

How Does That Work?




Jupyter Notebooks

Notebooks? Python?

- Jupyter Notebooks (**.ipynb**):
 - Consists of multiple cells
 - A cell can contain regular Python code *or* plain text (markdown)
 - Very good for visually explaining what is happening in your code, plots
 - You can also run samples of code at a time
 - Lecture code are in this format in this course
- Python File (**.py**):
 - Executable file. You can even run these on your machine without installing any IDE's!
 - Run in a linear format, from start to file.
 - Good for computation, or batching tasks, like a data engineering pipeline!
 - No intermittent output saved, no formatted text/figures like in a notebook.

A few different ways we can work with Notebooks

- Locally

- Jupyter IDE *through Anaconda*  Jupyter
- PyCharm Professional IDE* (Community version supports view code  **PyCharm**)
- Visual Studio Code 
Visual Studio Code
- And more :)

- Online

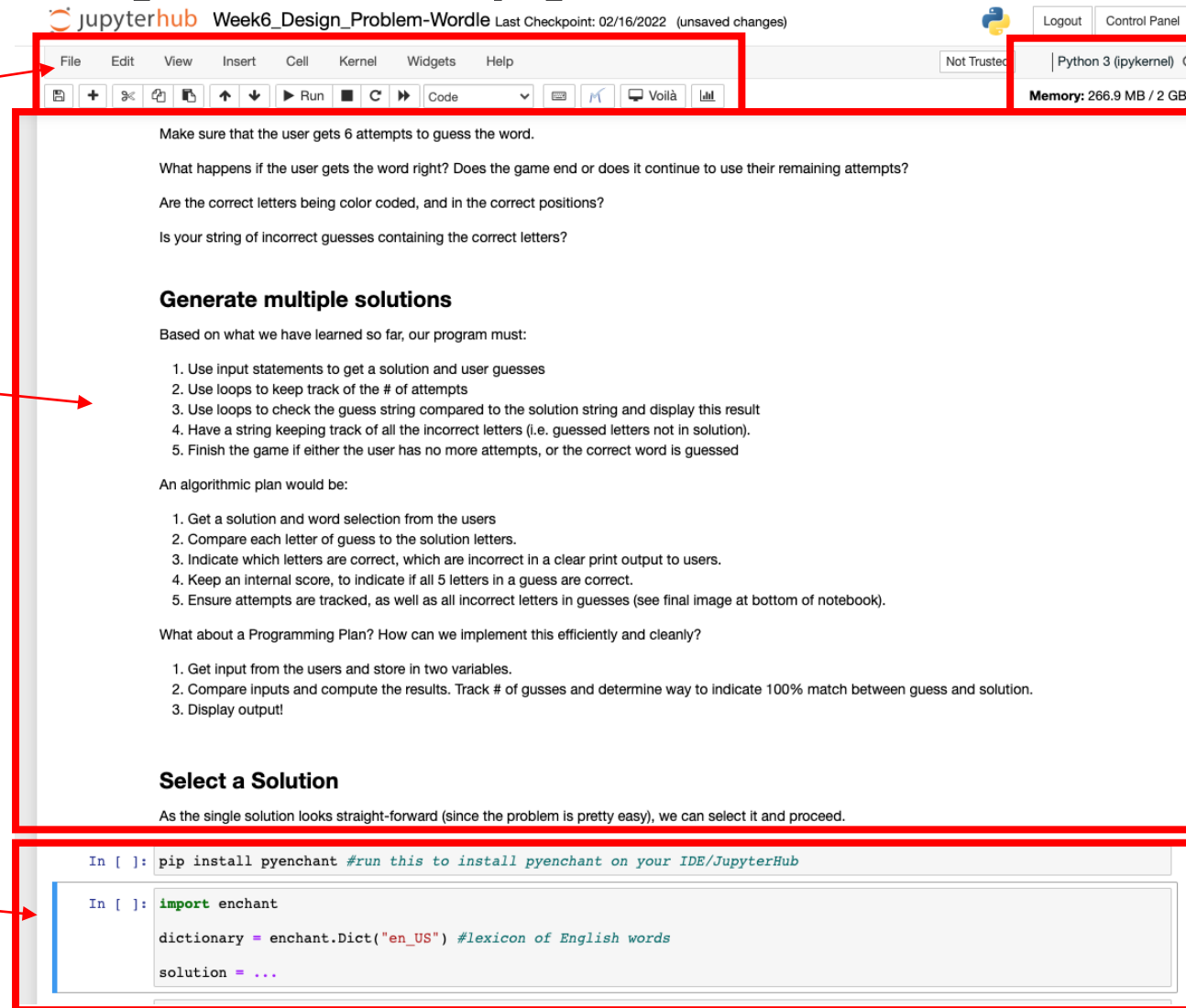
- UofT JupyterHub (such as the lecture note 

The Anatomy of a Jupyter Notebook (.ipynb)

Cell controls (run, restart, format, copy/paste...)

Markdown
(.md) Text
Cells

Python
Code!



The screenshot shows a Jupyter Notebook interface. At the top, there's a header bar with the Jupyter logo, the title 'Week6_Design_Problem-Wordle', and a 'Last Checkpoint' timestamp. Below this is a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and other actions. The main area contains a mix of text and code. The text is in a monospaced font, and the code is in a different monospaced font. The interface is clean and modern, with a light gray background.

File Edit View Insert Cell Kernel Widgets Help

Python 3 (ipykernel) O

Memory: 266.9 MB / 2 GB

Make sure that the user gets 6 attempts to guess the word.

What happens if the user gets the word right? Does the game end or does it continue to use their remaining attempts?

Are the correct letters being color coded, and in the correct positions?

Is your string of incorrect guesses containing the correct letters?

Generate multiple solutions

Based on what we have learned so far, our program must:

1. Use input statements to get a solution and user guesses
2. Use loops to keep track of the # of attempts
3. Use loops to check the guess string compared to the solution string and display this result
4. Have a string keeping track of all the incorrect letters (i.e. guessed letters not in solution).
5. Finish the game if either the user has no more attempts, or the correct word is guessed

An algorithmic plan would be:

1. Get a solution and word selection from the users
2. Compare each letter of guess to the solution letters.
3. Indicate which letters are correct, which are incorrect in a clear print output to users.
4. Keep an internal score, to indicate if all 5 letters in a guess are correct.
5. Ensure attempts are tracked, as well as all incorrect letters in guesses (see final image at bottom of notebook).

What about a Programming Plan? How can we implement this efficiently and cleanly?

1. Get input from the users and store in two variables.
2. Compare inputs and compute the results. Track # of gusses and determine way to indicate 100% match between guess and solution.
3. Display output!

Select a Solution

As the single solution looks straight-forward (since the problem is pretty easy), we can select it and proceed.

```
In [ ]: pip install pyenchant #run this to install pyenchant on your IDE/JupyterHub
```

```
In [ ]: import enchant
```

```
dictionary = enchant.Dict("en_US") #lexicon of English words
```

```
solution = ...
```

Kernel information
(Python version +
memory)

APS106

Demo.