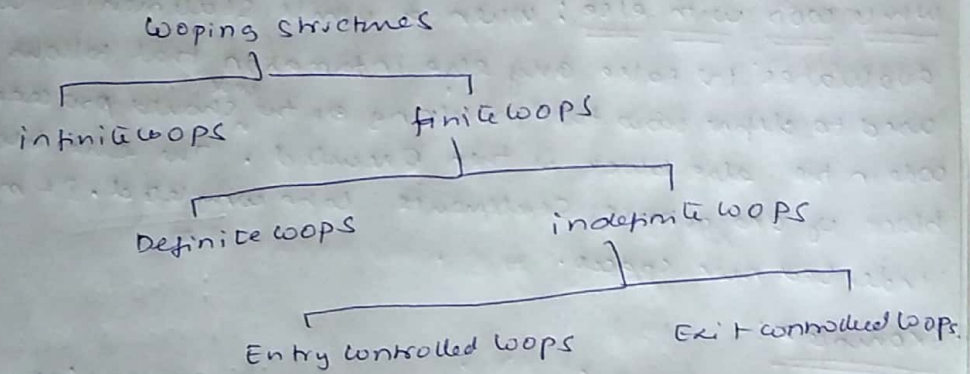
18/4/24 Looping Statements

iteration and repetition structures, which are frequently used in sequence, selection and repetation, makes use of Programming constructs like while, for and nested loop.

Benefits of Loops

① Loop aid in escalating the code reusability
② Loops simplify complex problems into simple ones, which are very easy to handle
③ Loops aid in easily crisscrossing the elements placed in arrays and linked list
④ Loops aid the developers to remove the redundancy of physical writing, and implement the upfront and small instructions repetitively.



There are Six primary loops or flow controls

1. If
2. for
3. while
4. break
5. continue
6. pass

There are four different functions

1. User defined function
2. Lambda function
3. recursive function
4. Built-in function - There are totally 68 built in function in looping

Loop Statements

While loop: It is a condition-controlled loop that supports repeated execution of a statement or block of statement controlled by a conditional expression.

Scanned by CamScanner

Example1:

```
>>> exercise = 1
>>> while exercise <=6 :
... Print ('exercise =', exercise)
... exercise += 1
```

Output
exercise = 1
exercise = 2
exercise = 3
exercise = 4
exercise = 5
exercise = 6

Condition in this case is <= 6 lesser than = 6 so 1 to 6 is printed

- It is a entry check loop. This looping occurs till the condition fails, in this case, the loop terminates when the value stored in the variable exercise is become 7

While loop with else: When a while statement's condition evaluates to false, any else information that follows it can be used to define how the processing of the code should proceed. The code in the else block is then executed. The else block is a block of one or more statements to be completed. It must be indented more spaces.

Example2:

```
>>> i = 1  # declare and initialize the variable i = 1
>>> # while loop begins
>>> while i <=6 :
...     # print value of "i"
...     Print ("while loop executed {i} number of times")
...     i = i+1 # increments the variable "i" by one
... else: # else executed when the condition (i<=6) is
          false
...     Print ('Else is executed")
```

Output: while loop executed 1 number of times
- do - 2 - do -
- do - 3 - do -
- do - 4 - do -
- do - 5 - do -
- do - 6 - do -

ELSE part is executed.

Scanned by CamScanner

FOR LOOP: for loop is a single-line statement that is easy to use. It ensures the sequential execution of all the stipulated conditions. It is used to execute a block of code repeatedly at predetermined times. It is used in combo with iterable object or sequence type like list, tuple, set, range etc.

Example3:

```
n = 1894
sum = 0
for counter in range (1, n+1):
    sum = sum + counter
print ("sum of 1 until %d: %d" %(n, sum))
```

Output: Sum of 1 until 1894: 1794565

for_else loop: The else keyword is useful in for loop in Python. The body of the loop is followed by an else block. In case the iteration fails, the else block's statements will be run.

Example 4:

```
for x in range(1,7)
    print ("for loop condition is true. value of 'x' now is", x)
else
    print ("for loop is over, so print else body")
```

Output:

for loop condition is true. Value of 'x' now is 1
- do - 2
- do - 3
- do - 4
- do - 5
- do - 6

for loop is over, so print else body.

NESTED LOOP: When a loop lies within or inside the body of another loop, it is called a nested loop. The loop lying within another loop is called the inner loop and the loop outer to the inner one is called the outer loop.

Scanned by CamScanner

Examples:

```
n = range(1,4)
for j in n:
Print(" Factorial of " + str(j) + " is: ")
mul = 1
for i in range (1, j+1)
    mul = mul * i:
Print(mul)
```

Output

```
Factorial of 1 is:
1
Factorial of 2 is
2
Factorial of 3 is
3
```

Example 6:

```
for x in range(5):
    y = 4
    while y >= x:
      Print ("#@ Need Break @#")
      y = 1
      Print (" ")
      break.
```

Output:

```
#@ Need break @#
- do -
- do -
- do -
- do -
```

Jump Statements: There may be times during the execution of the code when a pause or termination of the code is necessary due to the occurance of a recursion statement (It is also known as an abnormal loop termination) ~~due to the occurence of a recursion statement~~ TO instantaneously exit the body of a loop ~~or reject~~ the condition from the inside of the loop these control statements are used. For this purpose Break, continue and pass statements are used.

Break statement: The break statement offered by a python is a unique purpose statement. An immediate termination of the loop

Scanned by CamScanner

occurs when the break statement is executed. but any lines of code that are reachable after the loop body are ~~skipp~~ skipped over.

Example7:

```
x = 0
while True:
x += 1
if x % 2 == 0
    print(x)
if x >=12
    break
Print ("outside of while body")
```

Output: loop prints value = 2
-do- = 4
-do- = 6
-do- = 8
-do- = 10
-do- = 12
Outside of while body.

Continue statement when a continue statement is bumbed into an inner loop, the control navigates to the start of loop and the start of the next iteration. instead of implementing the statements of the contemporary iteration. The continue statement helps neglect a specific condition and continue with the execution.

Pass statement: pass is a null statement. The interpreter does not ignore a pass statement, but nothing happens and the statement results into no operation. The pass statement is useful when you don't write the implementation of a function but you want to implement it in the future.

Scanned by CamScanner