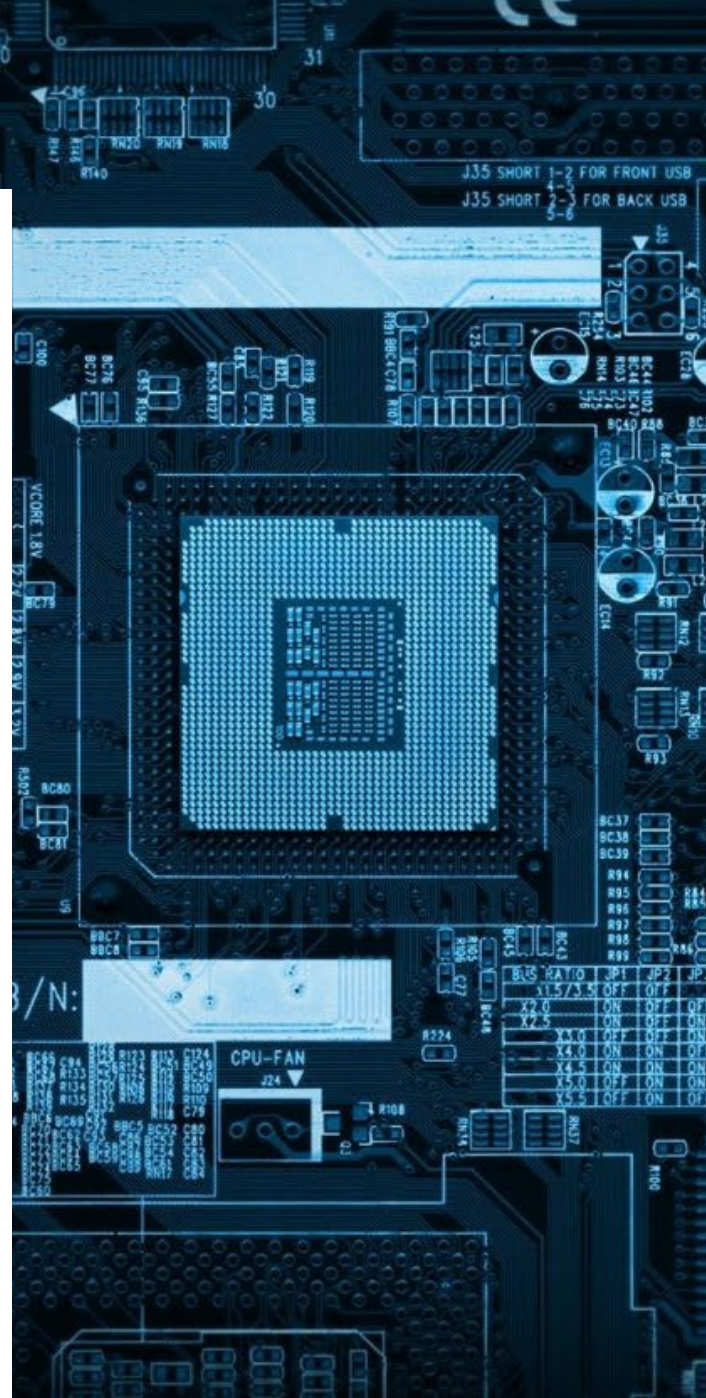


SCMA 249

Computer Programming in Actuarial Science

1/2022

LAB 3 Repetition



LAB 3


Repetition

A loop is a programming construct that enables repetitive processing of a sequence of statements. Python provides two types of loops to its users: the 'for loop' and the 'while loop'. The 'for' and 'while' loops are iteration statements that allow a block of code (the body of the loop) to be repeated a number of times.

3.1 For Loop

Python implements an iterator-based 'for loop'. It is a type of 'for loop' that iterates over a list of items through an explicit or implicit iterator. The loop is introduced by the keyword 'for' which is followed by a random variable name which will contain the values supplied by the object.

The 'for loop' is used when the number of iteration is fixed. The general syntax of 'for loop' is



```
for var in range(start, stop[, step]) :  
    statement
```

The **range** function returns the sequence of integers which started and ended with some given values. The general syntax of **range** is **range(start, stop [, step])**. A start argument is a starting number of the sequence, i.e., lower limit. *By default, it starts with 0 if not specified.*

Example 3.1

What do the followings print?

- 3.1.1 **for** **k** **in** **range**(4) :
 print(k)
- 3.1.2 **for** **x1** **in** **range**(2,6) :
 print(x1)
- 3.1.3 **for** **x2** **in** **range**(2,6,2) :
 print(x2)
- 3.1.4 **for** **x3** **in** **range**(12,0,-3) :
 print(x3)
- 3.1.5 **for** **a** **in** **range**(5,5) :
 print(a)
- 3.1.6 **for** **var** **in** **range**(8,20,-2) :
 print(var)

Answer	0 1 2 3
Answer	2 3 4 5
Answer	2 4
Answer	12 9 6 3
Answer	-
Answer	-

Example 3.2 Fixed iteration number

Find an average height of five students in the class by getting an input from a user.

```
aver_height = 0
#ask for the user input
for i in range(5) :
    print('Person',i+1)
    h = float(input())
    aver_height += h
print('average = ', (aver_height/5))
```

Example 3.3 Nested loop

What does this print?

3.3.1

```
for i in range(2):
    print("out")
    for j in range(2):
        print("in")
```

Answer

```
out
out
in
in
```

3.3.2

```
for i in range(2):
    print(i,"out")
    for j in range(2):
        print(j,"in")
```

Answer

```
0 out
0 in
1 in
1 out
0 in
1 in
```

3.3.3

```
for i in range(2):
    print(i,"out")
    for j in range(2):
        print(j,"in")
    print("xxxxxxx")
```

Answer

```
0 out
0 in
1 in
xxxxxxx
1 out
0 in
1 in
xxxxxxx
```

3.3.4

```
for i in range(2):
    print(i,"out")
    for j in range(2):
        print(j,"in")
    print("xxxxxxx")
```

Answer

```
0 out
0 in
xxxxxxx
1 in
xxxxxxx
1 out
0 in
xxxxxxx
1 in
xxxxxxx
```

3.2 While Loop

In the previous section, we have learned about for loop. The characteristic of for loop is a loop used when we want to repeat a thing with a specific number. Sometimes, though, we need to repeat something, but we don't know ahead of time exactly how many times it has to be repeated. Another loop will be taken into account for this situation which is **while loop**.

While loops repeat as long as a certain boolean condition is true. The syntax of while loop is

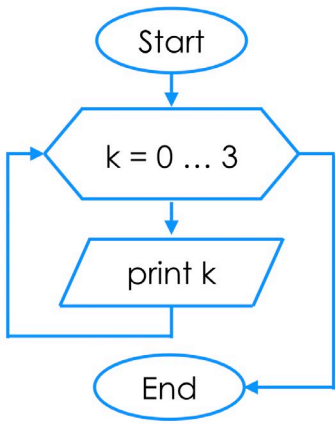
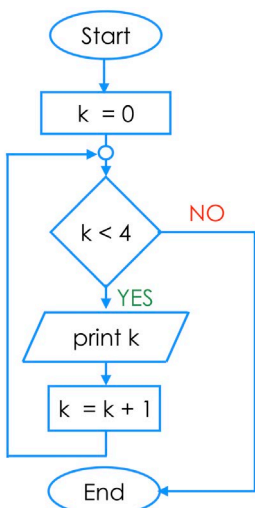
```
while condition :  
    statement
```

Example 3.4

Compile the following program and compare the result with the one in Example 3.1.1.

```
k = 0  
while k < 4:  
    print(k)  
    k += 1
```

Consider the results from Examples 3.1.1 (for loop) and 3.4 (while loop).

FOR LOOP	WHILE LOOP
<pre>for k in range(4) : print(k)</pre>	<pre>k = 0 while k < 4: print(k) k += 1</pre>
 <pre>graph TD Start([Start]) --> Range{k = 0 ... 3} Range --> Print[/print k/] Print --> End([End])</pre>	 <pre>graph TD Start([Start]) --> Init[k = 0] Init --> Decision{k < 4} Decision -- YES --> Print[/print k/] Print --> Increment[k = k + 1] Increment --> Decision Decision -- NO --> End([End])</pre>

Example 3.5

In Example 2.7, we wrote a program that played a simple random number guessing game. The problem with that program is that the player only gets one guess. We can, in a sense, replace the if statement in that program with a while loop to create a program that allows the user to keep guessing until they get it right.

```
from random import randint
secret_num = randint(1,10)
guess = 0
while guess != secret_num:
    guess = eval(input('Guess the secret number: '))
print('You finally got it!')
```

3.3 Break statement

A Python **break** statement ends the present loop and instructs the interpreter to start executing the next statement after the loop. It can be used in both 'for' and 'while' loops. Besides leading the program to the statement after the loop, a **break** statement also prevents the execution of the 'else' statement.

Example 3.6

Here is a program that allows the user to enter up to 10 numbers. The user can stop early by entering a negative number.

```
for i in range(10):
    num = eval(input('Enter number: '))
    if num<0:
        break
```

3.4 The else statement

There is an optional **else** that you can use with **break** statements. The code indented under the **else** gets executed only if the loop completes without a **break** happening.

Example 3.7

This is a simple example based off of Example 3.6 of the previous section.

```
for i in range(10):
    num = eval(input('Enter number: '))
    if num<0:
        print('Stopped early')
        break
else:
    print('User entered all ten values')
```

Example 3.8

Here are two ways to check if an integer **num** is prime. A prime number is a number whose only divisors are 1 and itself. The approach on the left uses a while loop, while the approach on the right uses a for/break loop:

```
num = eval(input('Enter the number:'))
i = 2
while i < num and num%i != 0:
    i = i + 1
if i == num:
    print('Prime')
else:
    print('Not prime')
```

Example 3.9a

```
s='amornsamankul'
n = 1
for i in s:
    print('List item', n, '=',i)
    n +=1
```

Example 3.9b

```
s='Amornsamankul'
for i in s:
    print(i, end="")
```

Example 3.10

```
for i in range(1,10):
    if i%2 == 0 :
        print(i, 'is Even number')
    else :
        print(i, 'is Odd number')
```

Example 3.11

```
x = y = 0
print('+-----+')
for row in range(1,11) :
    print(' ', end = "")
    print(' | ', end = "")
    for column in range(1,11) :
        y = x + column
        print(' ', end = "")
        if y < 10 :
            print(end = ' ')
        print(y, end=' ')
    x = y
    if y == 100:
        print(' | ')
    else:
        print(' | ')
print('+-----+')
```

Example 3.12

```
phone = input('Enter your phone number : ')
for tel in phone:
    if tel == '-':
        continue
    print(tel, end = "")
print("\nEnd of loop")
```

References

- <https://realpython.com/python-data-types/#integers>
- Kittipon P, Kittipob P, Somchai P, Sukree S. Python 101. V1.0.2., Chulalongkorn University Printing House; 2018.
- Andrew J. Python: The Ultimate Beginners Guide!., CreateSpace Independent Publishing Platform; 2016.
- Brian H. A Practical Introduction to Python Programming., Crative Commons Attribution-Noncommercial-Share Alike 3.0; 2015.