|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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.* |

Session 1

### Familiarisation

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

Wordle is a game app that has a daily 5 letter word that users need to try and guess. When a word is guess, the game will let the user know if each letter in the guessed word is either: In the daily word in the correct place, in the daily word but not in the correct place, or the letter is not in the word. They have 6 guesses to try and guess the daily word.

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.*

**Advanced\_template\_guess\_my\_word.py**

Enter your description

**Project-simple-todo-list.xlsx**

Enter your description

**Word-bank**

This folder has 2 txt files, the first as all the target words that the main code will use to select one to be the target word of that game. The other is a list of all words, this will be used to check to make sure that the word the user enters is an actual word

*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?

1. Set up a GitHub repo

2. plan algorithms for the guess/score functions

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(welcome, aim, instructions)

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

guess\_word = input("Enter a guess: ").lower()

if guess\_word in all\_words\_list and len(guess\_word) == WORD\_LENGTH:

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

for i, char in enumerate(char\_guess):

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

while guesses\_left > 0:

### 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.

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)

A diagram of a game

Description automatically generated

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

Char\_target = list of characters in target word.lower

Char\_guess = list of inputed word.lower

For each character in char\_guess

Score = emply list

If char\_guess index is equal to char\_target at the same index

Score = 2

Append the score to score list

Elif char\_guess in the char\_target

Score = 1

Append the score to score list

Else:

Score is 0

Append the score to score list

Print the guess\_score list

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?

Emailed senior developer full flow chart and above psudo code, the feedback received feedback:

My feedback:

Based on the flowchart provided,

1. add a file name (location\filename\) plus version control in the footer of the diagram.

2. use a larger arrow to make it easy to see the flow

3. for the box - restrict to only one task (it should correspond to only one instruction in python in this case)

Overall a very good flowchart and pseudo codes. It is clear, use of right colour and different shapes to convey the algorithm.

However, after looking at duplicate letter test cases. I realised that my code would produce:

Target: world

Guess: H E L L O

0, 0, 1, 2, 1

Instead of:

0, 0, 0, 2, 1

I have added a used\_char set list and my code now looks like this:

def score\_guess(char\_guess):

score = []

used\_char = set()

for i, char in enumerate(char\_guess):

if char == char\_target[i]:

score.append(2)

used\_char.add(i)

elif char in char\_target and char\_target.index(char) not in used\_char:

score.append(1)

used\_char.add(char\_target.index(char))

else:

score.append(0)

print(score)

if all(val == 2 for val in score):

print("Congratulations!")

return False

return True

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.)

Test Case 1: Single Correct Letter but duplicate letter in guess

If the guess word has duplicate letters, one being in the correct place of the target word. (the target word in this case only contains that letter once). The second of the duplicate letter should be marked as a miss

Arrange:

Target Word: apple

Act:

Start a new game and enter the guess eagle.

Assert:

Expected Output

⬜ 🟩 ⬜ ⬜ 🟨

Actual Output:

Test Case 2: Invalid guess

If the user guesses a word that is not in the all\_words list, the game will generate an invalid message.

Arrange:

Target word: Random choice (no change)

Act:

Guess ‘aaaaa’

Assert:

Expect output:

Sorry, please enter a valid guess

Actual Output:

Sorry, please enter a valid guess

Test Case 3: Game to finish after the 6th guess

Arrange:

Target word: HELLO

Act:

Start game and guess “World” 6 times

Assert:

Expected output:

Sorry, you lost 😭

The target word was: hello

Actual output:

Sorry, you lost 😭

The target word was: hello

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.

A screenshot of a computer

Description automatically generated

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?

I would be constantly running small chunks of the code, and used multiple print functions to check if the variables and lists were as intended

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

No coding issues at the dev code review

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

A screenshot of a computer program

Description automatically generated

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

Original algorithm did not account for the Single Correct Letter but duplicate letter in guess test, this was adjusted during the coding phase after

### 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:

A screenshot of a computer

Description automatically generated

1. Did the application crash during the review? If so, why?

no

1. Were any usability issues encountered that needed fixing?

no

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?

To add how many guess attempts remained and an option to play again after the game ends

1. Has your lecturer reviewed the information you entered here?

Yes – Signed: Kim Searle

*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.