Lab 4 – CSE 101 (Spring 2019)

1. Objectives

The primary objectives are[1]:

- To learn While loop,
- Generating random numbers,
- Learn nested For loop.

2. How to Construct While Loops?

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 a repeating conditional statement. After an if statement, the program continues to execute code, but in a while loop, the program jumps back to the start of the while statement until the condition is False. As opposed to for loops that execute a certain number of times, while loops are conditionally based, so you don't need to know how many times to repeat the code going in.

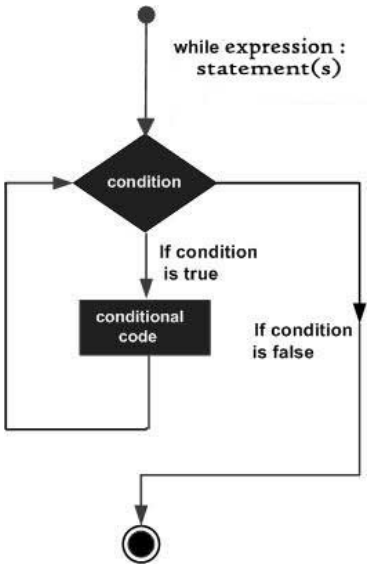

Figure 1: Illustration of While loop

In Python, while loops are constructed like so:

```
while [a condition is True]:

      [do something]
```

[1] Part of this tutorial is taken from the book "How to code in Python" by Lisa Tagliaferri.

Example: In the following program, we'll ask for the user to input a password. While going through this loop, 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.

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.')
```

3. Generating and guessing random numbers

Using the random module, we can generate pseudo-random numbers. The function random() generates a random number between zero and one [0, 0.1 .. 1]. Numbers generated with this module are not truly random but they are enough random for most purposes.

Random number between 0 and 1.

We can generate a (pseudo) random floating point number with this small code:

```
from random import *
print random()     # Generate a pseudo-random number between 0 and 1.
```

Generate a random number between 1 and 100

To generate a whole number (integer) between one and one hundred use:

```
from random import *
print randint(1, 100)    # Pick a random number between 1 and 100.
```

This will print a random integer. If you want to store it in a variable you can use:

```
from random import *
x = randint(1, 100)    # Pick a random number between 1 and 100.
print x
```

Random number between 1 and 10

To generate a random floating point number between 1 and 10 you can use the uniform() function.

```
from random import *
print uniform(1, 10)
```

Picking a random item from a list

Fun with lists

We can shuffle a list with this code:

```
from random import *
items = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
shuffle(items)
print items
```

To pick a random number from a list:

```
from random import *
items = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
x = sample(items,  1)   # Pick a random item from the list
print x[0]
y = sample(items, 4)    # Pick 4 random items from the list
print y
```

We can do the same thing with a list of strings:

```
from random import *
items = ['Alissa','Alice','Marco','Melissa','Sandra','Steve']
x = sample(items,  1)   # Pick a random item from the list
print x[0]
y = sample(items, 4)    # Pick 4 random items from the list
print y
```

Python break statement

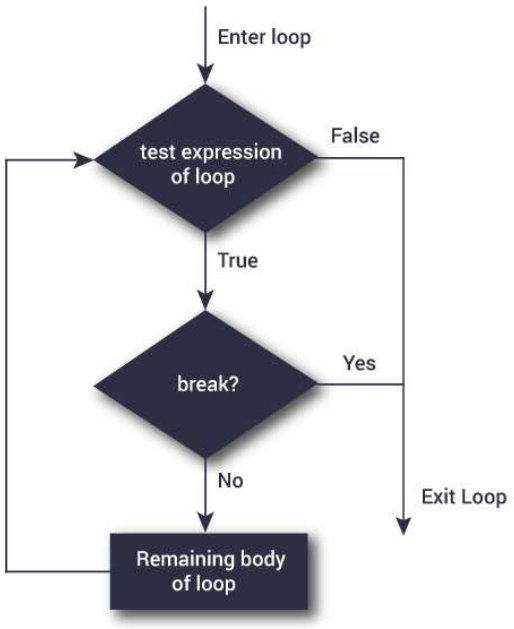

Figure 2: Illustration of break statement

In Python, break and continue statements can alter the flow of a normal loop. The break statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop. If break statement is inside a nested loop (loop inside another loop), break will terminate the innermost loop.

Exercise: 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, 25)
number_of_guesses = 0
while number_of_guesses < 5:
     print('Guess a number between 1 and 25:')
     guess = input()
     guess = int(guess)
     number_of_guesses = number_of_guesses + 1
     if guess == number:
          break
```

3. 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:

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

`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: Create the pyramid similar to printed by pyramid.py program using while loop. Name your program as **`pyramid1.py`** and submit on blackboard.

Exercise: Write a program ***`factorial.py`*** that contains two functions ***`def whilefact(num)`*** and ***`def forfact(num)`*** to calculate factorial of a number represented by argument num.

4. Submit the following programs on blackboard.

```
Explanation.txt

pyramid1.py

factorial.py
```