Day 3

While Loops

ORBTRONICS

Objectives

Students should be able to:

- Understand how while loops work and use them.
- Use while loops with counters.

Loops

Loops

- Loops allow us to run some code over and over without manually copying the code.
- When creating a loop, we need to specify what code will be repeated and when the loop should stop.

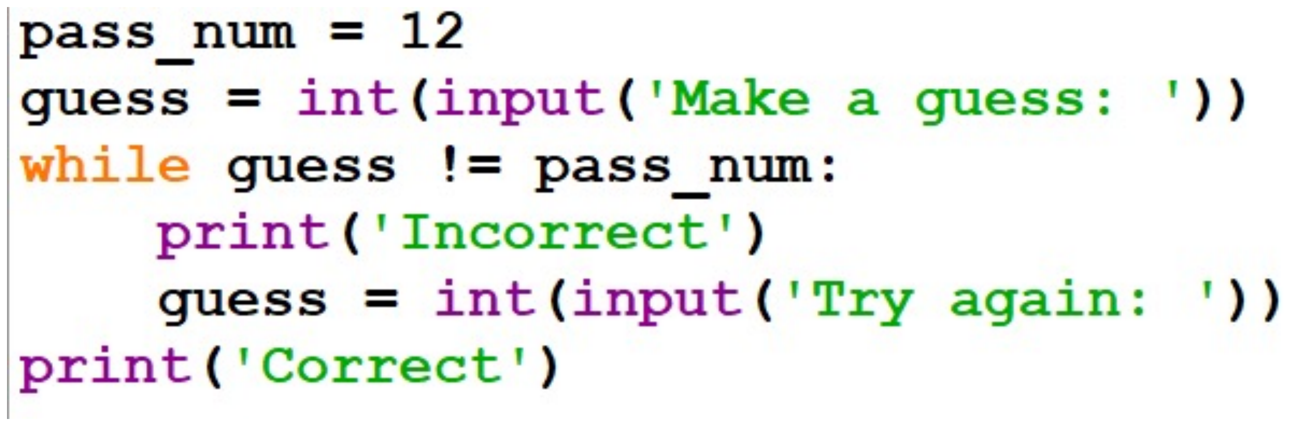
While Loops

While Loops

- A while loop works by first checking some condition, if the condition is true, it runs some code.
- After running that code, it then check the condition again, and if it is still true, it runs the code again.
- This keeps repeating until the condition is false.
- The syntax of a while loop is similar to an if statement, except that the word “while” is used instead of “if”.

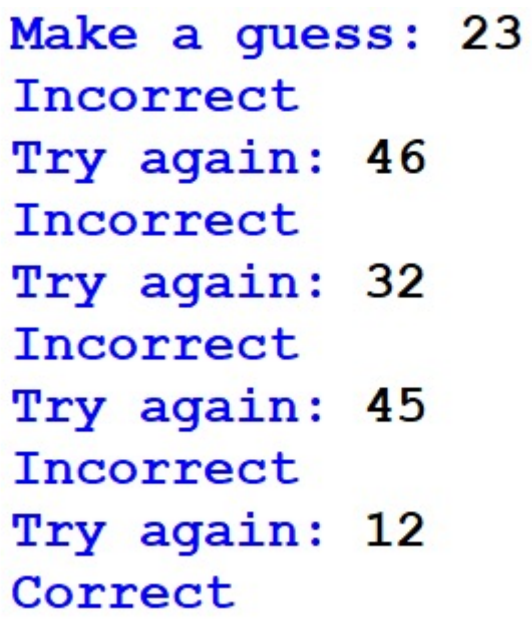
Example 1

<u>Source Code</u>



```python
pass_num = 12
guess = int(input('Make a guess:  '))
while guess != pass_num:
    print('Incorrect')
    guess = int(input('Try again:  '))
print('Correct')
```

<u>Output</u>



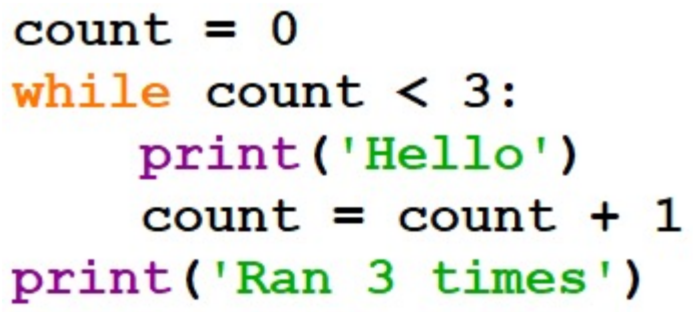
```
Make a guess: 23
Incorrect
Try again: 46
Incorrect
Try again: 32
Incorrect
Try again: 45
Incorrect
Try again: 12
Correct
```

Example 1 - Explanation

- Firstly, we set the variable pass_num to 12.
- We want the user to keep guessing this number until they guess it correctly, so we ask them to enter their guess on line 2.
- The while loop checks to see if their guess is not equal to the pass number. If they’re not equal, then it will run the code inside the while loop, which will tell the user they got it incorrect, and will ask them to try again.
- The while loop then checks again to see if the guess is not equal to the pass number and if it isn’t, it runs the code inside again. This repeats until the guess is equal to the pass number, meaning the guess was correct.
- The code outside the while loop then runs, which in this case, tells the user that the guess was correct.

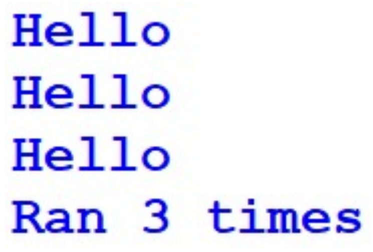
Example 2

Source Code



```
count = 0
while count < 3:
    print('Hello')
    count = count + 1
print('Ran 3 times')
```

Output



```
Hello
Hello
Hello
Ran 3 times
```

Explanation

- Firstly, we set the variable count to 0.
- The while loop checks to see if count is less than 3. It is, so “Hello” is printed out, and then we increase count by 1 so it is now equal to 1.
- The while loop checks again, and count is still less than 3, so “Hello” is printed again and count gets increased to 2.
- This is repeated one more time and count gets increased to 3. Since count is no longer less than 3 the while loop stops, and the final print statement outside the while loop runs. In total, “Hello” was printed 3 times.

Example 3

Source Code

```
pass_num = 12
count = 1
guess = int(input('Make a guess: '))

while guess != pass_num and count < 3:
    print('Incorrect')
    guess = int(input('Try again: '))
    count = count + 1

if guess == pass_num:
    print('Correct')
else:
    print('Out of tries')
```

Output 1

```
Make a guess: 14
Incorrect
Try again: 12
Correct
```

Output 2

```
Make a guess: 15
Incorrect
Try again: 10
Incorrect
Try again: 9
Out of tries
```

Example 3 - Explanation

- We want the user to guess the pass number, but they should only get 3 tries.
- Firstly, we set the variable pass_num to 12, the variable count to 1, and we ask the user to guess the pass number. The count variable represents the number of guesses made.
- The while loop checks to see if their guess is incorrect and if they have made less than 3 guesses. If that condition is met, then we ask them again. The while loop will stop when either they have made 3 guesses, or they guess correctly.
- We then need to check why the while loop stopped so we check if their guess is correct. If so, we say they guessed correctly, and if not, that means they ran out of tries.

Example 4

Program 1

```
count = 0
while count < 100:
     print(count)
     count = count + 1
```

Program 2

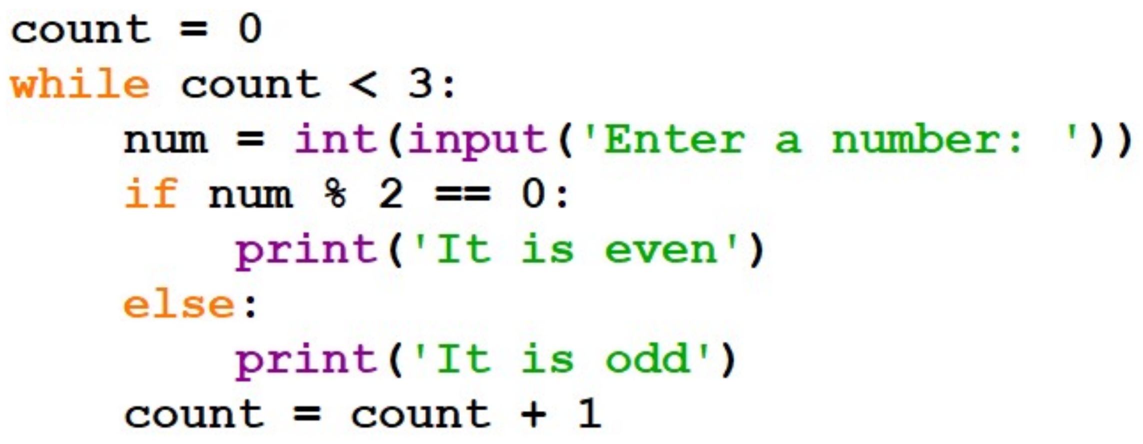
```
count = 10
while count < 50:
     count = count + 2
     print(count)
```

Explanation

- The first program has a count variable which starts at 0. While this count variable is less than 100, we print it out, and increase is by 1.
- Therefore, the numbers 0 to 99 will be printed out.
- The second program starts count at 10, and while it is less than 50, it is increased by 2, and then printed out. Since we increase by 2, every other number gets printed out (in this case the even numbers). Since we increase by 2 before printing out, the even numbers from 12 to 50 will be printed out.

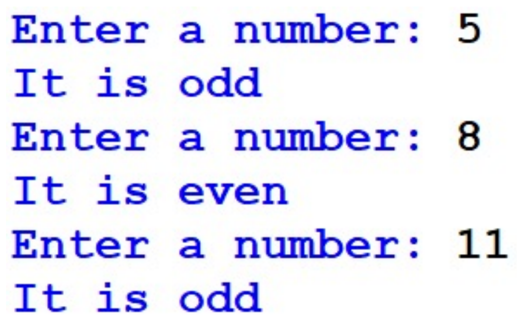
Example 5

Source Code



```
count = 0
while count < 3:
    num = int(input('Enter a number: '))
    if num % 2 == 0:
        print('It is even')
    else:
        print('It is odd')
    count = count + 1
```

Output



```
Enter a number: 5
It is odd
Enter a number: 8
It is even
Enter a number: 11
It is odd
```

Explanation

- This program shows us that we can put if-elif-else statements inside of while loops.
- We can also put while loops inside of if-elif-else statements and while loops inside of other while loops.
- This program lets the user enter 3 numbers and for each of them displays if the number is even or odd.