Python Programming - IV

Looping & It's Control

CONTENTS

LOOPING

WHILE LOOp

A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true. A uniform indent is mandatory for rest of the statements to be in a loop.

Example: Printing Factorial of a number

```
num = int(input("Enter a number to calc it's factorial:"))
temp=1
fact=1

while temp <= num:
    fact*=temp
    temp=temp+1

print("Factorial of", num, "is :",fact)
```

WAP: To Print first 10 Odd numbers by correctig foll. code

```
n=1
while n<=10:
    if(n%2!=0):
        print(">", n)
    n=n+1
else:
    print("Bye")
```

FOR Loop

The for statement in Python has the ability to iterate over the items of any sequence, such as a list or a string. Unlike the traditional FOR loop of C / C++ / Java, Python uses different structure than others.

Example: Printing first 10 numbers of Fibonacci series

```
a=0; b=1
fib=1

for i in range(10):
    print(fib)
    fib = a+b
    a=b
    b=fib
```

WAP: To Print a the Star Pattern using single for loop as shown below:

```
*
**
***
****
*****
```

```
s=""
for i in range(5):
    s+= "*"
    print(s)
else:
    print("Goodbye")
```

NESTED LOOp

Python programming language allows the use of one loop inside another loop. The following section shows a few examples to illustrate the concept.

FOR LOOP

```
for iterating_val in sequence:
      for iterating_val in sequence:
            statement(s)
      statement(s)
```

WHILE LOOP

```
while expression:
      while expression:
            statement(s)
      statement(s)
```

Example: Printing Multiplication tables from 1 to 10

```
for i in range(1,11):
    for j in range(1,11):
        k=i*j
        print(k, end="\t")
    print('\n')
```

Example: Printing Embroidery Star Pattern

```
n = 4
while n>0:
    for i in range(4):
        x="*\t" *n
        print(x)
    n-=1
else:
    exit(0)
```

BREAK

The break statement is used for premature termination of the current loop. After abandoning the loop, execution at the next statement is resumed, just like the traditional break statement in C.

The most common use of break is when some external condition is triggered requiring a hasty exit from a loop. The break statement can be used in both while and for loops.

If you are using nested loops, the break statement stops the execution of the innermost loop and starts executing the next line of the code after the block.

SYNTAX:

```
break
```

EXAMPLE: Using break statement

```
for i in range(10):
    print(i)
    if(i==5):
        break
else:
    print(i)
```

CONTINUE

The continue statement in Python returns the control to the beginning of the current loop.
When encountered, the loop starts next iteration without executing the remaining statements in the current iteration.

SYNTAX:

```
continue
```

EXAMPLE: Using Continue statement

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

Affects the statements written after continue statement

EXAMPLE: BREAK in FOR - Mul. Tables

```
for i in range(1,11):
    for j in range(1,11):
        k=i*j
        print(k, end="\t")
        if i%3==0:
            break

    print()
```

PASS

It is used when a statement is required syntactically but you do not want any command or code to execute.

The pass statement is a null operation; **nothing happens** when it executes.

The pass statement is also useful in places where your code will eventually go, but has not been written yet i.e. in stubs).

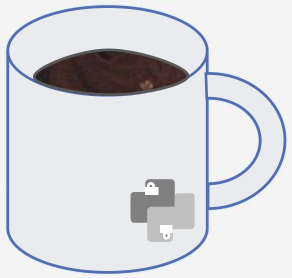
SYNTAX:

```
pass
```

EXAMPLE: Using Pass in order to replace a Statement which is undecided

```
num=1
if num<0:
   print("Number is Negative")
elif num>0:
   pass
else:
   print("Number is Zero")
```

CODE
COFFE
REPEAT



SEE YOU AFTER A COFFE BREAK