

Supplementary Assessment Period COM161 Coding Project (Python)

Submission Deadline: noon on Thursday 27th November Week 10

This Assignment contributes **100%** to the overall assessment mark.

Feedback: feedback will be provided following the meeting of the Board of Examiners.

This is an **individual** assignment. When submitting your assignment you are agreeing to the following statement:

I declare that this is all my own work. Any material I have referred to has been accurately referenced and any contribution of Artificial Intelligence technology has been fully acknowledged. I understand the importance of academic integrity and have read and understood the University's General Regulation: Student Academic Integrity and the Academic Misconduct Procedure. I understand that I must not upload my work before, during or after submission to any unapproved plagiarism detectors or answer sharing platforms, or equivalent, and that only University-approved platforms should be used.

The University Academic Misconduct Policy on **plagiarism** is available at:
<https://www.ulster.ac.uk/student/exams/cheating-and-plagiarism>

Car Park Management Program

Overview

This assignment requires you to write a basic Python program that manages details of cars currently parked in a local car park. The program will use simple text files to store data and perform basic operations to manage parking. You are required to document testing for the program and provide a video walkthrough demonstrating the program execution and detailed code appraisal.

Requirements

1. Data Files (see end of spec for templates)

- SPACES.txt: Contains details of parking spaces (Space ID, Location, Type e.g., Standard/Disabled/EV).
- CARS.txt: Contains car details (Registration, Owner Name, Contact).
- PARKED.txt: Contains details of current parked cars (Space ID, Registration, Time In, Expected Time Out).

2. Program Overview

- Read data from the provided text files to initialise the program.
- Store spaces, cars, and parking record information in data structures of your choice.
- Allow a car to be parked in an available space.
- Allow a car to leave the car park (update / remove the record of the parking).
- Allow the user to view all cars currently parked.
- Allow the user to view all free spaces.
- Write the list of currently parked cars back to PARKED.txt when the program exits.

Core Functional Requirements

1. Initialisation

- Read data from SPACES.txt, CARS.txt, and PARKED.txt into data structures of your choice.

2. Park Car

- Input: Car Registration and required duration (to nearest 15 mins e.g. 90mins).
- Validate that the car exists in CARS.txt.
- Check for a valid and available space in the carpark.
- Timestamp entry time and date.
- Create a parking record and add it to the list of parked cars.

- Ensure EV or Disabled spaces can only be used by appropriate cars.
- 3. Leave Carpark**
 - Input: Car Registration or Space ID.
 - Validate if the car is currently parked.
 - Remove the corresponding record from the list.
 - 4. View Car Parking Status**
 - Display all cars currently parked.
 - Display all free spaces in the car park.
 - 5. Exit Program**
 - Save the current list of parked cars to PARKED.txt.

Video Walkthrough Guidelines – 5 minutes in length

You are required to produce a podcast, lasting no more than 5 minutes, which provides a walkthrough, **evidencing** each program requirement outlined above. A detailed guide, outlining what the podcast **needs to contain**, can be found at the end of this specification. Vodcasts that go beyond the time limit will be penalised. Vodcasts should be encoded to mp4 format. **The vodcast component is essential and mandatory component of the assessment. Failure to accompany the code submission with a vodcast will result in a mark of zero.**

The video walkthrough should comprise:

A. Program Demo Walkthrough

Evidence of completeness, confidence in operation, testing and awareness of any limitations.

1. Run your program in Visual Studio Code or PyCharm or IDLE.
2. Present an overview of the main menu options available.
3. Demonstrate each option, discussing the program output to confirm the correct operation.
4. Discuss the approach that you adopted to debug and test the operation of the program.
5. Demonstrate known issues / bugs that exist within the program and indicate whether these were due to technical complexity, lack of time or some other reason.
6. Provide self-assessment of how well you feel you have achieved the main elements of the assignment.

B. Source Code Walkthrough

Application of concepts and evidence of understanding of code structure and function.

1. Present an overview of the code, highlighting the total number of lines that it contains and the number of functions you have implemented.
2. Discuss the approach adopted to ensure the code is well organised and structured.
3. Discuss the code / operation behind each menu feature presented within the vodcast.
4. Evidence the use of commenting of code with your program.
5. Discuss challenging code segments or perhaps segments that you have commented out, which you found difficult to implement.
6. Discuss any validation or exception handling that has been implemented within the program.

Submission

Submit the following files in the SAP area in the SAP folder for the COM161 module on Blackboard by:
12:00 noon Thursday 27th November 2025.

1. All source code in a zip file.
2. A Word document containing your test cases and results.
3. A video walkthrough, 5 minutes in length.

Note: You **must** ensure that the submitted material can be uncompressed, opened and executed using a Python Interpreter and that Screencasts conform to mp4 format. Corrupted files will be treated as a non-submission.

This is an individual assignment and all work submitted must be your own.

Indicative Assessment Criteria

Marks will be awarded based upon an assessment of the evidence provided within the recorded vodcast, only according to the following criteria.

ASSESSED COMPONENT
APPLICATION OF CONCEPTS [30 MARKS] <ul style="list-style-type: none">• Appropriate input / output• Implementation of Decision structures• Implementation of Data structures
PROGRAM EXECUTION [30 MARKS] <ul style="list-style-type: none">• Correctness & completeness of features e.g.<ul style="list-style-type: none">◦ Park car functionality.◦ Car leaving functionality.◦ View parking functionality.
PROGRAM STRUCTURE [20 MARKS] <ul style="list-style-type: none">• Evidence of a formal program design e.g. flowchart or pseudocode• Implementation of appropriate functions / methods• Code style, including naming of variables• Usage of comments and overall readability
EVIDENCE OF TESTING [10 MARKS] <ul style="list-style-type: none">• Evidence presented within vodcast• Implementation of exception handling
OTHER FACTORS [10 MARKS] <ul style="list-style-type: none">• Adherence to vodcast guidelines• Clarity & depth of knowledge demonstrated• Awareness of program limitations
TOTAL MARKS AVAILABLE [100 MARKS]

Grading Rubric

	Poor	Satisfactory	Good	Excellent	Mark
APPLICATION OF CONCEPTS [30 MARKS]	<p>Lacking fundamental processing components (structures + file/io)</p> <p>Not able to discuss implementation or defend choices. (0-11)</p>	<p>Appropriate decision & data structures but some issues (e.g. manipulation of original file)</p> <p>Broad discussion of implementation with satisfactory defence of choices made.</p> <p>Perhaps overuse of GenAI (12-16)</p>	<p>Good file processing, robust handling of menu options through suitable structures.</p> <p>Good understanding of the concepts implemented demonstrate.</p> <p>Concepts demonstrated are central to the module teachings. (18-20)</p>	<p>Extensive use of taught concepts (structures, file IO, functions, string manipulation,)</p> <p>Use of multiple data structures (e.g. list, dict). Detailed understanding and rationale for the concepts. (21)</p>	
PROGRAM EXECUTION [30 MARKS]	<p>No functionality evidenced, unable to demonstrate working program. (0-11)</p>	<p>Program executes but only meets 1 or at most 2 features e.g. parking / leaving a space and / or provides some outputs but inaccurate.</p> <p>Some instability observed during execution. (12-16)</p>	<p>Program executes as expected with at most some specific instability e.g. crashing on full carpark. This instability should be accompanied by oral description of issue(s)</p> <p>Overall, good program correctness. (18-20)</p>	<p>Program executes with all expected functionality.</p> <p>Program features are provided and demonstrate excellent understanding of the concepts.</p> <p>Clear and concise description and understanding of each executed feature is evidenced. (21+)</p>	
PROGRAM STRUCTURE [20 MARKS]	<p>No consideration of program planning / design.</p> <p>Lack of appropriate functions, poor variable names and lacking comments. (0-7)</p>	<p>Evidence of some high-level program design. Functions used with scope for further refinement.</p> <p>Variable names are largely appropriate. Sparsely commented. (8-11)</p>	<p>Evidence of planning and understanding of program flow.</p> <p>Appropriate functions for file IO and each menu option.</p> <p>Clear, concise naming of variables & good use of comments. (12-13)</p>	<p>Flowchart (pseudo code) planning + design evidenced.</p> <p>Functions logically sequenced & some evidence of these having been optimised to reuse other functions to minimise code redundancy.</p> <p>Coding style, variable naming and use of comments meeting elements of PEP8 standards. (14+)</p>	
EVIDENCE OF TESTING [10 MARKS]	<p>None (0-3)</p>	<p>Some evidence of testing e.g. able to broadly discuss identified issues found.</p> <p>Lacking exception handling. (4-5)</p>	<p>Good evidence of testing, known issues and solutions discussed.</p> <p>Some evidence of using the debug tools OR evidence of exception handling. (6)</p>	<p>Robust evidence of testing, citing approach adopted. Debugging used with perhaps some evidence of formal testing approaches.</p> <p>Exception handling evidenced and discussed. (7+)</p>	
PRESENTATION [10 MARKS]	<p>Poor adherence to vodcast guidelines.</p> <p>Lacking clarity & not able to discuss main issues (0-3)</p>	<p>Satisfactory adherence to guidelines but perhaps rushed or verbose presentation.</p> <p>Able to broadly discuss what was done.</p> <p>Lacks clear discussion of the program limitations (4-5)</p>	<p>Close adherence to the guidelines, well-paced and clearly presented.</p> <p>Able to discuss the main limitations. (6)</p>	<p>Able to articulate the main limitations and the challenges overcome. (7+)</p>	

Sample Program File Templates

You should plan to use these templates to create appropriate files for processing. This file setup provides:

- **2 cars already parked** (one standard, one EV).
- **4 free spaces** for students to test adding cars.
- **At least one disabled space** for validation testing.

You are free to **extend** these files to fully explore your own test cases.

SPACES.txt

```
# Space ID, Location, Type e.g., Standard/Disabled/EV.  
S001, Level 1 - Bay 01, Standard  
S002, Level 1 - Bay 02, Standard  
S003, Level 1 - Bay 03, Disabled  
S004, Level 1 - Bay 04, EV  
S005, Level 2 - Bay 01, Standard  
S006, Level 2 - Bay 02, Standard
```

CARS.txt

```
# Registration, Owner Name, Contact, Entitlement  
AB12CDE, Alice Johnson, alice.johnson@email.com, Standard  
XY34ZRT, Bob Smith, bob.smith@email.com, EV  
EV99CAR, Charlie Green, charlie.green@email.com, Disabled  
ZZ11AAA, Dana White, dana.white@email.com, Standard  
DD22BBB, Ethan Brown, ethan.brown@email.com, Standard
```

PARKED.txt

```
#Space ID, Registration, Time In, Expected Time Out  
S001, AB12CDE, 2025-09-30 09:15, 2025-09-30 17:00  
S004, EV99CAR, 2025-09-30 08:45, 2025-09-30 18:00
```