

Lab 4 – CSE 101 (Spring 2021)

Objectives

The primary objectives are to learn the following:

- i. While loop
- ii. Generating random numbers
- iii. Using the break statement
- iv. Nested for loop

1. How to Construct While Loops

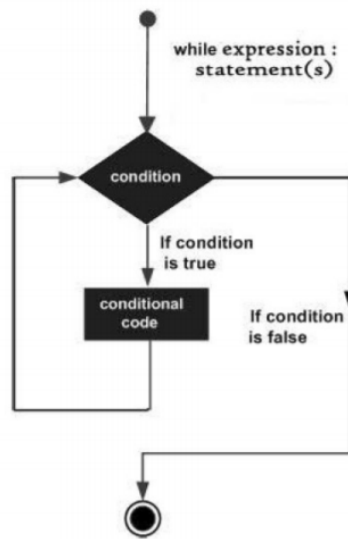


Figure 1: Illustration of While loop

A while loop implements the repeated execution of code based on a given Boolean condition. The code that is in a while block will execute as long as the while statement evaluates to True. You can think of the while loop as similar to a for loop, except that it repeats indefinitely rather than a set amount of times. In a while loop, the program will jump back to the start of the while statement after it finishes executing whatever is in the while block, until the original condition is False. While loops are especially useful if you don't know how many times you'll need to repeat something when you write the code.

In Python, while loops are constructed like so:

```
while [a condition is True]:  
    [do something]
```

Example: In the following program, we will ask for the user to input a password. After the user enters a password there are two possible outcomes: If the password is correct, the while loop will exit. If the password is not correct, the while loop will continue to execute and ask for the password again.

Try running the code to see how it works yourself before moving on to the next step.

password.py program:

```
password = 'secret'
userInput = ''
while userInput != password:
    print('What is the password?')
    userInput = input()
print('Yes, the password is ' + password + '. You may enter.')
```

2. Generating and guessing random numbers

Using the `random` module, we can generate random numbers. Go through the examples below and type in the code in your Python console to see what results you get.

Generate a random number between 1 and 100

To generate a whole number (integer) between one and one hundred use the following slightly different function, still imported from the `random` module:

```
from random import *
# Generate and print a random integer between 1 and 100 (inclusive).
print(randint(1, 100))
```

This will print a random integer. If you want to store it in a variable to use in a later part of your code, you can do so like this:

```
from random import *
# Generate a random number between 50 and 60 and store in the x
variable.
x = randint(50, 60)
print(x)
```

3. Python break statement

In Python, `break` and `continue` statements can alter the flow of a normal loop. The `break` statement terminates the loop in which you call `break`. This causes the program to flow immediately to the next statement after the body of the loop. If you call `break` inside a nested loop (a loop inside another loop), `break` will stop the innermost loop, but keep executing the outer loop.

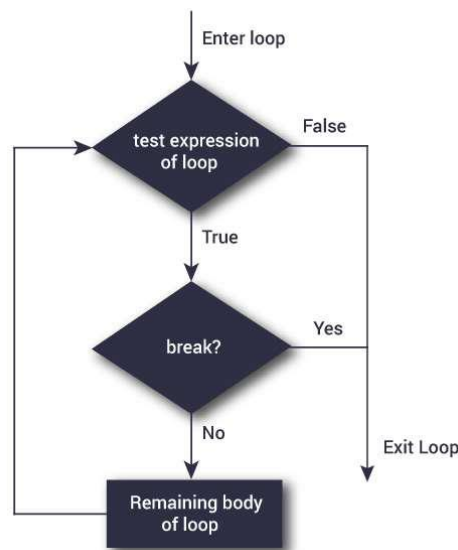


Figure 2: Illustration of break statement

Exercise 1: What does the following program `guess.py` do? Write your answer in the file `explanation.txt` and submit it on Blackboard.

guess.py program

```
import random
number = random.randint(1, 5)
number_of_guesses = 0
while number_of_guesses < 5:
    print('Guess a number between 1 and 5:')
    guess = input()
    guess = int(guess)
    number_of_guesses += 1
    if guess == number:
        break
print('The secret number was', number)
```

Exercise 2: Write a program `randomones.py` that generates random integers between 1 and 6, to simulate rolling a 6-sided die. Using a while loop, have your program keep generating random numbers until the number 1 is generated twice, at which point the program prints out the total number of random numbers it generated and stops.

So an example numbers that are generated might be:

5 6 1 3 5 3 1

After which point the program would stop (since two 1's were generated) and print:

"Generated 7 numbers"

4. Nested For Loops

Loops can be nested in Python, as they can with other programming languages. A nested loop is a loop that occurs within another loop, structurally similar to nested if statements. These are constructed like so:

```
# Outer Loop
for [first iterating variable] in [outer loop]:
    [do something]
    # Nested Loop
    for [second iterating variable] in [nested loop]:
        [do something]
```

Here are some code examples of nested loops:

nestedloop.py program

```
num_list = [1, 2, 3]
alpha_list = ['a', 'b', 'c']
for number in num_list:
    print(number)
    for letter in alpha_list:
        print(letter)
```

pyramid.py program

```
for i in range(1,6):  
    for j in range(i):  
        print("*",end=' ')  
    print("\n",end='')
```

Exercise 3: Create the pyramid similar to the above pyramid.py program using only while loops. You must not use any for loops in your solution. Name your program `whilepyramid.py` and submit it on Blackboard.

Exercise 4: Write a program named `factorial.py` that contains the following two functions:

```
def while_factorial(num)  
def for_factorial(num)
```

These should calculate the factorial of a given number represented by the parameter `num` using a while loop and a for loop respectively. Write test code to test each of your functions.

5. Submit the following files on Blackboard.

1. explanation.txt (explanation of the guess.py program) – 1 pt
2. randomones.py (generating two random 1's and printing the total random numbers generated) – 1 pt
3. whilepyramid.py (Pyramid printing using while loop) – 1 pt
4. factorial.py (factorial program that include factorial functions using for loop and while loop) – 2 pt