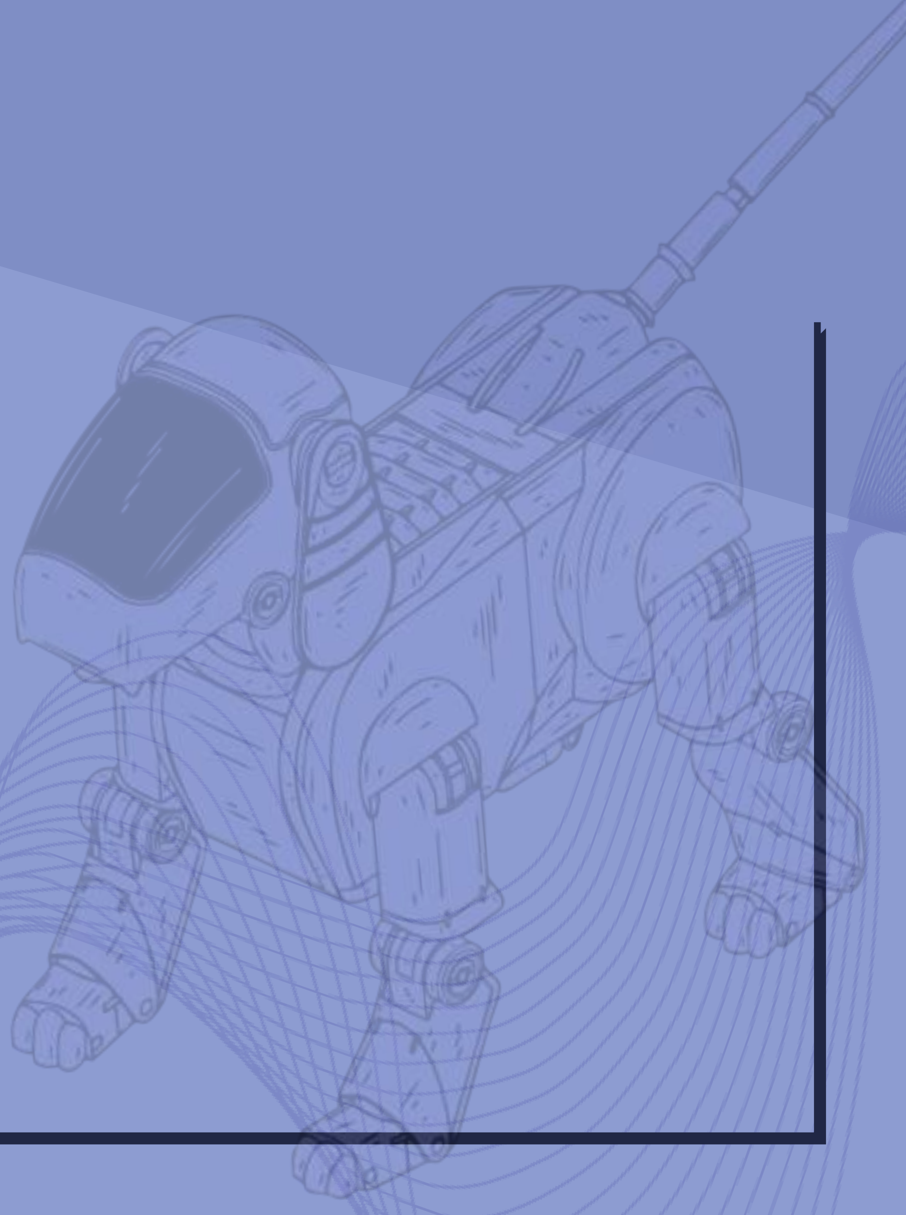
Process

Your code should…

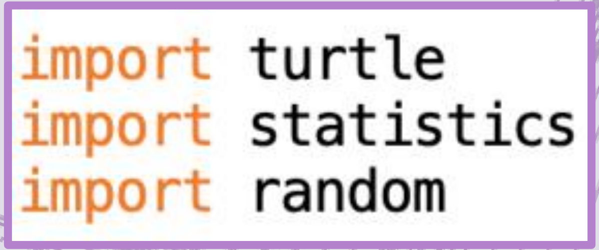
- Ask the user to either enter a letter or the whole word
- If they guess the whole word correctly, they will win and the game will end
- If they guess a letter correctly, the user will be informed of the position of the letter within the secret word
- This code will repeat, until the user runs out of lives or until the user eventually guesses the whole word



cair
4 YOUTH

Python libraries and modules

Python libraries are a set of useful functions that eliminate the need for writing codes from scratch.



```
import turtle
import statistics
import random
```

They can be brought into the program using the "import" keyword and can save valuable time when writing complex programmes.

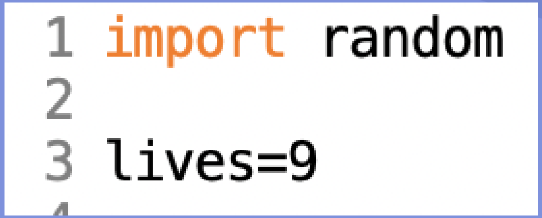
One common example of a python library that we will be using in this code is the "turtle" library which enables us access to graphical functions in python.

Step 1

Importing the random module and variables

Line 1: imports the random module which will be used to generate a random word from the list of potential words stored in the programme.

Line 2: creates a variable called “lives” which stores the number of lives that the user will start with (9). Remember, a variable is a named location in memory that stores values that can be changed during the running of the program- ”lives” will decrement every time the user guesses a letter or the final word wrong.



```
1 import random
2
3 lives=9
```

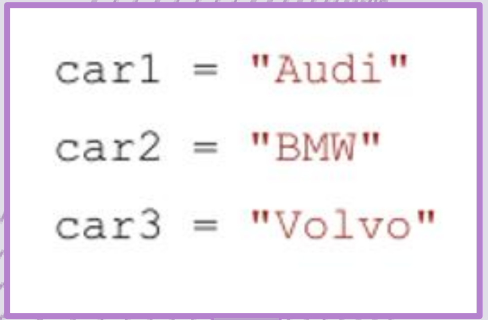
Lists

A list is a container which can hold several of items of the same type.

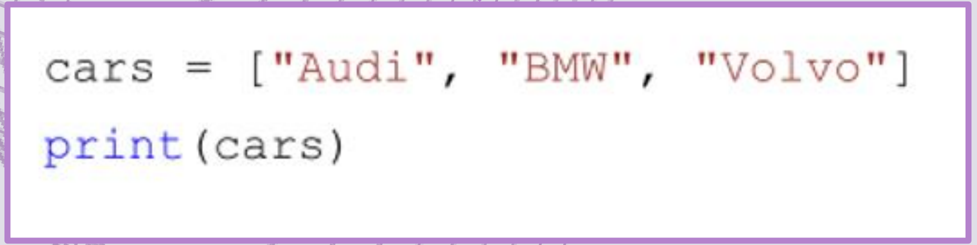
Each item stored in a list is called an element, and its index is the position where it is within the array. Remember that **Python** starts counting from 0, so the first element is at [0] not [1].

If you have a list of items (for example car names) storing the values individually means that it can be hard to find a specific value (especially if you want to store 300 cars not 3).

We will use lists in our program to store information that we will read from the external file.



```
car1 = "Audi"
car2 = "BMW"
car3 = "Volvo"
```

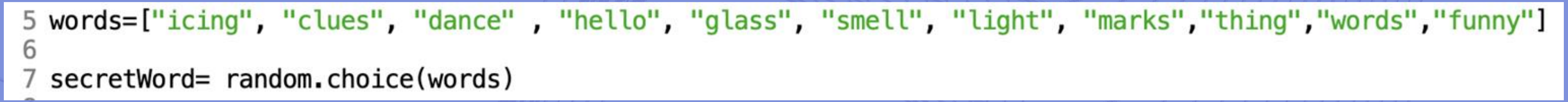


```
cars = ["Audi", "BMW", "Volvo"]
print(cars)
```

Step 2

The secret words

Line 5: a list called “words” is created which will store a list of possible 5 letter words. All words in the list must be 5 letters, as the final code will not work if not. You can use the words from the list below or come up with your own to make your game original and exciting.



```
5 words=["icing", "clues", "dance" , "hello", "glass", "smell", "light", "marks","thing","words","funny"]
6
7 secretWord= random.choice(words)
```

Line 7: a variable called “SecretWord’” is created which will store a random word from the ”words” list created in line 5. This is the word that the user will be trying to guess during the running of the game, and will be a different word each time.

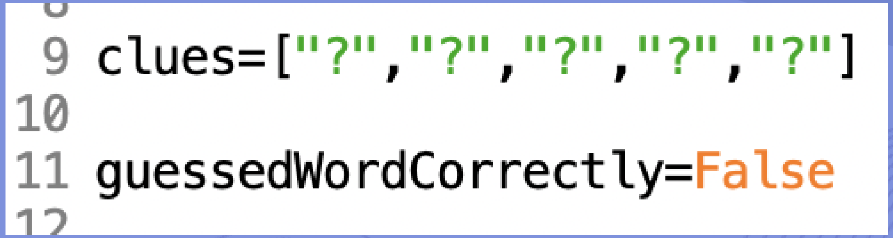
cair
4 YOUTH

Step 3

The clues...

Step 9: much like the grey boxes in the wordle game that you just played, these question marks will act as placeholders until the user guesses some letters that are in the secret word. These question marks will then be replaced with correct letters in the correct places.

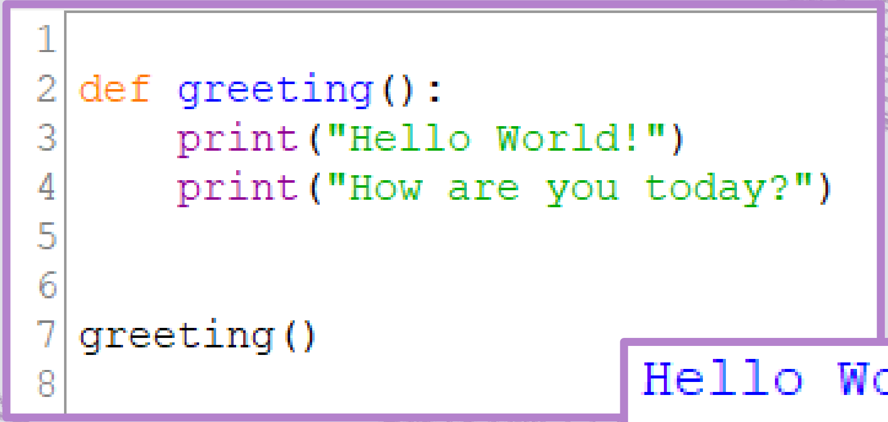
Step 10: “GuessedWordCorrectly” is a Boolean variable which will be changed to True when the user guesses the “SecretWord” correctly.



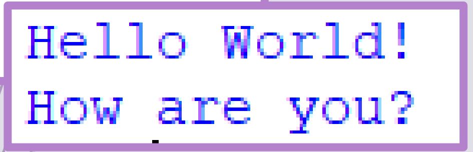
```
clues=["?","?","?","?","?"]

guessedWordCorrectly=False
```

Subroutines

Subroutines are sets of instructions designed to perform a frequently used operation within a programme.



```python

def greeting():
    print("Hello World!")
    print("How are you today?")


greeting()

```



```
Hello World!
How are you?
```

Subroutines can store code and will only be run when ‘called’.

There are two main types of the subroutine: procedures and functions.

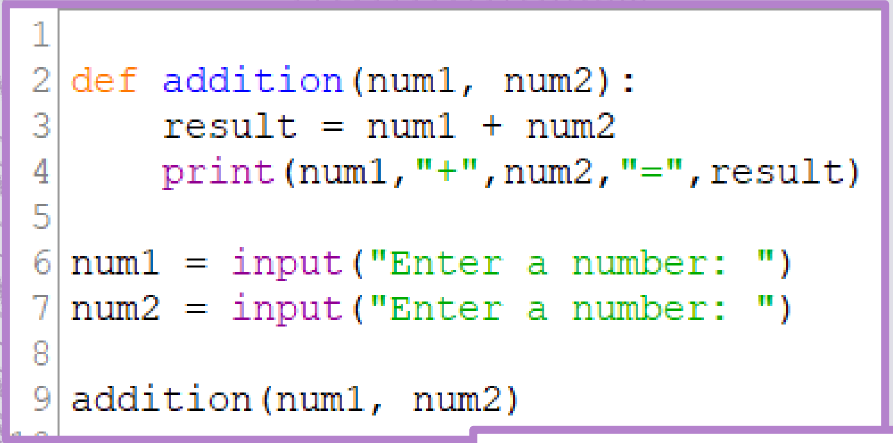
Procedures are not required to return a value, whereas functions must return a value.

Subroutines are great ways of writing more maintainable code and lead to more structured, organized and understandable programmes.

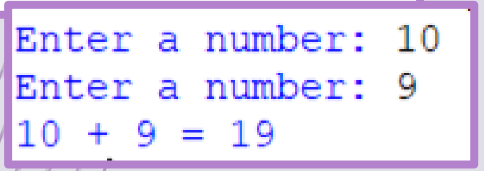
Parameters and arguments

Information can be passed into subroutines using parameters, which act as placeholders for the real values assigned when called.

The values put in the brackets when the subroutine is called are called arguments.



```python

def addition(num1, num2):
    result = num1 + num2
    print(num1,"+",num2,"=",result)

num1 = input("Enter a number: ")
num2 = input("Enter a number: ")

addition(num1, num2)
```



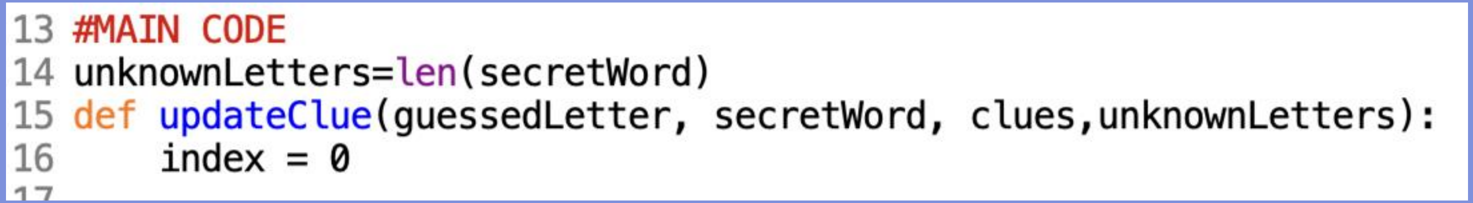
```
Enter a number: 10
Enter a number: 9
10 + 9 = 19
```

cair
4 YOUTH

Step 4

Update clue

Line 14: creates a variable called “UnknownLetters” which will store the length of the secret word (which we already know in this code is 5 letters long).



```
13 #MAIN CODE
14 unknownLetters=len(secretWord)
15 def updateClue(guessedLetter, secretWord, clues,unknownLetters):
16     index = 0
```

Line 15: a subroutine is declared called “updateClue”, which has various parameters passed into it, which are declared later on in the programme.

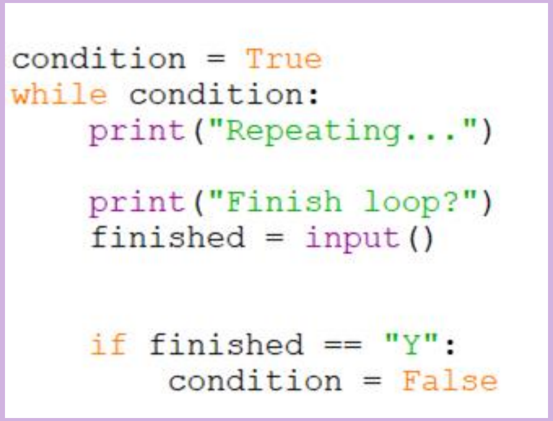
Line 16: a variable called “index” is created which will be used later to look at each possible letter in the random secret word.

Loops

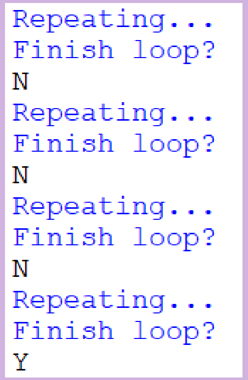
A loop is a sequence of instructions that is continually repeated until a certain condition is reached.

In Python there are two main loops: ‘WHILE Loops’ and ‘FOR Loops’.

While Loops are condition controlled and will repeat until their condition is false.

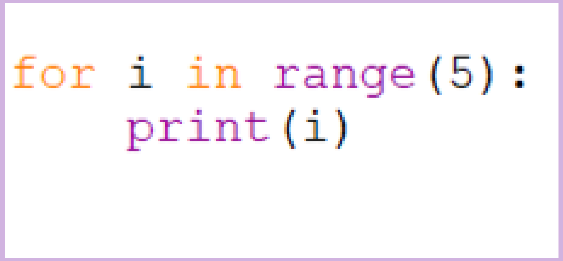


```
condition = True
while condition:
    print("Repeating...")

    print("Finish loop?")
    finished = input()


    if finished == "Y":
        condition = False
```

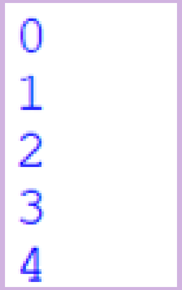


```
Repeating...
Finish loop?
N
Repeating...
Finish loop?
N
Repeating...
Finish loop?
N
Repeating...
Finish loop?
Y
```

For loops are count controlled and will repeat a set number of times.



```
for i in range(5):
    print(i)
```



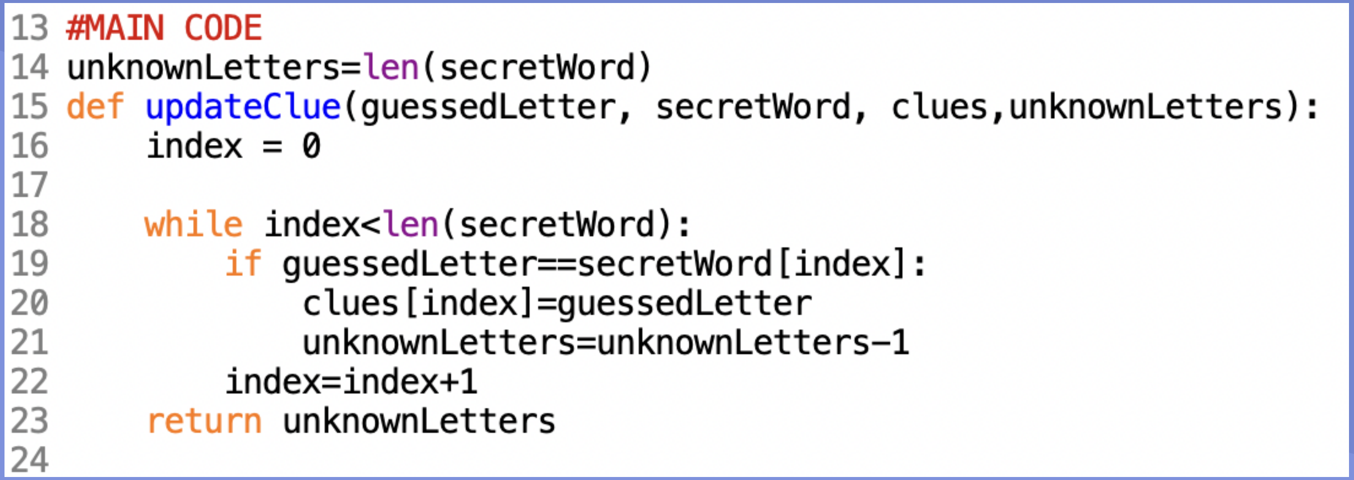
```
0
1
2
3
4
```

Step 5

Update clue while loop

Line 18: the code will continue to run until it has checked each of the 5 letters in the secret word to see if the user has entered a correct letter.

Line 19 and 20: starting with the first letter in the secret word, it is checked to see if the letter entered by the user is the same as the one located in this first position. If it is then the same position in the clues array (that looks like ?????), will be updated with this letter and the number of letters that the user has left is decreased by one.



```
13 #MAIN CODE
14 unknownLetters=len(secretWord)
15 def updateClue(guessedLetter, secretWord, clues,unknownLetters):
16     index = 0
17 
18     while index<len(secretWord):
19         if guessedLetter==secretWord[index]:
20             clues[index]=guessedLetter
21             unknownLetters=unknownLetters-1
22         index=index+1
23     return unknownLetters
24 
```

Line 22: the index will then be increased by one, and the loop will run again, and will do so for each of the remaining letters in the word until it is time for the user to enter their next guess.

Line 23: this “UnknownLetters” variable is returned (making this subroutine a function rather than a procedure).

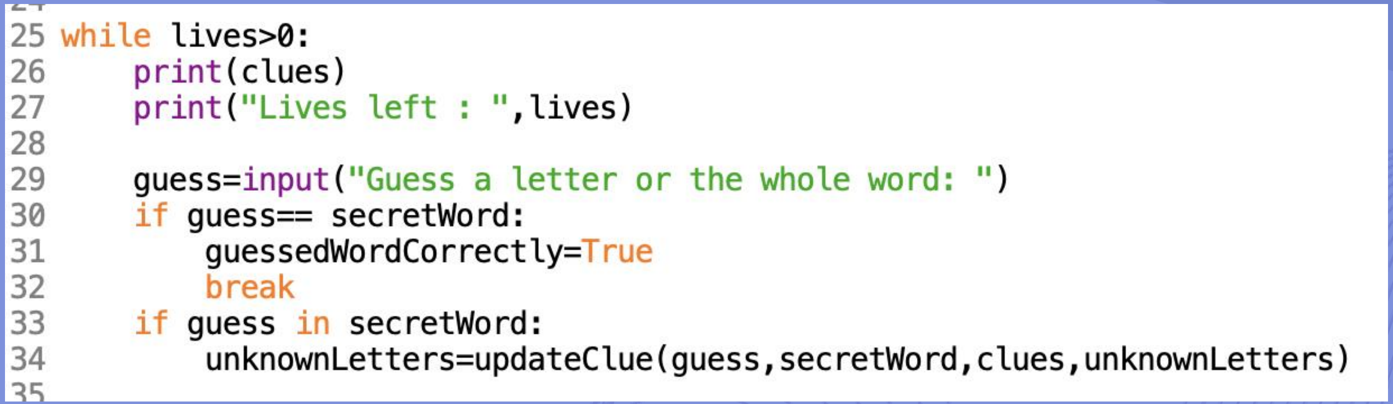
cair 4 YOUTH

Step 6

The main code outside the subroutine

Lines 25, 26 and 27: while the user still has lives, the clues list (which is updated with the letters that the user guesses) and the amount of lives the user has left are displayed.

Line 29: the user is asked to enter a word or just a single letter.



```
25 while lives>0:
26     print(clues)
27     print("Lives left : ",lives)
28 
29     guess=input("Guess a letter or the whole word: ")
30     if guess== secretWord:
31         guessedWordCorrectly=True
32         break
33     if guess in secretWord:
34         unknownLetters=updateClue(guess,secretWord,clues,unknownLetters)
35
```

Line 30, 31 and 32- if the word is the same as the secret word, then the user will have won the game and won't be able to enter any more guesses. This is because GuessedWordCorrectly is changed to True.

Line 33 and 34- if the guess is a letter that is found within the secret word, then the update clue subroutine is run, and the amount of letters that the user has to guess is updated.

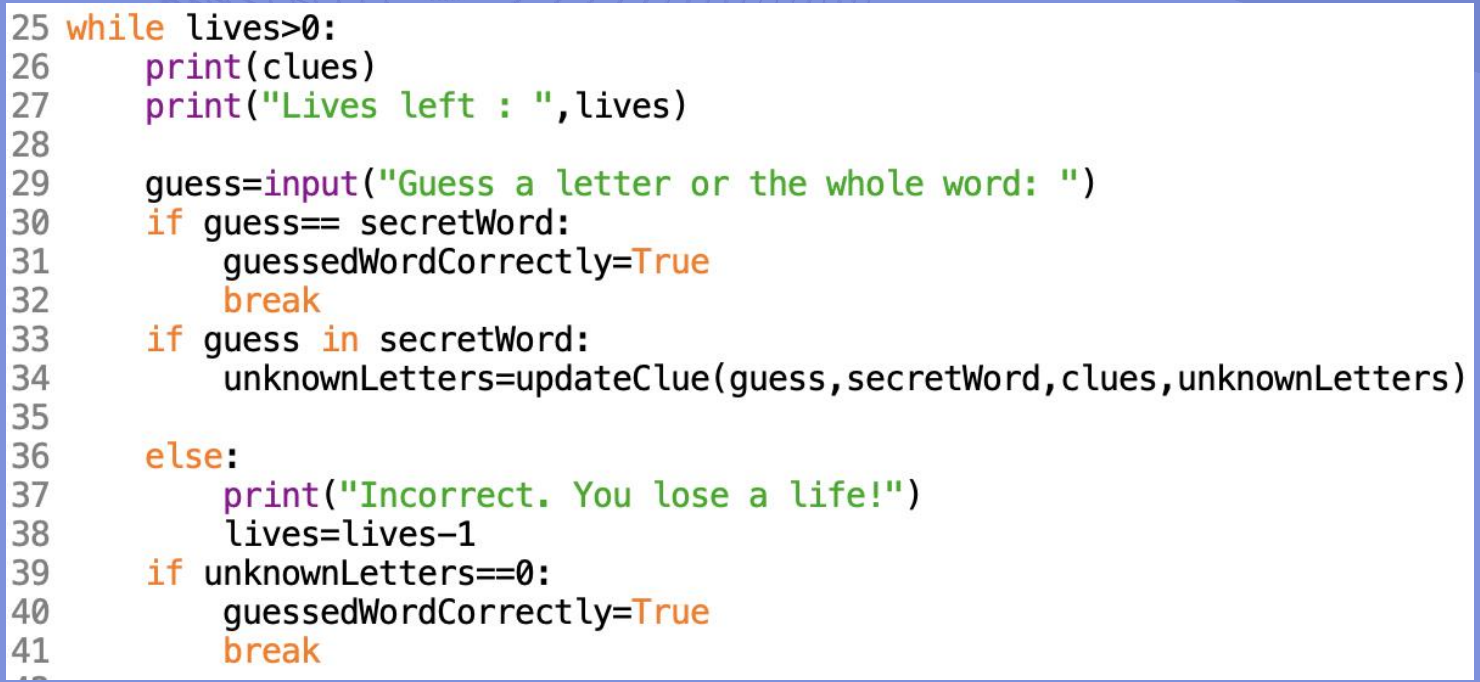
cair 4 YOUTH

Step 7

Guessing no words or letters or guessing all of them

Lines 36, 37, 38- if the user guesses neither a letter nor the whole word correctly, then they will be told, and will lose a life. the code will run again from line 25, granted that the user still has some lives left.

Lines 39, 40, 41- the final check to see if the user has got the answer correct, is if they enter all of the correct letters, in which case the UnknownLetters variable would be changed to 0. If this is the case, then the user will have also won the game because they will have no more letters to guess.



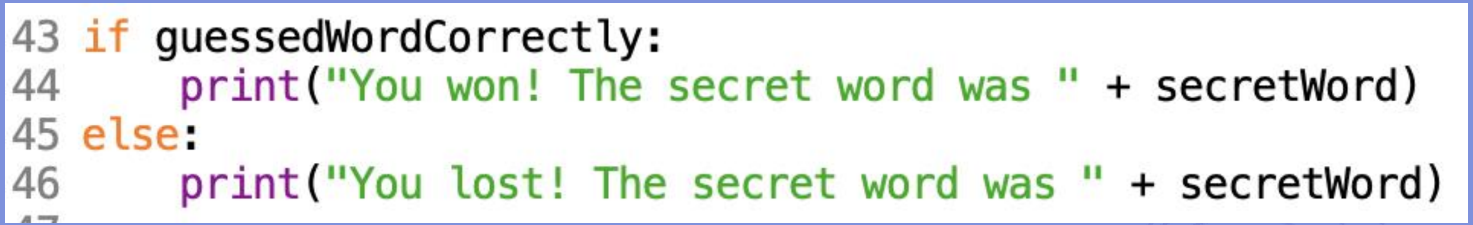
```
25 while lives>0:
26     print(clues)
27     print("Lives left : ",lives)
28
29     guess=input("Guess a letter or the whole word: ")
30     if guess== secretWord:
31         guessedWordCorrectly=True
32         break
33     if guess in secretWord:
34         unknownLetters=updateClue(guess,secretWord,clues,unknownLetters)
35
36     else:
37         print("Incorrect. You lose a life!")
38         lives=lives-1
39     if unknownLetters==0:
40         guessedWordCorrectly=True
41         break
```

cair
4 YOUTH

Step 8

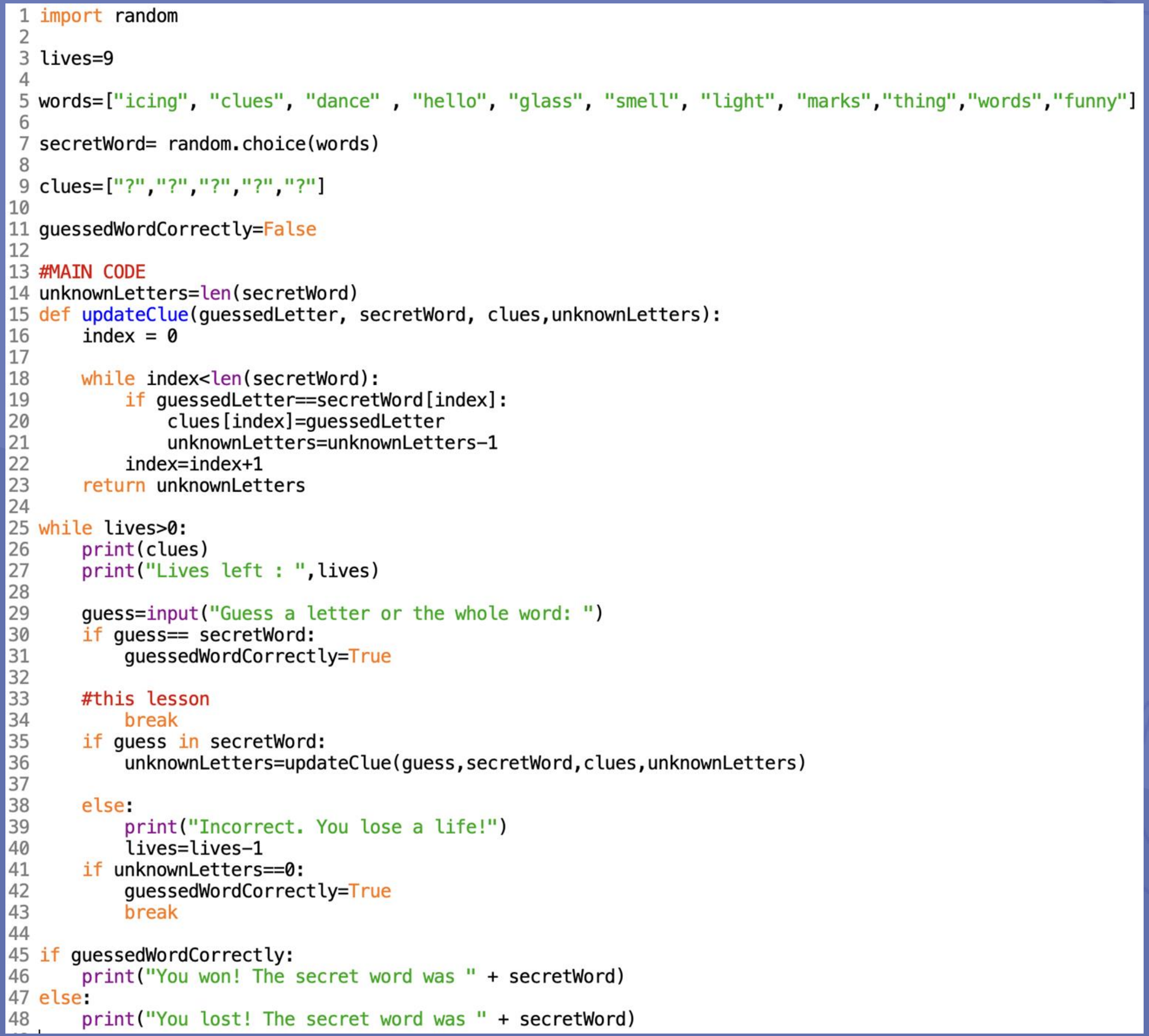
Win or lose?

Lines 43 - end: After the user has either run out of lives or guessed the secret word correctly, the relevance of these two messages will be displayed.

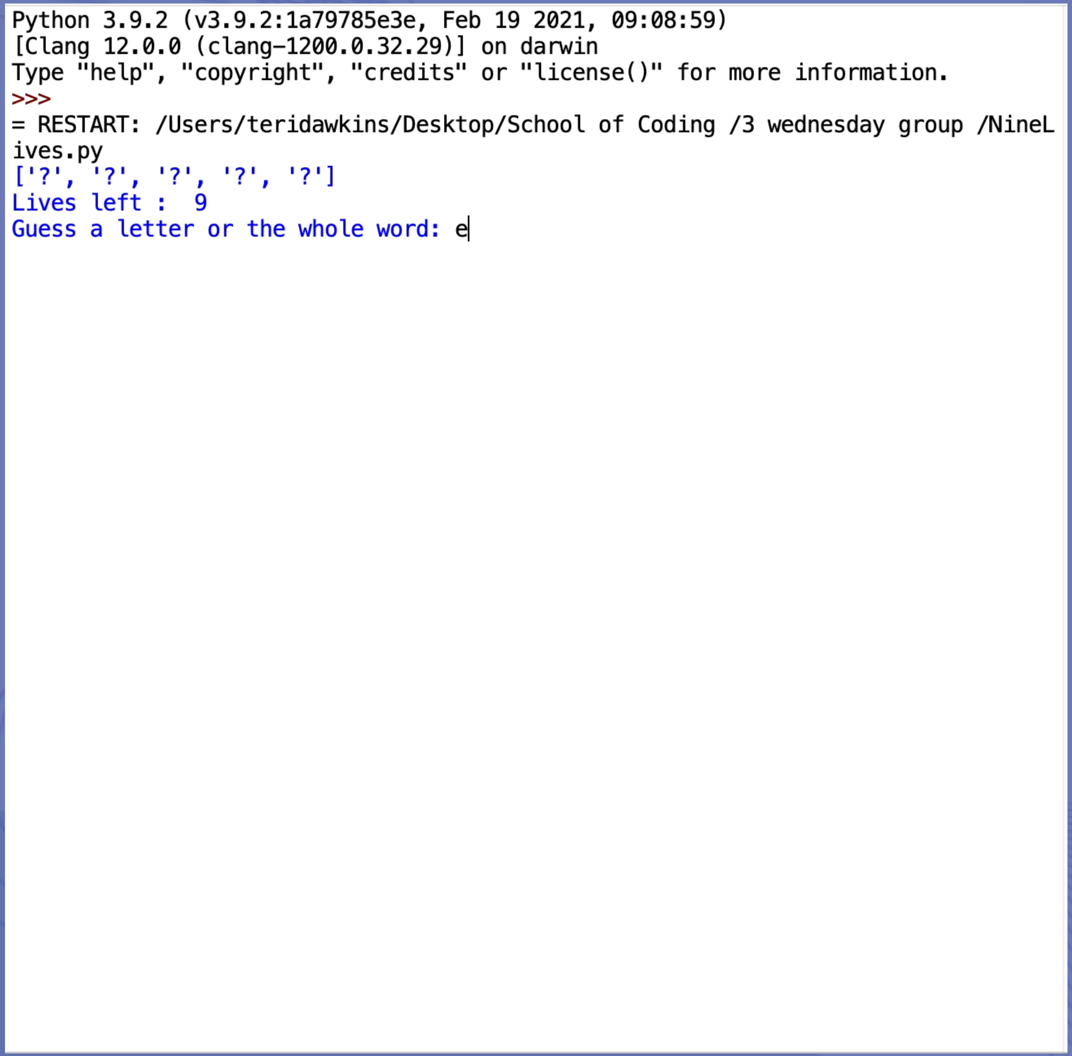


```
43 if guessedWordCorrectly:
44     print("You won! The secret word was " + secretWord)
45 else:
46     print("You lost! The secret word was " + secretWord)
```

The final code...



```python
import random

lives=9

words=["icing", "clues", "dance" , "hello", "glass", "smell", "light", "marks","thing","words","funny"]

secretWord= random.choice(words)

clues=["?","?","?","?","?"]

guessedWordCorrectly=False

#MAIN CODE
unknownLetters=len(secretWord)
def updateClue(guessedLetter, secretWord, clues,unknownLetters):
    index = 0

    while index<len(secretWord):
        if guessedLetter==secretWord[index]:
            clues[index]=guessedLetter
            unknownLetters=unknownLetters-1
        index=index+1
    return unknownLetters

while lives>0:
    print(clues)
    print("Lives left : ",lives)

    guess=input("Guess a letter or the whole word: ")
    if guess== secretWord:
        guessedWordCorrectly=True

    #this lesson
        break
    if guess in secretWord:
        unknownLetters=updateClue(guess,secretWord,clues,unknownLetters)

    else:
        print("Incorrect. You lose a life!")
        lives=lives-1
    if unknownLetters==0:
        guessedWordCorrectly=True
        break

if guessedWordCorrectly:
    print("You won! The secret word was " + secretWord)
else:
    print("You lost! The secret word was " + secretWord)
```



```
Python 3.9.2 (v3.9.2:1a79785e3e, Feb 19 2021, 09:08:59)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: /Users/teridawkins/Desktop/School of Coding /3 wednesday group /NineL
ives.py
['?', '?', '?', '?', '?']
Lives left :  9
Guess a letter or the whole word: e
```