Python Loops

ython has two primitive loop commands:

- `while` loops
- `for` loops

What is while loop in Python?

The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true.

We generally use this loop when we don't know the number of times to iterate beforehand.

Syntax of while Loop in Python

```
while test_expression:

    Body of while
```

In the while loop, test expression is checked first. The body of the loop is entered only if the `test_expression` evaluates to `True`. After one iteration, the test expression is checked again. This process continues until the `test_expression` evaluates to `False`.
In Python, the body of the while loop is determined through indentation.

The body starts with indentation and the first unindented line marks the end.

Python interprets any non-zero value as `True`. `None` and `0` are interpreted as `False`.

Flowchart of while Loop

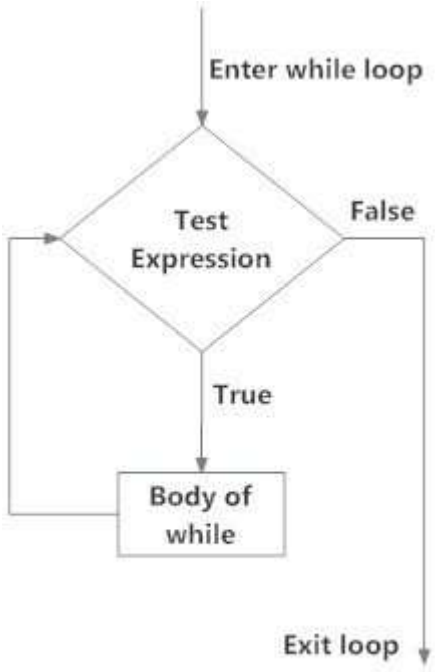

Fig: operation of while loop

Flowchart for while loop in Python

Example:

Print i as long as i is less than 6:

```
i = 1

while i < 6:

  print(i)

  i = i+1
```

Note: remember to increment i, or else the loop will continue forever.

The `while` loop requires relevant variables to be ready, in this example we need to define an indexing variable, `i`, which we set to 1.

The break statement:

With the break statement we can stop the loop even if the while condition is true:

Example:

```
i = 1

while i < 6:

  print(i)
```

```
  if (i == 3):
    break
  i += 1
```

The continue Statement:

With the `continue` statement we can stop the current iteration, and continue with the next:

Continue to the next iteration if i is 3:

Example:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)

# Note that number 3 is missing in the result
```

With the `else` statement we can run a block of code once when the condition no longer is true:

Print a message once the condition is false:

Example:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

```
# Program to add natural
# numbers up to
# sum = 1+2+3+...+n

# To take input from the user,
# n = int(input("Enter n: "))

n = 10

# initialize sum and counter
sum = 0
i = 1

while i <= n:
    sum = sum + i
    i = i+1    # update counter

# print the sum
print("The sum is", sum)
```

In the above program, the test expression will be `True` as long as our counter variable `i` is less than or equal to `n` (10 in our program).

We need to increase the value of the counter variable in the body of the loop. This is very important (and mostly forgotten). Failing to do so will result in an infinite loop (never-ending loop).

Finally, the result is displayed.

Python For Loops

A `for` loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the `for` keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the `for` loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Example

Print each fruit in a fruit list:

```
fruits = ["apple", "banana", "cherry"]

for x in fruits:

  print(x)
```

The `for` loop does not require an indexing variable to set beforehand.

Looping Through a String

Even strings are iterable objects, they contain a sequence of characters:

Example

Loop through the letters in the word "banana":

```
for x in "banana":

  print(x)
```

The break Statement

With the `break` statement we can stop the loop before it has looped through all the items:

Example

Exit the loop when `x` is "banana":

```
fruits = ["apple", "banana", "cherry"]

for x in fruits:

  print(x)

  if x == "banana":

    break
```

```
fruits = ["apple", "banana", "cherry"]

for x in fruits:

  print(x)

  if x == "banana":

    break
```

Example

Exit the loop when `x` is "banana", but this time the break comes before the print:

```
fruits = ["apple", "banana", "cherry"]

for x in fruits:

  if x == "banana":

    break

  print(x)
```

The continue Statement

With the `continue` statement we can stop the current iteration of the loop, and continue with the next:

Example

Do not print banana:

```
fruits = ["apple", "banana", "cherry"]

for x in fruits:

  if x == "banana":

    continue

  print(x)
```

The range() Function

To loop through a set of code a specified number of times, we can use the `range()` function,

The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Example

Using the range() function:

```
for x in range(6):
  print(x)
```

Note that `range(6)` is not the values of 0 to 6, but the values 0 to 5.

The `range()` function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: `range(2, 6)`, which means values from 2 to 6 (but not including 6):

Example

Using the start parameter:

```
for x in range(2, 6):
  print(x)
```

The `range()` function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: `range(2, 30, 3)`:

Example

Increment the sequence with 3 (default is 1):

```
for x in range(2, 30, 3):
  print(x)
```

Else in For Loop

The `else` keyword in a `for` loop specifies a block of code to be executed when the loop is finished:

Example

Print all numbers from 0 to 5, and print a message when the loop has ended:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Note: The `else` block will NOT be executed if the loop is stopped by a `break` statement.

Example

Break the loop when `x` is 3, and see what happens with the `else` block:

```
for x in range(6):
  if x == 3: break
  print(x)
else:
  print("Finally finished!")


#If the loop breaks, the else block is not executed.
```

Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

Example

Print each adjective for every fruit:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]


for x in adj:
```

```
  for y in fruits:

    print(x, y)
```

The pass Statement

`for` loops cannot be empty, but if you for some reason have a `for` loop with no content, put in the `pass` statement to avoid getting an error.

```
for x in [0, 1, 2]:

  pass

# having an empty for loop like this, would raise an error without the pass statement
```

Python Loop Exercises

Practice all exercises using python IDE and see the outputs:

1) Take 10 integers from keyboard using loop and print their average value on the screen.

Solution:

```
sum = 0
i = 10
while i>0:
print (“Enter number:”)
number = input()
sum = sum+num
i = i-1
print (“Average is:”, sum/10)
```

2) Print the following patterns using loop:

Solution:

```
#a
i = 1
while i<=4:
  print "*"*i
  i = i+1

#b
i = 1
j = 2
while i>=1:
  a =  " "*j+"*"*i+" "*j
  print a
  i = i+2
  j = j-1
  if i>5:
    break
i = 3
j = 1
while i>=1:
  a =  " "*j+"*"*i+" "*j
  print a
  i = i-2
  j = j+1

#c
#Do yourself
```

3) Print multiplication table of 24, 50 and 29 using loop.

Solution:

```
i = 1
while i<=10:
  print 24*i
  i = i+1
```

4) Factorial of any number n is represented by n! and is equal to 1*2*3*....*(n-1)*n. E.g.-
4! = 1*2*3*4 = 24
3! = 3*2*1 = 6
2! = 2*1 = 2
Also,
1! = 1

0! = 1

Write a program to calculate factorial of a number.

Solution:

```
print "Enter number"
number = input()
fac = 1
if number == 0:
  print 1
else:
  while number>=1:
    fac = fac*number
    number = number-1
  print fac
```

5) Find the output:

```
5 * 1 = 5

5 * 2 = 10

5 * 3 = 15

5 * 4 = 20

5 * 5 = 25
```

Solution:

```
a = 5


b = 1


while b <= 5:


        print ("%d * %d = %d" %(a, b, a*b))


        b+=1
```

6)

```
# Python program to illustrate
# while loop
```

```
count = 0
while (count < 3):
    count = count + 1
    print("Hello Geek")

7)
Prints all letters except 'e' and 's'
i = 0
a = 'geeksforgeeks'

while i < len(a):
    if a[i] == 'e' or a[i] == 's':
        i += 1
        continue

    print('Current Letter :', a[i])
    i += 1

8)
# break the loop as soon it sees 'e'
# or 's'
i = 0
a = 'geeksforgeeks'

while i < len(a):
    if a[i] == 'e' or a[i] == 's':
        i += 1
        break

    print('Current Letter :', a[i])
    i += 1

9)
Python program to demonstrate
# while-else loop

i = 0
while i < 4:
    i = i+1
    print(i)
else:  # Executed because no break in for
    print("No Break\n")

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

```
    break
else:  # Not executed as there is a break
    print("No Break")
```

10)

```
a = int(input('Enter a number (-1 to quit): '))

while a != -1:
    a = int(input('Enter a number (-1 to quit): '))
```

11)

```
# Python program to
# demonstrate break statement

s = 'geeksforgeeks'
# Using for loop
for letter in s:

    print(letter)
    # break the loop as soon it sees 'e'
    # or 's'
    if letter == 'e' or letter == 's':
        break

print("Out of for loop")
print()

i = 0

# Using while loop
while True:
    print(s[i])

    # break the loop as soon it sees 'e'
    # or 's'
    if s[i] == 'e' or s[i] == 's':
        break
    i += 1

print("Out of while loop")
```

12)

```
Python program to
```

```
# demonstrate continue
# statement

# loop from 1 to 10
for i in range(1, 11):

    # If i is equals to 6,
    # continue to next iteration
    # without printing
    if i == 6:
        continue
    else:
        # otherwise print the value
        # of i
        print(i, end=" ")
```