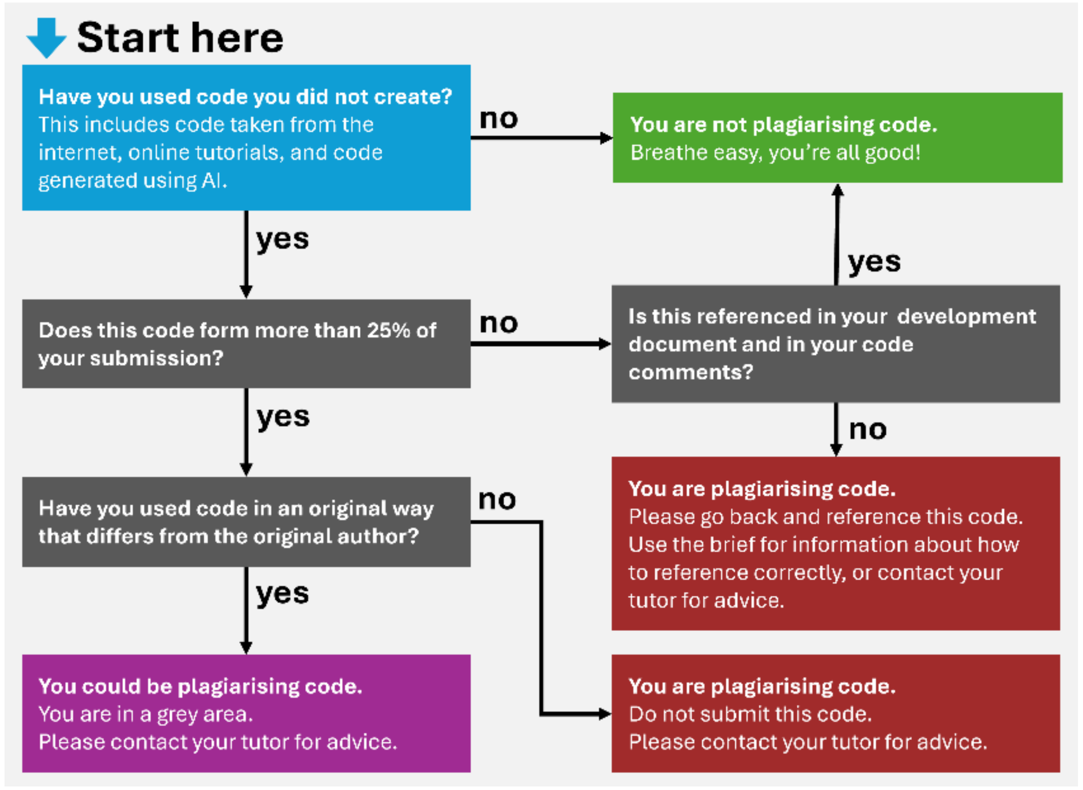
If you take any code from online resources (including Generative AI tools) to support the development of your program you should fully acknowledge their use by referencing the source in both code comments and in a reference list at the end of your development document.

- Code Comments: A reference in the code comments should be added ahead of the section of code that has been taken or heavily developed using online resources.
- *For standard online sources you should include the title of the webpage and full URL.*
- *For Generative AI tools you should include the name of the tool and the prompt provided to generate the code.*
- Reference List: A full reference list should also be included at the end of your development document. These references should be in alphabetical order.

- *For standard online sources you should follow the Harvard standard for websites*
- *For Generative AI tools you should follow the Harvard standard for generative AI*

If you are unsure of how to appropriately reference please consult your tutor. Failure to appropriately acknowledge use of online resources may result in an academic misconduct accusation.

Also be aware that use of online resources to develop your code should be kept to a minimum and excessive use may be deemed academic misconduct. The flowchart below is offered as guidance to what is and is not acceptable. Again if you are unsure please consult your tutor.


Start here
Have you used code you did not create? This includes code taken from the internet, online tutorials, and code generated using AI.
no
You are not plagiarising code. Breathe easy, you're all good!
yes
yes
Does this code form more than 25% of your submission?
no
Is this referenced in your development document and in your code comments?
yes
no
Have you used code in an original way that differs from the original author?
no
You are plagiarising code. Please go back and reference this code. Use the brief for information about how to reference correctly, or contact your tutor for advice.
yes
You could be plagiarising code. You are in a grey area. Please contact your tutor for advice.
You are plagiarising code. Do not submit this code. Please contact your tutor for advice.

Marking Criteria

Assignment S2: Utility App will be marked against the following criteria:

1. System Design - 15%
2. Technical Implementation - 50%
3. User Experience - 15%
4. Documentation - 20%

Criteria	Weighting	Mark Range Description	Mark Range
System Design	**15%**	A very limited design that does not adhere to the brief's specification	0 - 19 (Low Fail)
		A poor system design that omits key features. The vending machine may not be fit for purpose.	20 - 39 (Fail)
		A basic design that meets the minimum requirements stated in the brief.	40 - 49 (Third)
		A fair design that meets the minimum requirements to a suitable standard. Additional features are limited in number and/or complexity (e.g. product categorisation)	50 - 59 (2:2)
		A good design that meets the minimum requirements well and presents a number of additional features	60 - 69

			(2:1)
		Very good system design that provides several additional features. The design of these additional features extends their complexity beyond base requirements.	70 - 79 (First)
		Excellent system design that has many additional features including ones beyond those listed on the brief.	80 - 89 (High First)
		Beyond expectations for this level of study.	90 - 100 (Outstanding)
Technical Implementation	**50%**	A very limited implementation that demonstrates little evidence of programming skill.	0 - 19 (Low Fail)
		A poor implementation that contains errors. Limited programming techniques deployed. May not compile. Coding Conventions have not been observed (comments, indentation, camelCase etc).	20 - 39 (Fail)
		A basic implementation that deploys limited programming techniques. The program runs but may not function as expected. Coding Conventions have been observed to a limited degree (comments, indentation, camelCase etc)	40 - 49 (Third)
		A fair implementation that deploys several key programming techniques. Code is functional but may have minor errors in its implementation and likely suffers from duplication and other code inefficiencies. Coding Conventions have been observed to a fair standard (comments, indentation, camelCase etc), although there is room for improvement	50 - 59 (2:2)

		A good implementation that demonstrates an the fundamental programming techniques covered in the module including arrays and functions. Code has some minor duplication that could be improved with use of more advanced techniques or greater efficiency. Coding Conventions have been observed (comments, indentation, camelCase etc), with only minor errors present	60 - 69 (2:1)
		A very good implementation that successfully deploys appropriate programming techniques to minimise code duplication including the effective use of functions that pass and return values. Usage of OOP techniques may also be evident. Code is very well structured and coding conventions have been observed without error (comments, indentation, camelCase etc).	70 - 79 (First)
		An excellent implementation that demonstrates the range of techniques covered in class including those more advanced in nature. Use of techniques beyond the scope of the module may also be evident. Code structure and functionality is high quality. Coding Conventions have been observed with precision and comments are detailed (comments, indentation, camelCase etc)	80 - 89 (High First)
		Beyond expectations for this level of study.	90 - 100 (Outstanding)
User Experience	**15%**	A very limited user experience that may not be operable.	0 - 19 (Low Fail)
		A poor user experience that presents minimal instructions to the user. Application is operable but provides erroneous information.	20 - 39 (Fail)

		A basic user experience with limited clarity. May not reveal all necessary information to the user and/or provides ambiguous instructions.	40 - 49 (Third)
		A fair user experience that provides key information but lacks detail and precision (e.g. missing £ signs or decimal placed outputs)	50 - 59 (2:2)
		A good user experience that is accurate throughout. Console outputs are on the whole clear, yet there is room for refinement.	60 - 69 (2:1)
		A very good user experience that provides clear and detailed information to the user. Effort has been invested into the presentation of the console.	70 - 79 (First)
		An excellent user experience that implements techniques beyond those taught in class.	80 - 89 (High First)
		Beyond expectations for this level of study.	90 - 100 (Outstanding)
Documentation	**20%**	Very limited documentation that provides little or no insight into the development process. Required sections are missing or structure is unacceptable.	0 - 19 (Low Fail)
		Poor documentation that provides little insight into the development process. Content is unclear or poorly structured. Required sections may be missing.	20 - 39 (Fail)
		Basic documentation that provides some insight into the development process. One or more sections may be limited in depth or clarity and their is likely too much	40 - 49

		emphasis on description rather than explaining the techniques used and decisions made during development.	(Third)
		Fair documentation that provides key insights into the development process, though some sections could provide more depth or further details. Content is well structured and is generally communicated clearly.	50 - 59 (2:2)
		Good documentation that provides sound insights into the development process. The techniques used are clearly understood and decisions made are contextualised well. The critical reflection contains relevant insights. Content is well structured and communicated clearly with only minor flaws	60 - 69 (2:1)
		Very good documentation that clearly and concisely documents the development process. The critical reflection contains thoughtful takeaways. Content is well presented and is communicated clearly.	70 - 79 (First)
		Excellent documentation that provides astute insights into the development process. Content is very well presented and communication is highly effective.	80 - 89 (High First)
		Beyond expectations for this level of study.	90 - 100 (Outstanding)

Intended Learning Outcomes

ILO	Assessed
An understanding of the key features of programming including input, output, maths, sequence, iteration and repetition.	
The application of coding conventions to ease the review, maintenance, troubleshooting and debugging of developed software.	☐
The successful implementation of a prototype system that is driven by procedural programming techniques.	☐
An ability to specify applications, discuss their technical implementation and reflect critically on the results.	☐