|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task:** | | **2** | | |
| **Task Title:** | | **Project** | | |
| **Task Code:** | | **AT2 PRJ Task 2** | | |
|  | |  | | |
| Assessment type (): | | | | |
|  | Questioning (Oral/Written) | |  | Portfolio |
|  | Practical Demonstration | |  | Project |
|  | 3rd Party Report | |  | Other – Please Specify |

|  |
| --- |
| The base requirements this assessment task include:   * IDE or editor for developing Python programs (only IDLE and PyCharm supported by the college) * Access to Office 365 & Microsoft Word   Use of some of these items may not occur in this part of the assessment task. |
| Assessment Due NOTICE: While this assessment is due on the second last week of the course. You **MUST** review elements of this submission with your lecturer before submission to pass this assessment.  Refer to Blackboard for most accurate dates, which may alter due to unforeseen circumstances. |
| Instructions The project consists of the following parts:   1. This journal and general project instructions 2. A template python program that you can use as a basis of your development work   Follow the steps outlined in this document and complete the required coding project.  You must follow the instructions in and complete the tasks for all components to complete this course. You may need to submit additional files |
| Important If you are using a different IDEs or a different structure for your application, then assistance with those tools and forms may be limited. Discuss with your lecturer before straying too far off the path. |
| Scenario You are currently working as a junior software developer at words-are-us, a Perth-based company that develops content and apps to build engagement.  You have been contracted to gain a deeper understanding of the popular tool Wordle and develop a prototype app that emulates the original wordle algorithm but provides these capabilities from the command line. As a junior developer, you have access to your peers, the senior developer (played by your lecturer), and occasional access to the customer (usually also played by your lecturer). |
| Specific Instructions This file is to be submitted as a journal that demonstrates the following aspects of your competency:   * Your ability to prioritise your own work * Obtain feedback to validate and enhance your design decisions * Confirm that the application meets specifications * Evaluate and reflect on decisions you made in the process of development   Save the file as:   * XXX\_ICTPRG302\_Proj\_Journal.docx   Replacing XXX with your initials.  For example, Adrian Gould would use AG\_ICTPRG302\_Proj\_Journal.docx for his submitted filename.  Upload any code as a archive (zip) containing your .py files. If applicable, remove the virtual environment (**venv** or **.venv**) from the zip-file before uploading it to Blackboard. |
| Answering Questions When a step includes a question, you must attempt to answer it. A word count is sometimes given, but most answers require at most 1-3 paragraphs.  All answers must be in complete sentences unless indicated. You must use your own words unless otherwise specified. |
| Sources of Information In industry, it is good practice to keep track of where information was obtained. This is especially true if it is a written document, or even code.  If you answer any questions using information from web sites, please include the site name and URL (Web site address) after the answer. Likewise, include the title and author for books and magazine articles. For example:   * RS Electronics Ltd: <https://au.rs-online.com/> * Slack API Documentation, Users List Method: <https://api.slack.com/methods/users.list>  Code Storage and Issue Tracking An industry-standard tool for managing program source is git. GitHub is a free service that provides hosting for git repositories. You may choose to use this system for source control, task management, or both. However, it is not required for this project.  You may also use OneDrive within your college Office365 to store a backup of your code or keep a copy on a USB thumb drive. |
| *This space left intentionally blank.* |

## ctions

Session 1

### Familiarisation

1. In your own words, briefly explain: what is Wordle?

Players have six attempts to guess a five-letter word, with each guess providing feedback through colored tiles. A green tile indicates the correct letter in the correct position, a yellow tile shows a correct letter in the wrong position, and a grey tile means the letter is not in the word at all.

1. Download the project files from Blackboard, list the files included in the project and briefly state their intended purpose. *The first file has been entered for you*:

**ICTPRG302-AT2-PRJ.docx**:

*That’s this project journal. I need to complete this journal every week so that I can pass this unit. This document will also help me tackle the project.*

**All words**

A list of all correct 5 letter words that are valid input

**Target words**

A list of 5 letter words that will be used as the target word

**Python template**

A framework with the necessary functions for the program to work

*Space for your answers*

### Task management

In your project there is an Excel spreadsheet that you can use to manage your tasks. You can use it to provide evidence of managing and prioritising your own tasks. The list of tasks in the sheet are examples only and you can change the list to suit your needs. You will be asked to submit a copy of this list later in this journal to demonstrate that you are following a plan (you are, right?).

1. What are two tasks you will work on next?

Score function  
valid input function

Sessions 2 – 3

### Requirements and design

1. After discussion with the senior developer and the client, you have divided your requirements into distinct elements. Some of those elements are listed below, propose at least one Python function (e.g. print(), input(), open(), len()), or selection (if/elif/else), or iteration (for/while) construct that could help meet the requirement :

* Select a random word (target) from the target words file

# listed for you

import random

# open() and read the file into a list 'words'

target\_word = random.choice(words)

* Present game instructions to player (That is, display an output to the screen)

Print(“{instructions}”)

* Prompt for guess and check that the guess is **valid\***

Input(“{prompt}”)

* Score the guess by providing clues on each character’s match to the target word’s characters

For(len(word))

* Repeat until the maximum number of attempts or the player correctly guesses the word

While attempt\_number < max\_attempts

### Algorithm

You ask for a flowchart to help you understand how the application will work. The senior developer draws a similar diagram to this on the whiteboard:



“What are the double lines?” You ask, pointing at the Score Guess box:

Shape

Description automatically generated with low confidence

“It indicates that there’s a bit more of a process in here,” they reply, “it’s where you step in actually”.

They explain that you need to develop the algorithm for how *an individual guess* will be scored.

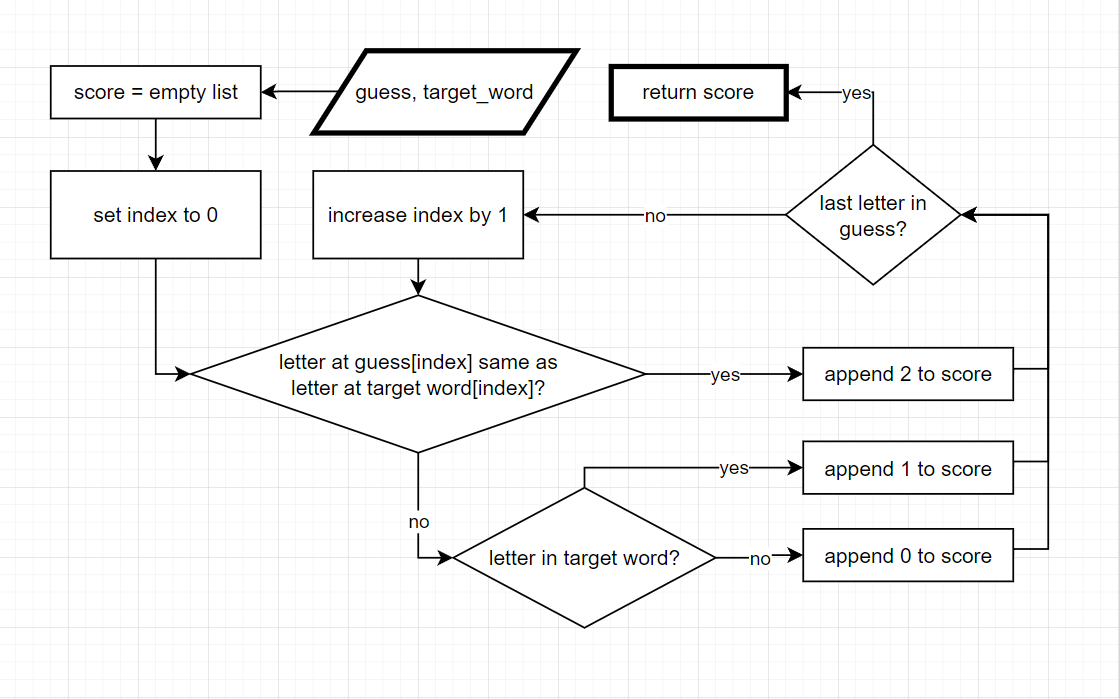
Complete two versions of the scoring algorithm Score Guess - (1) as a flowchart and (2) as pseudocode.

1. Draw a flowchart of the scoring algorithm. The input of the algorithm is a guess word and a target word, and the output is a score for each character in the guess word.

Include an image of your flowchart. You can use any tool you like to draw the flowchart (you can even take a picture of the whiteboard), but we recommend any of the following:

<https://app.diagrams.net/>

<https://mermaid.live/> (a bit harder)



1. Write pseudocode of the same scoring algorithm you developed previously

Get input: guess, target word

Set variable score = empty list

Set variable index = 0

If: letter at guess[index] same as letter at target word[index]  
then: append 2 to score

Else if: letter at guess[index] in target word  
then: append 1 to score

Else:

Then: Append 0 to score

If: last letter in guess

Then: return score

Else:

Then increase index by 1

Go back to line 4

Review your algorithm and pseudo-code with the senior developer (your lecturer)**.** The following questions should be answered during the review:

1. Given the following inputs, what output will your algorithm generate?

Guess: hello; Target: hello – Result: 22222

Guess: hello; Target: crane – Result: 01000

Guess: hello; Target: hzzzz – Result: 20000

Guess: hello; Target: zhzzz – Result: 10000

Guess: hello; Target: HELLO – Result: 22222

Guess: hello; Target: world – Result: 00121

1. List any differences between what you thought your algorithm should do and what the senior developer thought. Did the cases above make you rethink elements of your algorithm?

The last one is incorrect as the first l should be marked as 0 because the letter it is referencing in the target word is correct for the next letter

Session 4

### Testing and Edge cases

When you completed the algorithm, you came across a seemingly simple input:

Guess: hello; Target: world

However, you may be unsure what the output should be. There are two L’s in the guess, but only one L in the target. Should each character be scored individually? Discuss with your peers and lecturer how you think it should be scored and proceed appropriately.

Notice how there are always cases that fall “between the cracks”.

Testing is how we ensure that our application is suitable and of a reasonable quality. It is also a good way, as we just saw, to clarify requirements.

Here is an example test case:

Example Test: Score hello/world

**Arrange**:

Provided that the target word is world, start a new game.

**Act:**

When prompted for a guess, enter hello

**Assert:**

The response shall be:

**H E L L O**

**- ­- ? + ?**

Note to Cert IV Programming Students:   
You may want to take on the challenge of coding repeated characters consistently with the original Wordle Game, if you take it on, and we would really like you to, then amend the previous example appropriately!

1. Include two additional test cases using arrange/act/assert structure.   
   (Discuss with your lecturer.)

**Arrange**:

Provided that the target word is ababa, start a new game, wait until prompted for a guess

**Act**:

enter aaaaa

**Assert**:

The response shall be:

A A A A A

+ - + - + -

**Arrange**:

Provided that the target word is babab, start a new game, wait until prompted for a guess

**Act**:

enter acaca

**Assert**:

The response shall be:

A C A C A

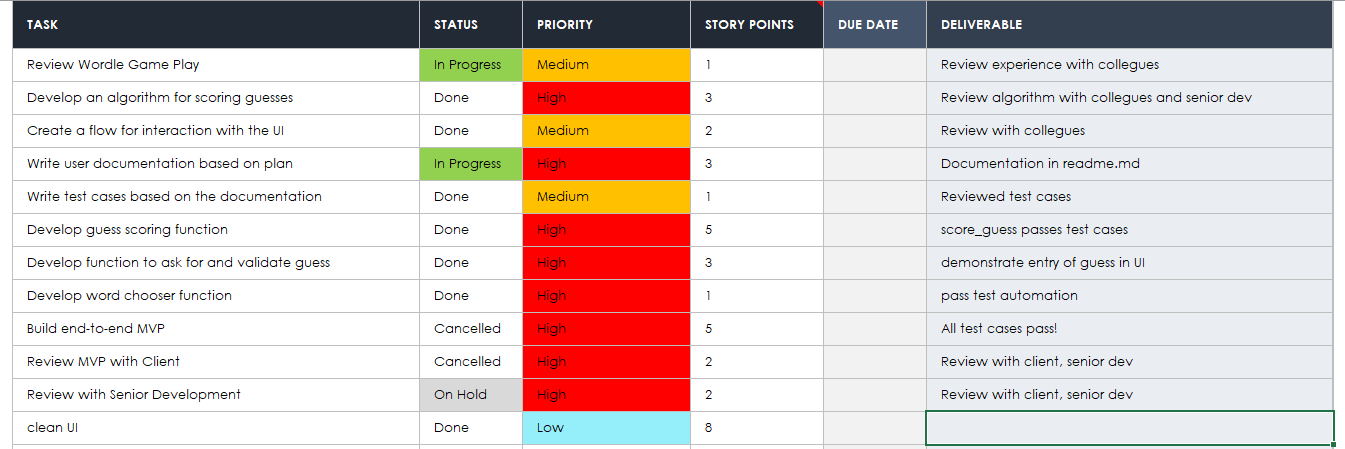
? - ? - -

Session 5

### Coding the Scorer (continued)

1. Include a screenshot of your task list demonstrating that you are following a plan to complete the code

Show: items that are “Done” or “In Progress”; show that you have added appropriate dates to at least some items.



*Space for your screenshot*

Session 6

### Debugging

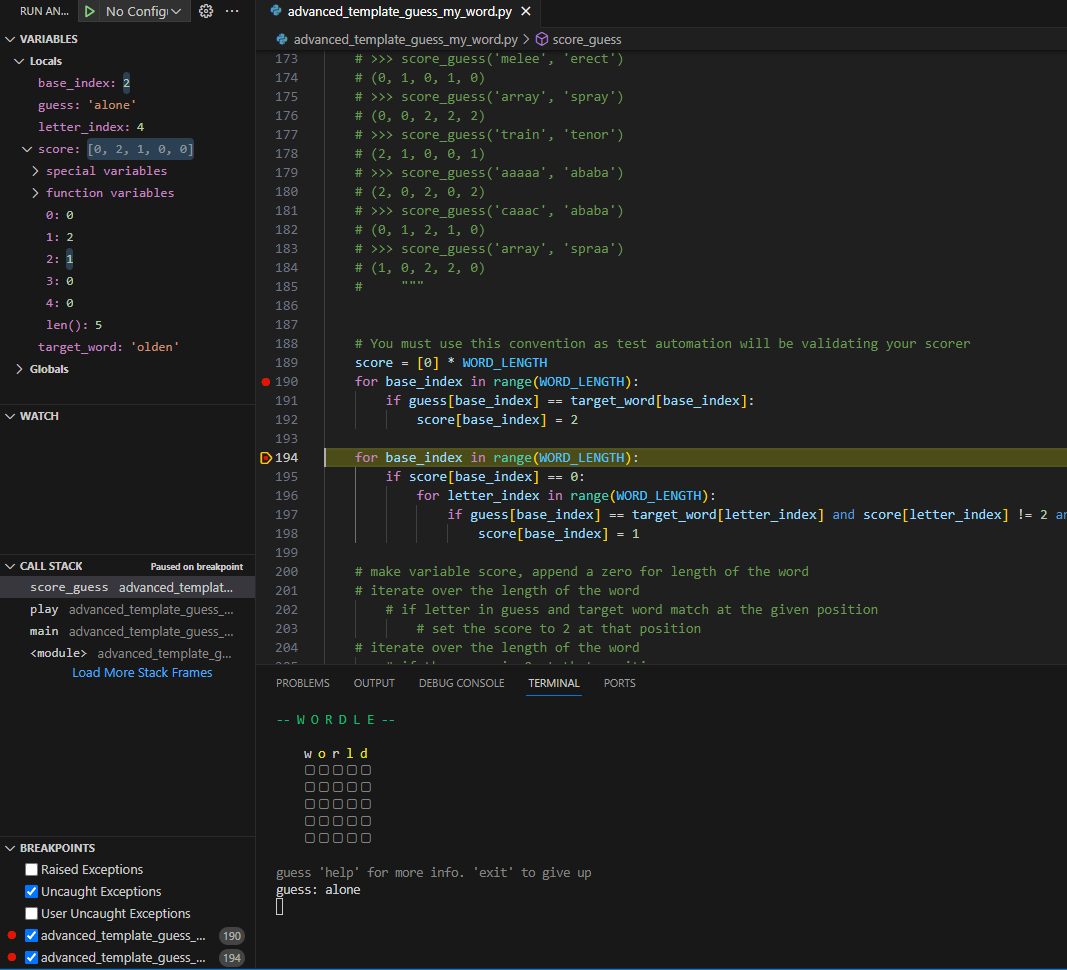
Debugging means exactly what it sounds like *de* ­– *bugging*. That is, removing bugs from your code. Debugging is itself an art form and can get advanced very quickly.

The good news is you have already been debugging since you started this course! Every time you get an error, pause, think about what caused the error, and make changes to your code, you are debugging! Every time you get unexpected results, and you add print statements to your code to figure out why - you are debugging. These debugging techniques are perfectly valid and can serve you well until intermediate levels of Python.

See the following for more:

<https://blackboard.northmetrotafe.wa.edu.au/webapps/blackboard/content/listContentEditable.jsp?content_id=_3109381_1&course_id=_32613_1>

1. Give a brief example of how you debugged code during the completion of this application.  
   Specifically, how were you able to determine the contents of a variable?



Session 7

### Catch up and start of reviews

Nothing to do here but code!

This is a good chance to remind you that your lecturers are here to help. If you feel like you are falling behind, this is a good time to contact your lecturer (if you haven’t already!). There are also code samples we provide that you can rely on. We are almost at the finish line. Remember that you do not have to showcase perfect code for your review, and no matter what you provide, we are going to find something for you to change!

*Space for your well being*

Session 8

### Code review

Review your code with the senior developer (lecturer) and write down any changes you need to make because of the review. You can also add them to your task list and paste a screenshot here:

1. List any coding style issues encountered in the review here

Constants in upper case

1. List any other changes that you needed to make to your code

Minor readability improvements

1. Were there any inconsistencies with the algorithm you created earlier in the term?

### Client review

Review your code with client (lecturer) and write down any changes you need to make because of the review. You can also add them to your task list and paste a screenshot here:

1. Did the application crash during the review? If so, why?
2. Were any usability issues encountered that needed fixing?

During the review the client recognised the need for a new feature (and agreed to pay more for them, of course):

*“I’d like to save the number of tries and the secret word to a file. When the user finishes a game, I want them to see the average number of tries it takes them to make a guess!”*

1. What other new feature did you agree to add to the application?
2. Has your lecturer reviewed the information you entered here?

Yes – Signed: <your name>

*Additional Space if Required*

Session 9

### Final steps

Ensure you have made the required changes in the code that you agreed to in both your developer and client reviews. Finally, submit your code along with this document and all the best!

Please note: we prefer to receive a zip of your code.

### You DID IT!!!

Well done on completing your first software development project!

You’ve come a long way.