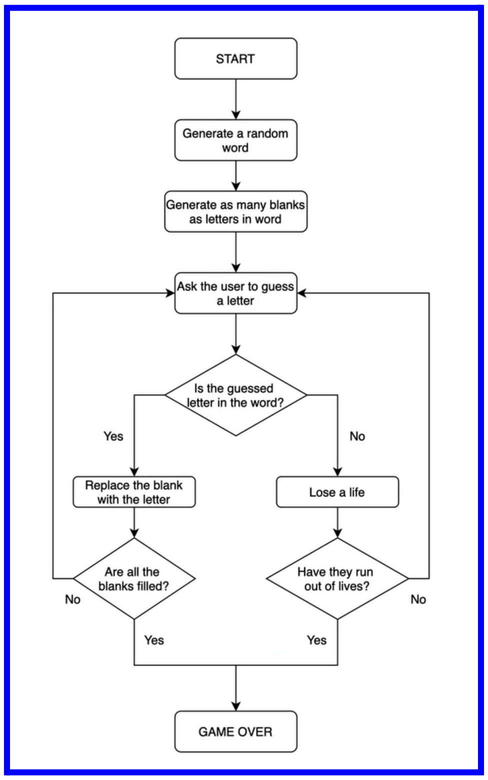
DAY 8: Hangman

What we're going to be building throughout this day is basically Hangman or the code equivalent of this game. So the way the game works is you have to guess a word, and for every wrong letter you submit, you end up taking a life away from this little man. And of course, the longer it takes you to get to the word, the more you put your little man in danger. So this is what the final completed project is going to look like.
https://appbrewery.github.io/python-day7-demo/

How to break a Complex Problem down into a Flow Chart

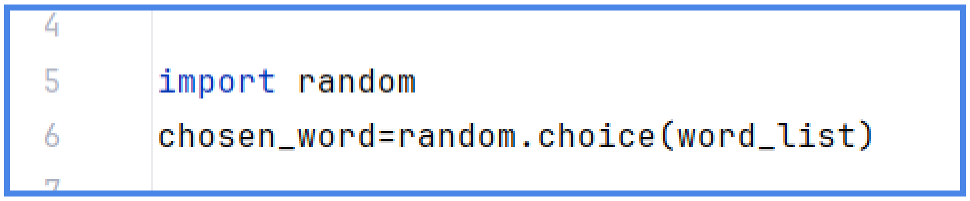
It just means that we can draw out a flowchart to represent the logic of our game. Like this:



START
Generate a random word
Generate as many blanks as letters in word
Ask the user to guess a letter
Is the guessed letter in the word?
Yes
No
Replace the blank with the letter
Lose a life
Are all the blanks filled?
Have they run out of lives?
No
No
Yes
Yes
GAME OVER

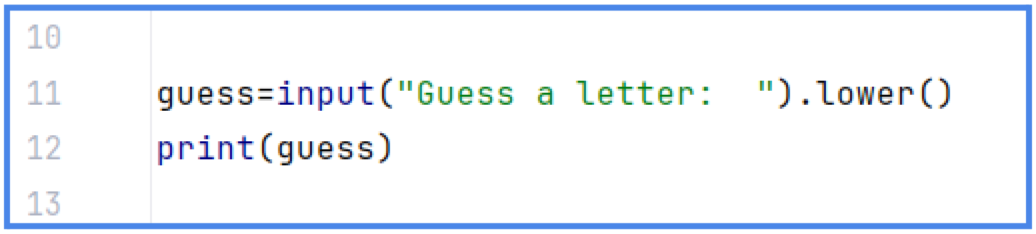
Step 1 - Picking a Random Words and Checking Answers

```
 # TODO-1 - Randomly choose a word from the word_list and assign it to a
variable called chosen_word. Then print it.
```



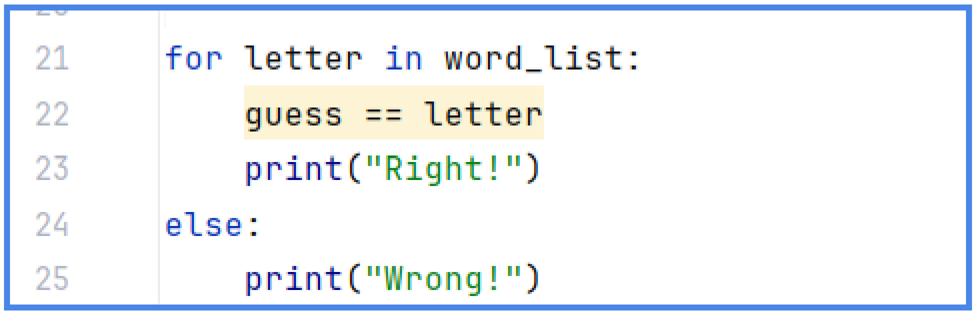
```
import random
chosen_word=random.choice(word_list)
```

```
# TODO-2 - Ask the user to guess a letter and assign their answer to a
variable called guess. Make guess lowercase.
```



```
guess=input("Guess a letter:  ").lower()
print(guess)
```

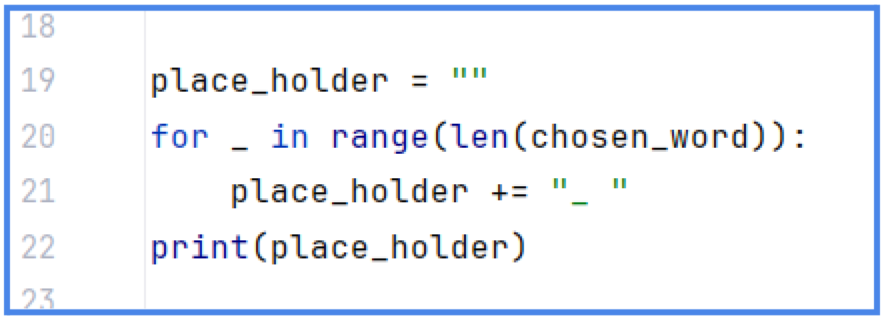
```
# TODO-3 - Check if the letter the user guessed (guess) is one of the letters
in the chosen_word. Print "Right" if it
#  is, "Wrong" if it's not.
```



```
for letter in word_list:
    guess == letter
    print("Right!")
else:
    print("Wrong!")
```

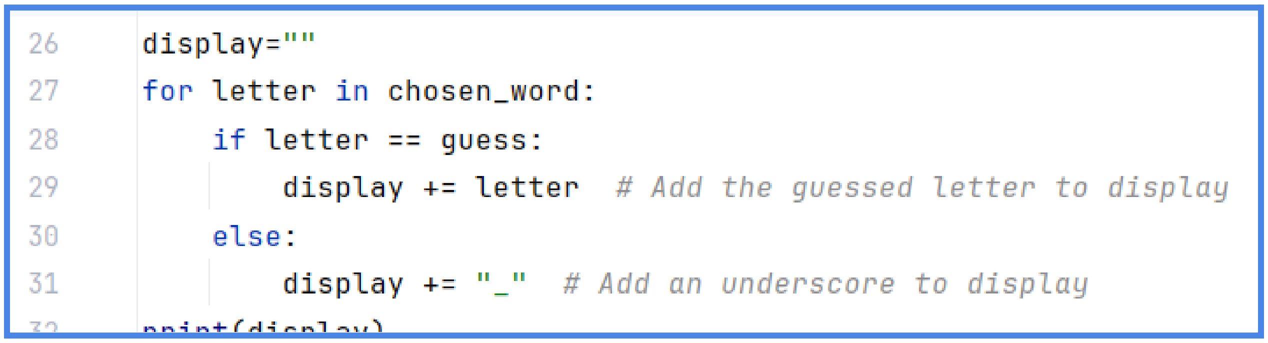
Step 2 - Replacing Blanks with Guesses

- TODO 1:Create an empty String called `placeholder`.
- For each letter in the chosen_word, add a `_` to `placeholder`.
- So if the `chosen_word` was "apple", `placeholder` should be `_ _ _ _ _` with 5 `"_"` representing each letter to guess.
- Print out `hint`.



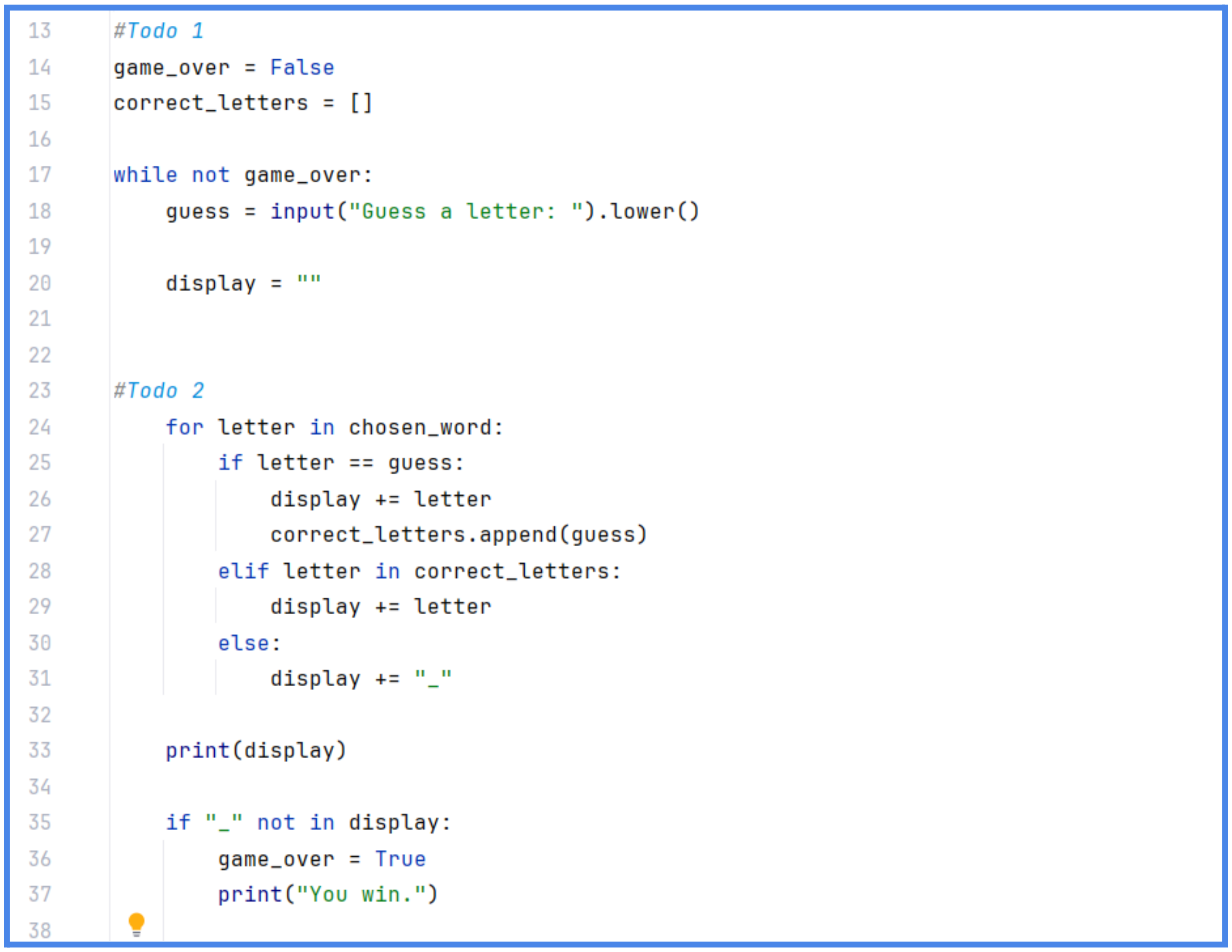
```python
place_holder = ""
for _ in range(len(chosen_word)):
    place_holder += "_ "
print(place_holder)
```

- TODO 2: Create an empty string called "display".
- Loop through each letter in the `chosen_word`
- If the letter at that position matches `guess` then reveal that letter in the `display` at that position.
- e.g. If the user guessed "p" and the chosen word was "apple", then `display` should be `_ p p _ _`.
- Print `display` and you should see the guessed letter in the correct position.
- But every letter that is not a match is represented with a "_".



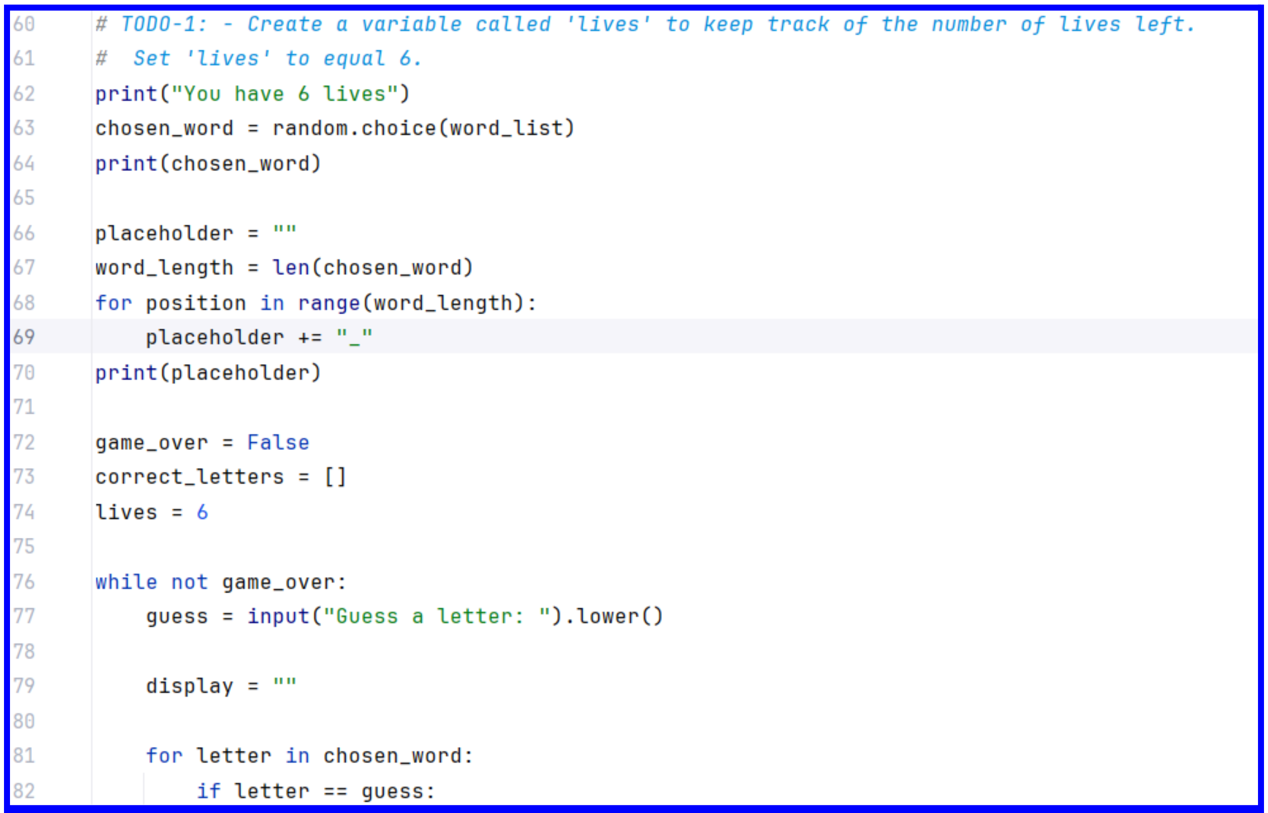
```python
display=""
for letter in chosen_word:
    if letter == guess:
        display += letter  # Add the guessed letter to display
    else:
        display += "_"  # Add an underscore to display
print(display)
```

Step 3 - Checking if the Player has Won

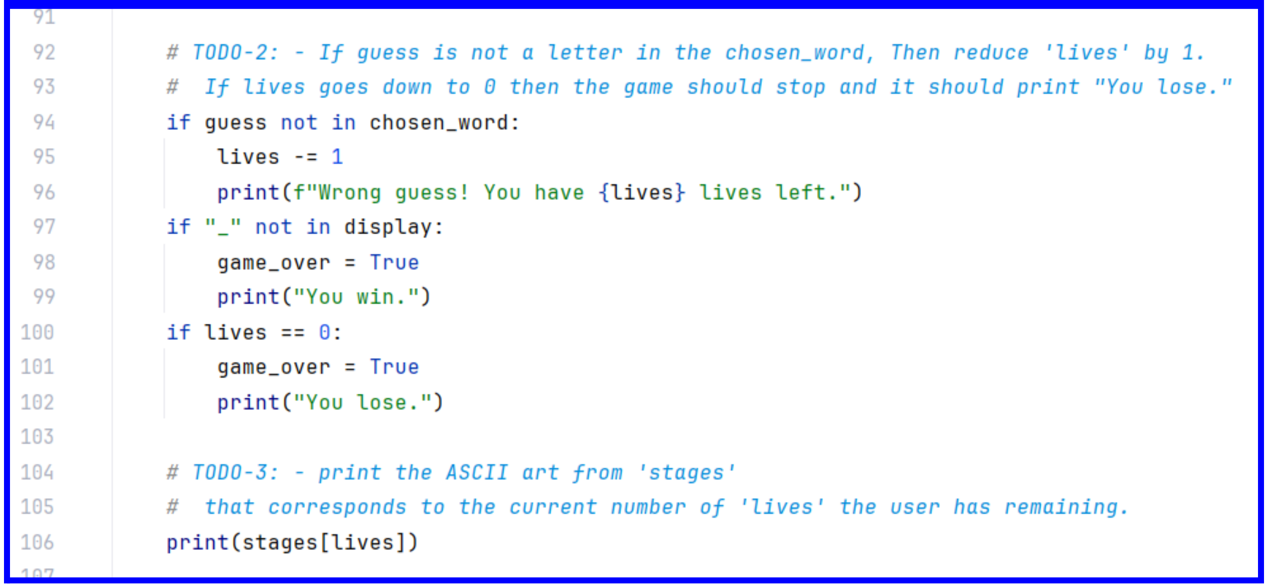


```python
#Todo 1
game_over = False
correct_letters = []

while not game_over:
    guess = input("Guess a letter: ").lower()

    display = ""


#Todo 2
    for letter in chosen_word:
        if letter == guess:
            display += letter
            correct_letters.append(guess)
        elif letter in correct_letters:
            display += letter
        else:
            display += "_"

    print(display)

    if "_" not in display:
        game_over = True
        print("You win.")

```

Step 4 - Keeping Track of the Player's Lives

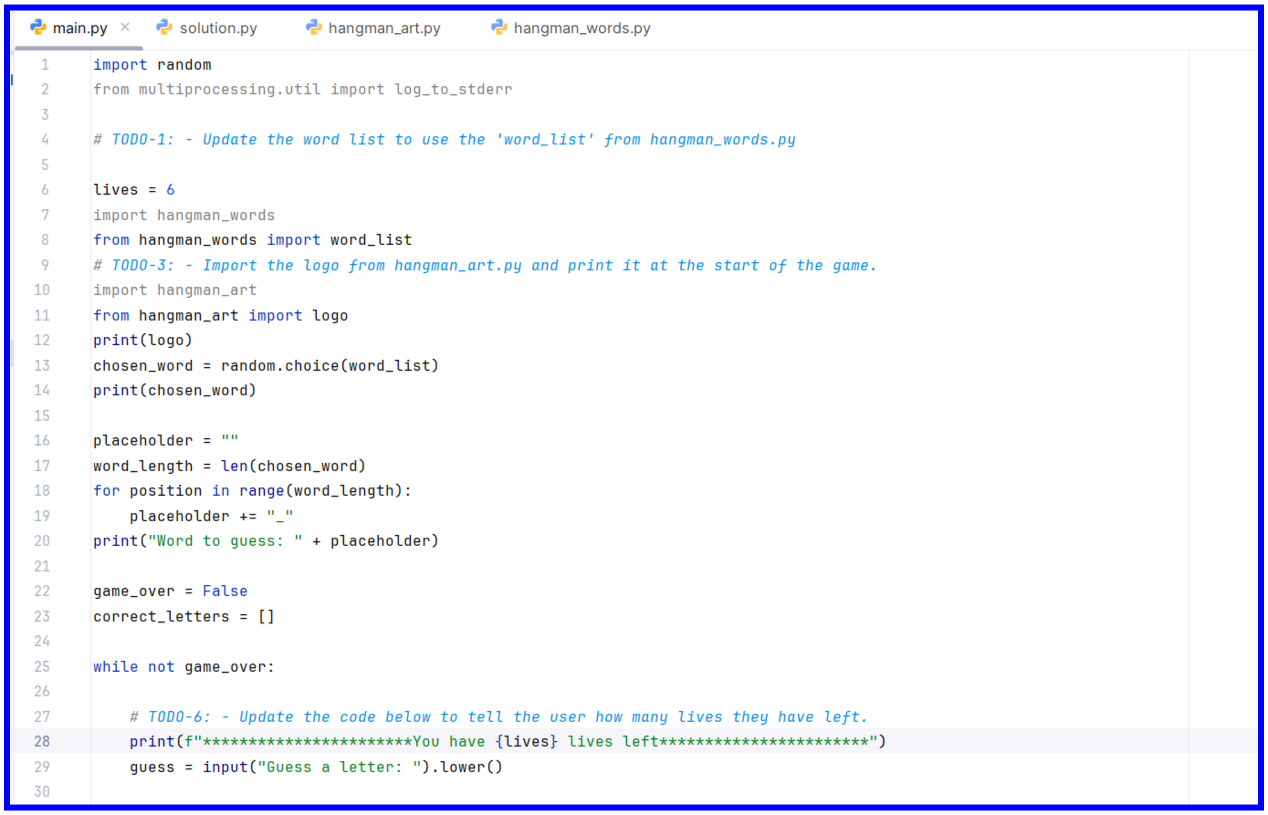


```python
# TODO-1: - Create a variable called 'lives' to keep track of the number of lives left.
#  Set 'lives' to equal 6.
print("You have 6 lives")
chosen_word = random.choice(word_list)
print(chosen_word)

placeholder = ""
word_length = len(chosen_word)
for position in range(word_length):
    placeholder += "_"
print(placeholder)

game_over = False
correct_letters = []
lives = 6

while not game_over:
    guess = input("Guess a letter: ").lower()

    display = ""

    for letter in chosen_word:
        if letter == guess:
```

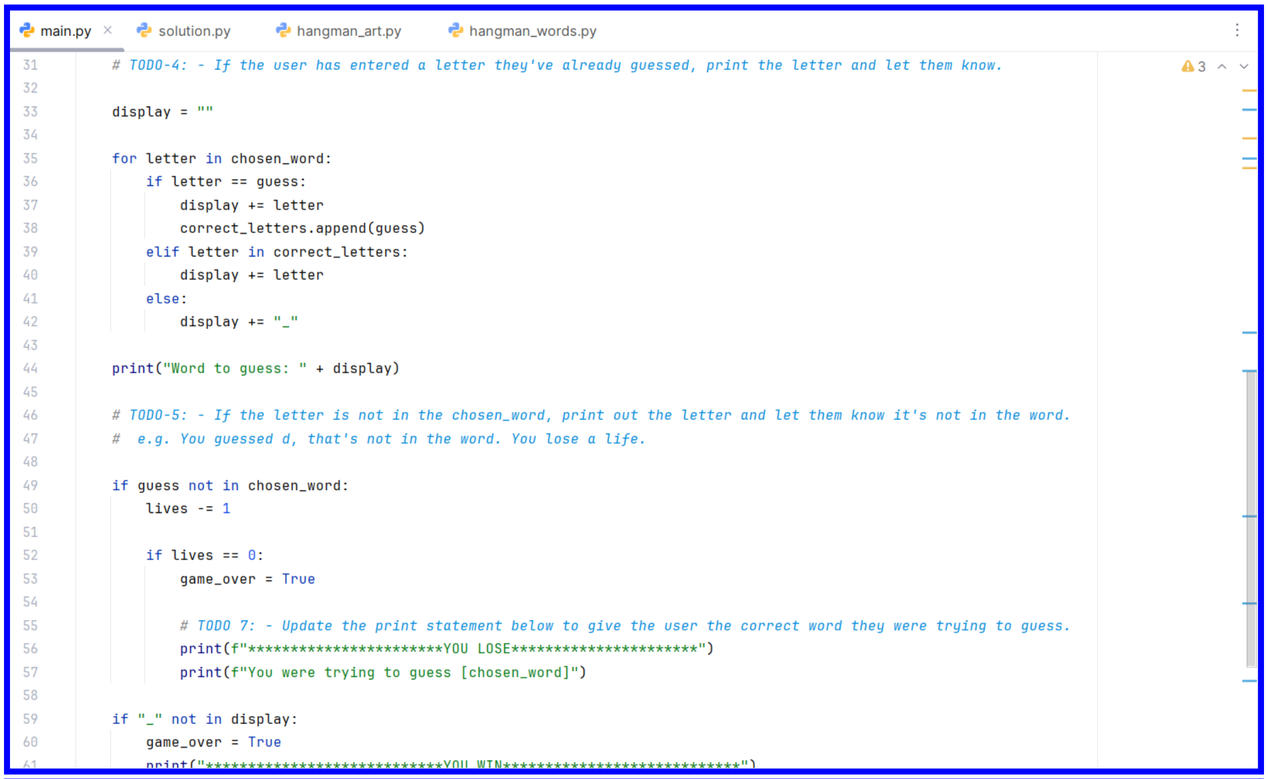


```python
    # TODO-2: - If guess is not a letter in the chosen_word, Then reduce 'lives' by 1.
    #  If lives goes down to 0 then the game should stop and it should print "You lose."
    if guess not in chosen_word:
        lives -= 1
        print(f"Wrong guess! You have {lives} lives left.")
    if "_" not in display:
        game_over = True
        print("You win.")
    if lives == 0:
        game_over = True
        print("You lose.")

    # TODO-3: - print the ASCII art from 'stages'
    #  that corresponds to the current number of 'lives' the user has remaining.
    print(stages[lives])
```

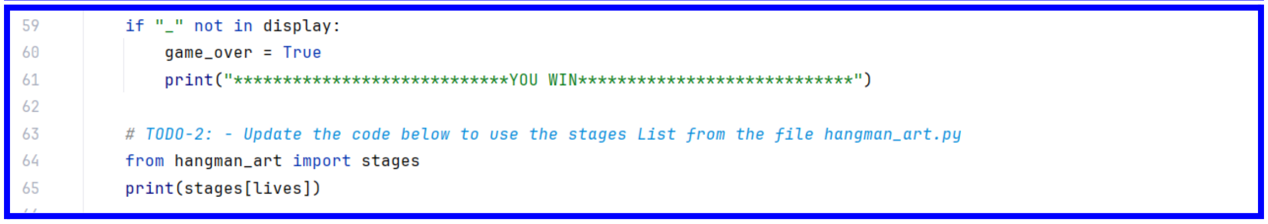
Step 5: Improving the User Experience



```python
import random
from multiprocessing.util import log_to_stderr

# TODO-1: - Update the word list to use the 'word_list' from hangman_words.py

lives = 6
import hangman_words
from hangman_words import word_list
# TODO-3: - Import the logo from hangman_art.py and print it at the start of the game.
import hangman_art
from hangman_art import logo
print(logo)
chosen_word = random.choice(word_list)
print(chosen_word)

placeholder = ""
word_length = len(chosen_word)
for position in range(word_length):
    placeholder += "_"
print("Word to guess: " + placeholder)

game_over = False
correct_letters = []

while not game_over:

    # TODO-6: - Update the code below to tell the user how many lives they have left.
    print(f"***********************You have {lives} lives left***********************")
    guess = input("Guess a letter: ").lower()

```



```python
    # TODO-4: - If the user has entered a letter they've already guessed, print the letter and let them know.

    display = ""

    for letter in chosen_word:
        if letter == guess:
            display += letter
            correct_letters.append(guess)
        elif letter in correct_letters:
            display += letter
        else:
            display += "_"

    print("Word to guess: " + display)

    # TODO-5: - If the letter is not in the chosen_word, print out the letter and let them know it's not in the word.
    #  e.g. You guessed d, that's not in the word. You lose a life.

    if guess not in chosen_word:
        lives -= 1

        if lives == 0:
            game_over = True

            # TODO 7: - Update the print statement below to give the user the correct word they were trying to guess.
            print(f"***********************YOU LOSE***********************")
            print(f"You were trying to guess [chosen_word]")

    if "_" not in display:
        game_over = True
        print("*****************************YOU WIN*****************************")
```



```python
    if "_" not in display:
        game_over = True
        print("*****************************YOU WIN*****************************")

    # TODO-2: - Update the code below to use the stages List from the file hangman_art.py
    from hangman_art import stages
    print(stages[lives])
```