Nested While loop

A while loop inside while loop is called nested while loop.

Syntax:

```
while <condition>:  ➔ Outer while loop
   while <condition>: ➔ Inner while loop
       statement-1
       statement-2
   statement-3
   statement-4
```

while loop execute statements until given condition is True, if condition is False, it stop executing while.

Example:

```
# Write a program to generate tables from 1 to 10
# Using nested while

num=1
while num<=10:
    i=1
    while i<=10:
        p=num*i
        print(f'{num}x{i}={p}')
        i=i+1

    num=num+1
    input()
```

Output

```
1x1=1
1x2=2
1x3=3
1x4=4
1x5=5
1x6=6
1x7=7
1x8=8
```

```
1x9=9
1x10=10

2x1=2
2x2=4
2x3=6
2x4=8
2x5=10
2x6=12
2x7=14
2x8=16
2x9=18
2x10=20
….
```

Example:

```
# Write a program to generate armstrong numbers
# from 100-999

num=100
while num<=999:
    num1=num
    s=0
    while num1>0:
        d=num1%10
        s=s+(d**3)
        num1=num1//10
    if num==s:
        print(num)
    num=num+1
```

Output

```
153
370
371
407
```

Example:

```
i=1
while i<=10:
  j=1
  while j<=10:
     print(f'{i*j:4d}',end='')
     j=j+1
  print()
  i=i+1
```

Output

```
   1   2   3   4   5   6   7   8   9  10
   2   4   6   8  10  12  14  16  18  20
   3   6   9  12  15  18  21  24  27  30
   4   8  12  16  20  24  28  32  36  40
   5  10  15  20  25  30  35  40  45  50
   6  12  18  24  30  36  42  48  54  60
   7  14  21  28  35  42  49  56  63  70
   8  16  24  32  40  48  56  64  72  80
   9  18  27  36  45  54  63  72  81  90
  10  20  30  40  50  60  70  80  90 100
```

Branching statements

Branching statements are used to control the execution of while loop and for loop. Python support 2 branching statements

1. break
2. continue

break

“break” is a keyword, which represents branching statement in python.
This statement is used inside while loop or for loop.
This statement is used to terminate execution of while or for loop.

Example:

```
while True:
    print("Hello")
    break

for i in range(5):
    print("Bye")
    break
```

Output

```
Hello
Bye
```

Example:

```
while True:
    user=input("UserName :")
    pwd=input("Password :")
    if user=="nit" and pwd=="n123":
        print("Welcome to My Application")
        break
    else:
        print("Invalid UserName or Password")
```

Output

```
UserName :nit
Password :abc
Invalid UserName or Password
UserName :xyz
Password :n123
Invalid UserName or Password
UserName :nit
Password :n123
Welcome to My Application
```

Example:

```
# Write a program to find input numbers is prime number
# or not
num=int(input("Enter any number "))
c=0
for i in range(1,num+1):
    if num%i==0:
        c=c+1
    if c>2:
        break

if c==2:
    print(f'{num} is prime')
else:
    print(f'{num} is not prime')
```

Output

```
Enter any number 4
4 is not prime

Enter any number 5
5 is prime
```

continue

“continue” is a keyword in python
“continue” keyword represents branching statement in python.
This statement is opposite of “break” statement.
This statement is used only inside while or for loop
This statement is used to continue the execution of while or for loop

Example:

```
for i in range(5):
    if i%2==0:
        continue
    print("Hello")

num=1
while num<=10:
    if num%2!=0:
        num=num+1
        continue
    print(num)
    num=num+1
```

Output

```
Hello
Hello
2
4
6
8
10
```

While..else and for..else

Python support special syntax of using while with else and for with else. Always else block is executed after execution of while loop or for loop.

Syntax: while ..else	**Syntax: for..else**

while <condition>:
   statement-1
   statement-2
else:
   statement-3
   statement-4
	for variable in iterable:
	    statement-1
	    statement-2
	else:
	    statement-3
	    statement-4

“else” block is not executed if while or for loop are terminated using break statement.

Example:

```
for i in range(5):
   print("Inside for loop")
else:
   print("Inside else block")


i=1
while i<=5:
   print("Inside while loop")
   i=i+1
else:
   print("Inside else block")

for i in range(5):
   print("Inside for loop")
   break
else:
   print("Inside else block")


i=1
while i<=5:
   print("Inside while")
   break
else:
    print("Inside else block")
```

Output

```
Inside for loop
Inside for loop
Inside for loop
Inside for loop
Inside for loop
Inside else block
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside else block
Inside for loop
Inside while
```

Collection Data types (OR) Data Structures

Python data types are classified into 2 categories

1. Scalar Data types (5)
    a. Int
    b. Float
    c. Complex
    d. Boolean
    e. NoneType
2. Collection Data types (9)
    a. Sequential
        i. List
        ii. Tuple
        iii. Range
        iv. String
        v. Bytes
        vi. bytearray
    b. Sets
        i. Set
        ii. frozenset
    c. Mapping
        i. Dictionary