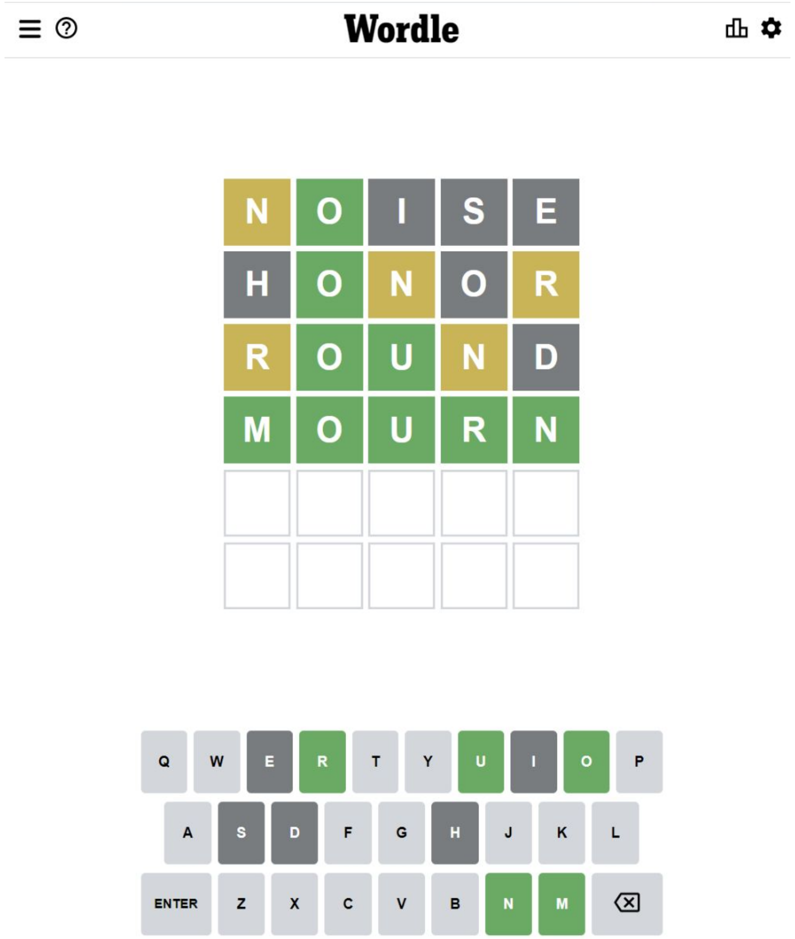
PyWordle

Alison Gale

Background

- Popular word guessing game created by Josh Wardle
- Led to spinoffs using different domains or dictionaries
- Folks have also made solvers to try to maximize scores



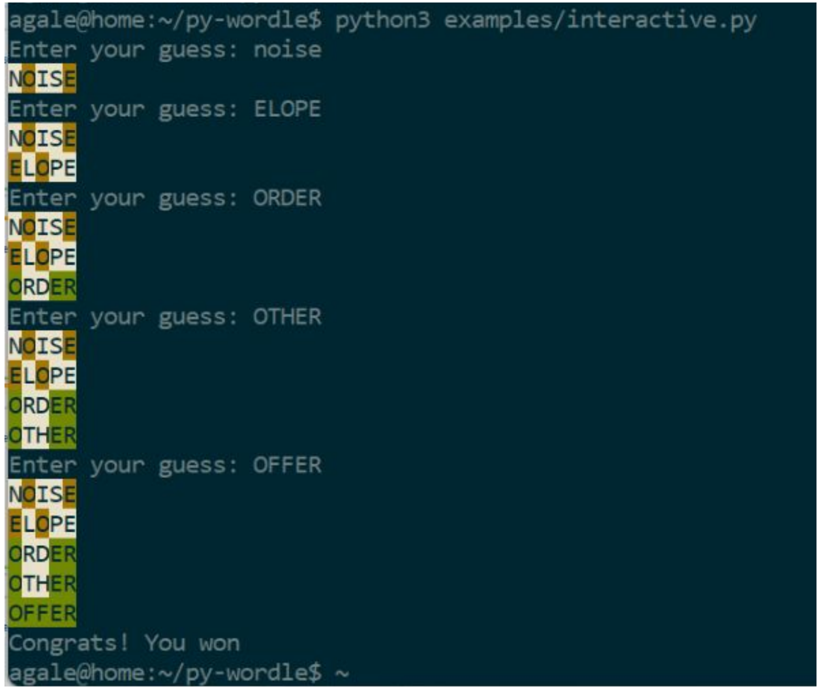
Data

- List of valid five-letter words
- List of “common” five-letter words as possible solutions

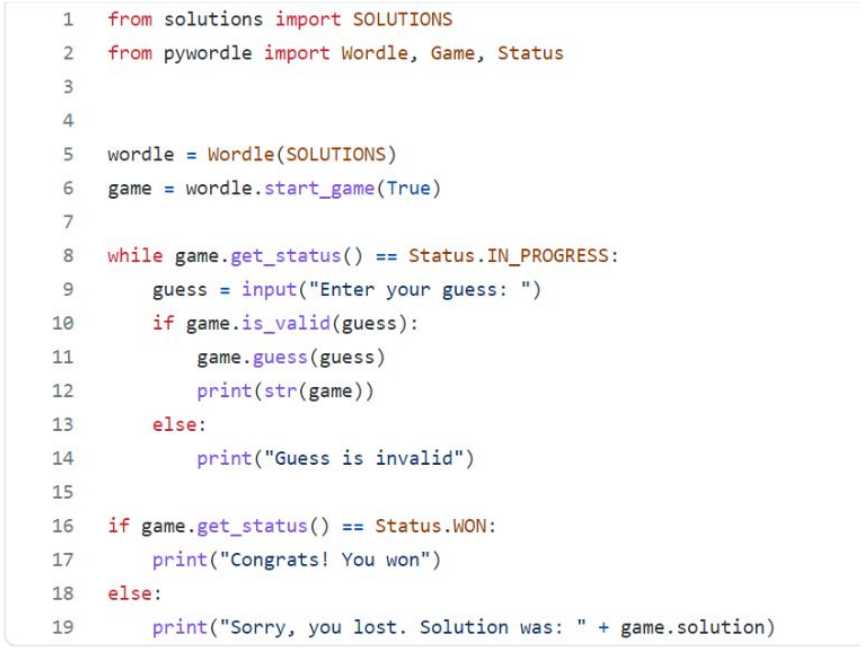
Use Cases

- Someone wants to play unlimited games
- Someone wants to build a custom game with special solutions
- Someone wants to build a solver

Demo



```
agale@home:~/py-wordle$ python3 examples/interactive.py
Enter your guess: noise
NOISE
Enter your guess: ELOPE
NOISE
ELOPE
Enter your guess: ORDER
NOISE
ELOPE
ORDER
Enter your guess: OTHER
NOISE
ELOPE
ORDER
OTHER
Enter your guess: OFFER
NOISE
ELOPE
ORDER
OTHER
OFFER
Congrats! You won
agale@home:~/py-wordle$ ~
```



```python
from solutions import SOLUTIONS
from pywordle import Wordle, Game, Status


wordle = Wordle(SOLUTIONS)
game = wordle.start_game(True)

while game.get_status() == Status.IN_PROGRESS:
    guess = input("Enter your guess: ")
    if game.is_valid(guess):
        game.guess(guess)
        print(str(game))
    else:
        print("Guess is invalid")

if game.get_status() == Status.WON:
    print("Congrats! You won")
else:
    print("Sorry, you lost. Solution was: " + game.solution)
```

Design

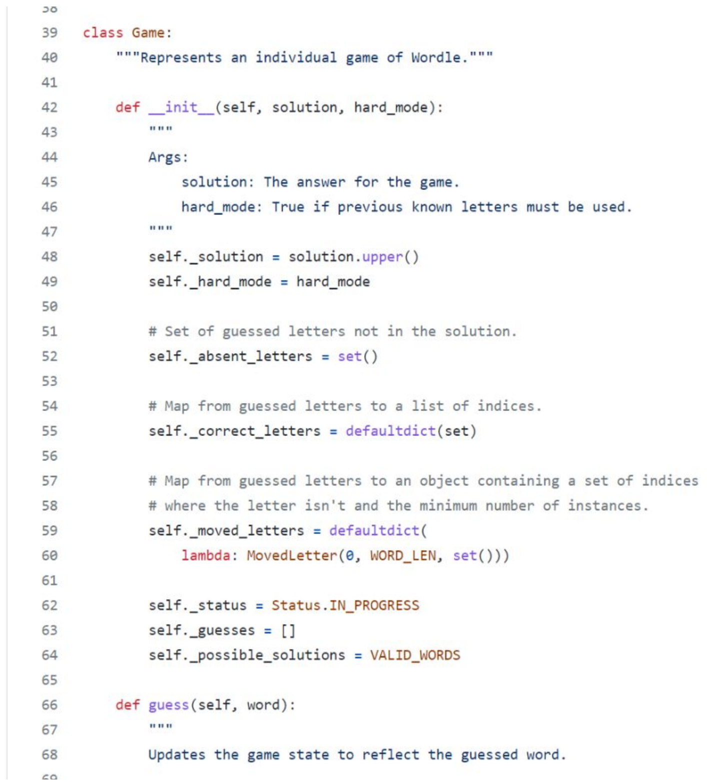
Wordle

```
__init__(solutions)
start_game(solution, hard_mode)
__repr__()
```

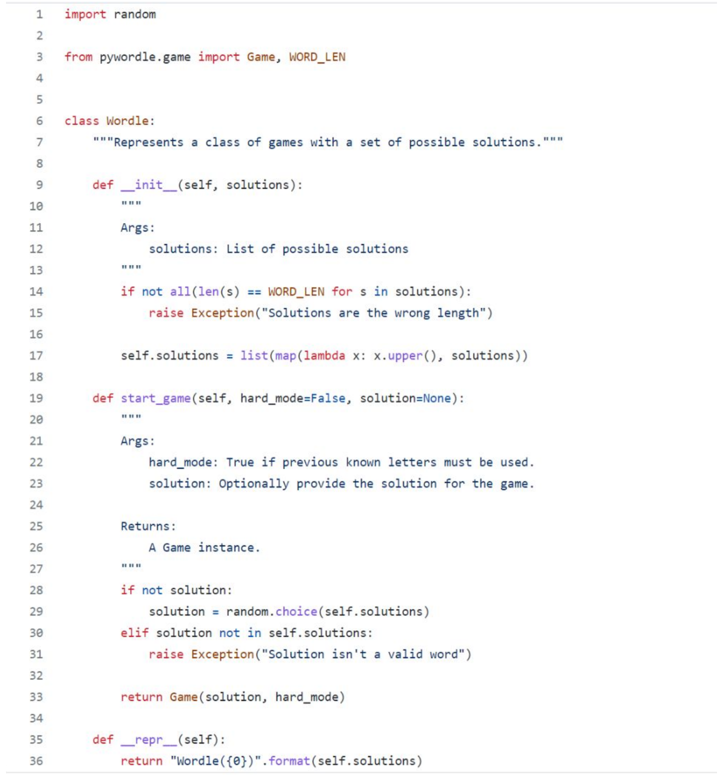
Game

```
__init__(solution, hard_mode)
guess(word)
is_valid(word)
get_status()
__str__()
__repr__()
```

Design

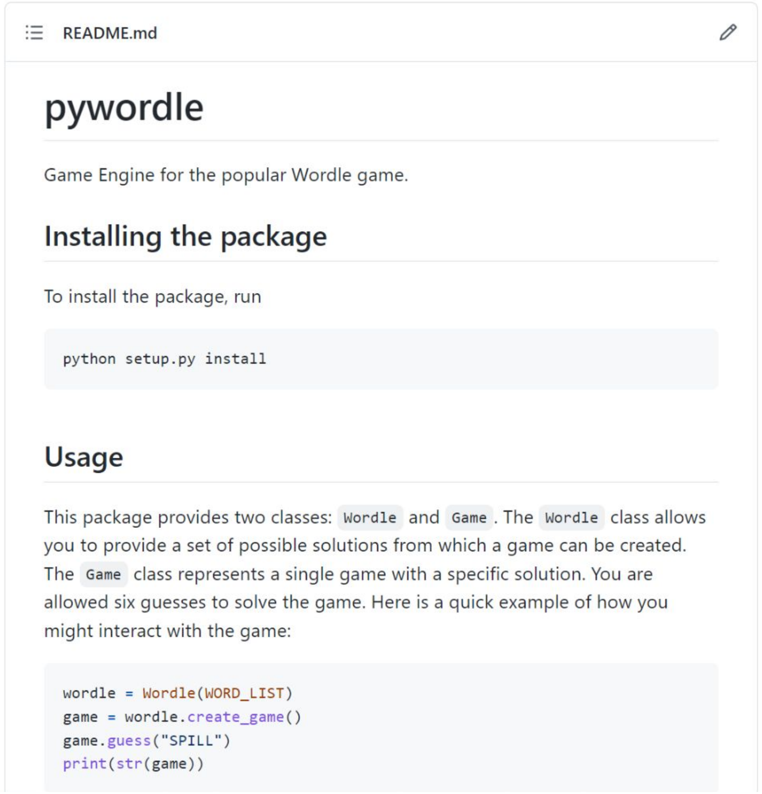


```python
class Game:
    """Represents an individual game of Wordle."""

    def __init__(self, solution, hard_mode):
        """
        Args:
            solution: The answer for the game.
            hard_mode: True if previous known letters must be used.
        """
        self._solution = solution.upper()
        self._hard_mode = hard_mode

        # Set of guessed letters not in the solution.
        self._absent_letters = set()

        # Map from guessed letters to a list of indices.
        self._correct_letters = defaultdict(set)

        # Map from guessed letters to an object containing a set of indices
        # where the letter isn't and the minimum number of instances.
        self._moved_letters = defaultdict(
            lambda: MovedLetter(0, WORD_LEN, set()))

        self._status = Status.IN_PROGRESS
        self._guesses = []
        self._possible_solutions = VALID_WORDS

    def guess(self, word):
        """
        Updates the game state to reflect the guessed word.
```



```python
import random

from pywordle.game import Game, WORD_LEN


class Wordle:
    """Represents a class of games with a set of possible solutions."""

    def __init__(self, solutions):
        """
        Args:
            solutions: List of possible solutions
        """
        if not all(len(s) == WORD_LEN for s in solutions):
            raise Exception("Solutions are the wrong length")

        self.solutions = list(map(lambda x: x.upper(), solutions))

    def start_game(self, hard_mode=False, solution=None):
        """
        Args:
            hard_mode: True if previous known letters must be used.
            solution: Optionally provide the solution for the game.

        Returns:
            A Game instance.
        """
        if not solution:
            solution = random.choice(self.solutions)
        elif solution not in self.solutions:
            raise Exception("Solution isn't a valid word")

        return Game(solution, hard_mode)

    def __repr__(self):
        return "Wordle({0})".format(self.solutions)
```

Project Structure



README.md

pywordle

Game Engine for the popular Wordle game.

Installing the package

To install the package, run

```
python setup.py install
```

Usage

This package provides two classes: `Wordle` and `Game`. The `Wordle` class allows you to provide a set of possible solutions from which a game can be created. The `Game` class represents a single game with a specific solution. You are allowed six guesses to solve the game. Here is a quick example of how you might interact with the game:

```
wordle = Wordle(WORD_LIST)
game = wordle.create_game()
game.guess("SPILL")
print(str(game))
```

```
py-wordle
├── LICENSE
├── README.md
├── TODO.md
├── docs
│   ├── design-spec.md
│   └── functional-spec.md
├── examples
│   ├── interactive.py
│   ├── solutions.py
│   └── simple.py
├── pywordle
│   ├── __init__.py
│   ├── game.py
│   ├── test_game.py
│   ├── test_wordle.py
│   ├── wordle.py
│   └── words.py
└── setup.py
```

Lessons Learned

- Building a general purpose module adds a lot of complexity
  - Game engine and solver had very disjoint constraints so I chose to focus on making the game engine
- Continuous integration is weird to set up when using test driven development
  - Recommend using it after an initial implementation is complete or write a small set of tests and the implementation in the same commit

Future Work

- Build in support for different length words
- Incorporate the interactive example with the library
- Add a new module to support solver use cases
- Visualize how the set of possible solutions was narrowed down