Iterative Control Structures (Python Loops)

What is a Loop?
A loop is a sequence of instructions that is continually repeated until a certain condition is reached.

Why Loop?
In a loop structure, the loop asks a question. If the answer requires an action, it is executed. The same question is asked again and again until no further action is required. Each time the question is asked is called an iteration.

TYPES OF LOOPS in PYTHON:
1. for 2. while
3. nested loops 4. break and continue (Loop Control Statements)

for loop in PYTHON:
It is used to iterate over the items of any sequence including the Python list, string, tuple etc.

Syntax
```
for variable_name in sequence :
    statement_1
    statement_2
```

Example:
```
for char in "PYTHON":
    print("The Character is: ",char)
```

Explanation:
for ==> Keyword or Iterator
char ==> Variable
in ==> Operator
"PYTHON" ==> String or Iterable
for char in "PYTHON": ==> Iteration(s)

Example:
```
course_list = ["Big Data", "Hadoop", "Spark", "Scala"]
for list in course_list:
        print(list)
```

Example:
```
primes = [2, 3, 5, 7]
for prime in primes:
    print(prime)
```

Example:
```
languages = ["C", "C++", "Perl", "Python"]
for x in languages:
    print(x)
```

Example:
```
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]
Sum = 0
for val in numbers:
     Sum = Sum+val
```

```
print("The sum is", Sum)
```

Example:

```
for num in [10,20,30,40,50]:
    if num >=40:
        print("Greater Than equal 40 is: ",num)
    else:
        print("Lesser Than 40 is: ",num)
```

Example:

```
dataset = ['BigData', 'python', 'Deep Learning']
for data in dataset:
    print(data.upper())
```

Collection Controlled Loops
These are also implemented with the help of for loop.We need a collection object as a source. Python have various collection class like: list, tuple, set, dict.

Example:

```
# with list
data = [12,45,67,23,15]
for item in data:
    print(item)
```

Example:

```
# with tuple
data = (12,45,67,23,15)
for item in data:
    print(item)
```

Example:

```
# with set
data = {12,45,67,23,15}
for item in data:
    print(item)
```

Example:

```
# with dictionaies
data = {1:"PY",2:"DS",3:"ML",4:"DL",5:"AI"}
for item in data:
    print(item)
```

Example:

```
# Program to iterate through a list using indexing
genre = ['pop', 'rock', 'jazz','hindustani']
# iterate over the list using index
for i in range(len(genre)):
    print("I like", genre[i])
```

range() and xrange() functions in PYTHON:
xrange() -PYTHON-2.x Version ==> Renamed as range()
range() -PYTHON-3.x Version

Range Controlled Loops:
Range Function
It generates lists containing arithmetic progression. It returns a

list of consecutive integers. The function has one, two or three parameters where last two parameters are optional. It is widely used in for loops.

3 variations of range() function:
range(stop) - Starts from 0 till (stop - 1)
range(start,stop) - Ends at (stop - 1)
range(start,stop,step) - Step can not be 0, default is 1

Syntax:
range(a) range(a,b) range(a,b,c)

range(a) : Generates a sequence of numbers from 0 to a, excluding a, incrementing by 1.

Syntax
for <variable> in range(<number>):

Example:
```
list(range(5))
```

Example:
```
for a in range(4):
  print(a)
```

Example:
```
MyItems=[1,2,3,4]
for i in range(len(MyItems)):
      print(MyItems[i])
```

Example:
```
MyData=int(input("Enter Any Number in Range: "))
for data in range(MyData):
    print(data)
```

Example:
```
print(range(10))#range(0, 10)
print(list(range(10)))#[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(tuple(range(10)))#(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
print(set(range(10)))#{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
print(frozenset(range(10)))#frozenset({0, 1, 2, 3, 4,5, 6, 7, 8, 9})
```

range(a,b) : Generates a sequence of numbers from a to b excluding b, incrementing by 1.

Syntax
```
for "variable" in range("start_number", "end_number"):
```

Note: Start Value must be less than end Value..!!

Example:
```
range(5,10)
```

Example:
```
for a in range(2,7):
        print(a)
```

Example:

```
snumber=int(input("Enter Any Start Number: "))
enumber=int(input("Enter Any End Number: "))
for data in range(snumber,enumber):
    print(data)
```

range(a,b,c) : Generates a sequence of numbers from a to b excluding b, incrementing by c.

Syntax

```
for "variable" in range("start_number", "end_number",increment):
```

Note:
Start Value must be less than end value, If it is increment..!!
Start Value must be bigger than end value, If it is decrement...!!

Example:

```
range(0,10,2)
```

Example:

```
for a in range(2,19,5):
        print(a)
```

Example:

```
snumber=int(input("Enter Any Start Number: "))
enumber=int(input("Enter Any End Number: "))
incre=int(input("Enter Any increment Number: "))
for data in range(snumber,enumber,incre):
    print(data)
```

Example: Multiplication Table

```
n=int(input("Enter Any Number: "))
for num in range(1, 11):
    print(n, "X", num, "=", n * num)
```

for loop with else clause
A for loop can have an optional else block as well. The else part is executed if the items in the sequence used in for loop exhausts. break statement can be used to stop a for loop. In such case, the else part is ignored.

Syntax:

```
for <variable> in <sequence>:
        <statements>
        <statements>
else:
        <statements>
        <statements>
```

Example:

```
digits = [0, 1, 5, 7]
for i in digits:
    print(i)
else:
    print("No items left.")
```

Example:

```
for i in range(5):
    print(i)
else:
    print('Iterated over everything :)')
```

LOOPING CONTROL STATEMENTS/ TRANSFER STATEMENTS
A statement that alters the execution of a loop from its designated sequence is a loop control statement.

break
To break out from a loop, you can use the keyword "break".

Syntax

```
for variable_name in sequence :
   statement_1
   statement_2
   ......
   if expression:
   break
```

Example:

```
digi = [0, 1, 5, 7]
for data in digi:
    if data==4:
        break
    print(data)
else:
    print("Loop Success")
```

Example:

```
for i in range(5):
    if i==2:
        break
    print(i)
else:
    print('Iterated over everything :)')
```

NOTE:
if we stop the loop, say with a break statement, then the else suite will not be executed

Example:

```
for x in "PYTHON":
    if(x=='O'):
        break
    print(x)
else:
    print("Loop Completed")
```

Continue
The continue statement is used to tell Python to skip the rest of the statements in the current loop block and to continue to the next iteration of the loop.

Syntax:

```
continue;
```

Syntax
```
for variable_name in sequence :
   statement_1
   statement_2
   ......
   if expression:
   continue;
```

Example:
```
for i in range(1,10):
    if i == 3:
        continue
    print(i)
```

Example:
```
digi=[1,2,3,4,5]
for data in digi:
    if data==4:
        continue
    print(data)
else:
    print("Loop Success")
```

Python while Loop Statements:
while Loop is used to execute number of statements or body till the condition passed in while is true. Once the condition is false, the control will come out of the loop.

Syntax:
```
while <expression>:
       Body
```

Syntax
```
while (expression) :
    statement_1
    statement_2
```

Example:
```
x=1
while x<=5:
    print(x)
    x=x+1
print("loop Finished")
```

Example: Reverse Numbers
```
n=5
while(n>0):
    print(n)
    n-=1
print("Good Bye")
```

Example:
```
# Program to add natural # sum = 1+2+3+...+n
sum = 0
i = 1
while i <= 10:
    sum = sum + i
```

```
    i = i+1    # update counter
print("The sum is", sum)
```

Example:

```
#The following program uses a nested for loop to find the prime
numbers from 2 to 100
i = 2
while(i < 100):
   j = 2
   while(j <= (i/j)):
      if not(i%j): break
      j = j + 1
   if (j > i/j) : print (i," is prime")
   i = i + 1
print ("Good bye!")
```

Example:

```
x=int(input("Enter Any Number less than 10: "))
while(x<=10):
    print(x)
    x+=1
print("Loop OK")
```

NOTE: Any infinite loop hit Ctrl+C to Exit from infinite loop..!!

Python: while and else statement

There is a structural similarity between while and else statement. Both have a block of statement(s) which is only executed when the condition is true.

Syntax:

```
while (expression) :
    statement_1
    statement_2
    ......
else :
    statement_3
    statement_4
```

Example:

```
x=1
while x<=5:
    print(x)
    x=x+1
else:
    print("loop Finished")
```

Example:

```
a=10
while a>0:
    print("Value of a is",a)
    a=a-1
else:
    print("Loop is Completed")
```