Looping Constructs in Python

Printing Numbers

- Write a Python code to print the first N natural numbers.

PAC - Printing Numbers

Input	Processing	Output
A Number, N	Start printing from 1 and increment each time to print the next value and continue upto N	Numbers from 1 to N

Algorithm - Printing first 'N' Natural Numbers

Step 1: Start

Step 2 : Read a number, N → Known

Step 3 : Initialize counter as 1 → Known

Step 4 : print counter → Known

Step 5 : Increment the counter by 1 → Known

Step 6: repeat Step 4 until the counter reaches N → ??

Step 7: Stop

Yet to learn

- Repeatedly execute a set of statements

Class Average

- Given marks secured in CSE1001 by the students in a class, design an algorithm and write a Python code to determine the class average. Print only two decimal digits in average

Class Average

Input	Processing	Output
Number of students in class, mark scored by each student	Determine total of marks secured by students Find average of marks	Class average of marks

Average marks scored by ‘N’ number of Students

Step 1: Start

Step 2 : Read Number Of Students

Step 3 : Initialize counter as 0

Step 4 : Input mark

Step 5 : Add the mark with total

Step 6 : Increment the counter by 1

Step 7: repeat Step 4 to Step 6 until counter less than number of students

Step 7: Divide the total by number of students and store it in average

Step 8: Display the average

Step 9: Stop

Test Cases

Input

5

90 85 70 50 60

Output

71.00

Processing Involved

Already Know

- To read values from user
- To check if a condition is satisfied
- Print characters

Yet to learn

- Repeatedly execute a set of statements

Need of iterative control

Repeated execution of set of statements

- An **iterative control statement** is a control statement providing repeated execution of a set of instructions
- Because of their repeated execution, iterative control structures are commonly referred to as “loops.”

Iterative Control Statements in Python

- ***while*** statement
  - Repeatedly executes a set of statements based on a Boolean expression (condition).
  - Ideal when stop criteria is not explicit

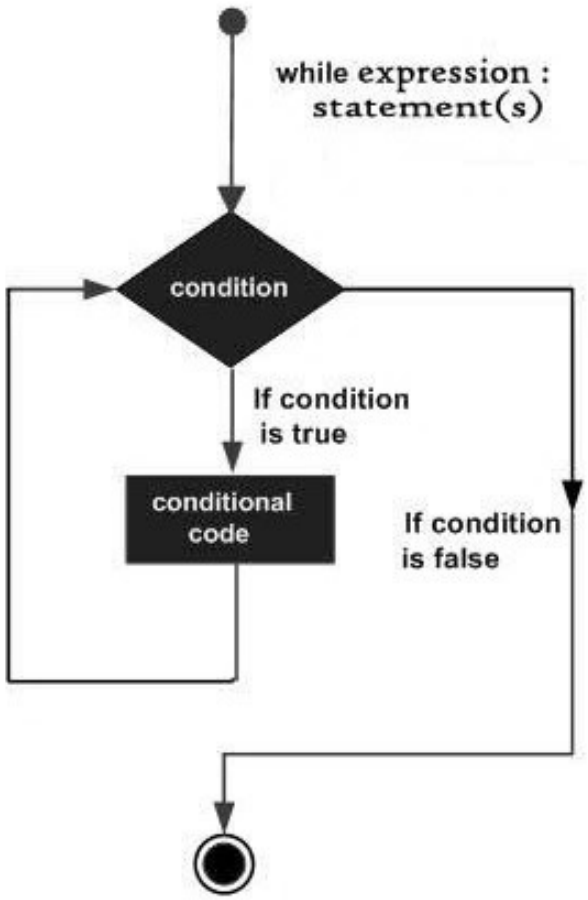
- ***for*** statement
  - Repeatedly executes a set of statements until the sequence is exhausted

Syntax of ***while*** in Python

```
while test:          # Loop test (condition is true)
    statements       # Loop body
else:                # Optional else
    statements
```

Run if didn't exit loop with break

Note the colon (:) following test and else and the indentation



Example: Printing first 'N' numbers

```
Let N=3
counter =1
while  counter  <= N:
    print(counter)
    counter = counter + 1
```

Iteration	counter	counter<=3	Print counter	counter=counter+1
1	1	True	1	counter=1+1 ->(2)
2	2	True	2	counter=2+1 ->(3)
3	3	True	3	counter=3+1 ->(4)
4	4	False	loop termination	

Print values from 1 to 9 in a line

```
a=1; b=10
while a < b:            # Another way to code counter loops
        print(a, end=' ')
        a += 1                  # Or, a = a + 1
```

Output:

1 2 3 4 5 6 7 8 9

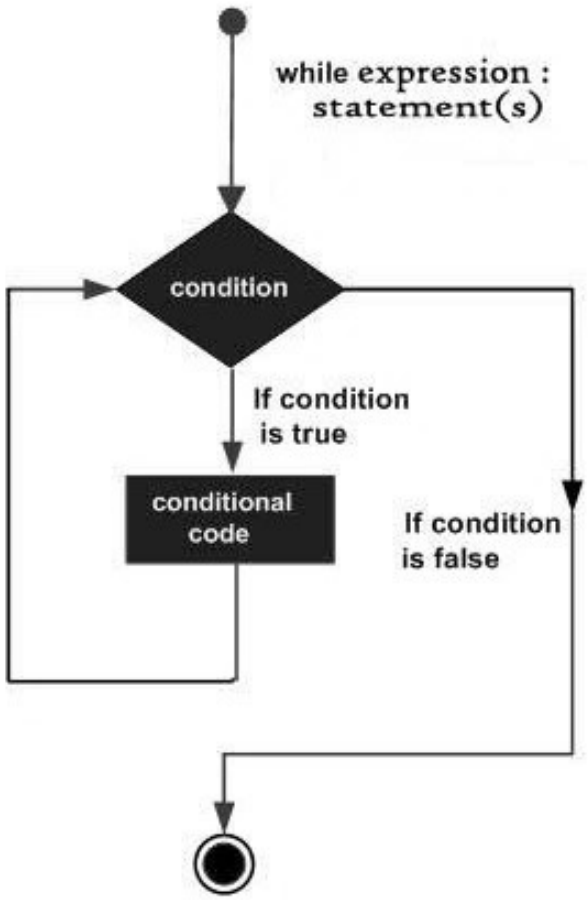
Include end=‘ ‘ in print statement to suppress default move to new line

Try it

- Write a python program to print the first N natural numbers in the reverse order.

- Write a python program to print only even numbers upto a given number N.

- Write a python program to find the sum of first ‘N’ Natural numbers.



Example: Sum of first 'N' numbers

```
sum =0
counter =1
N=3
while  counter  <= N:
    sum=sum + counter
    counter = counter + 1
print(sum)
```

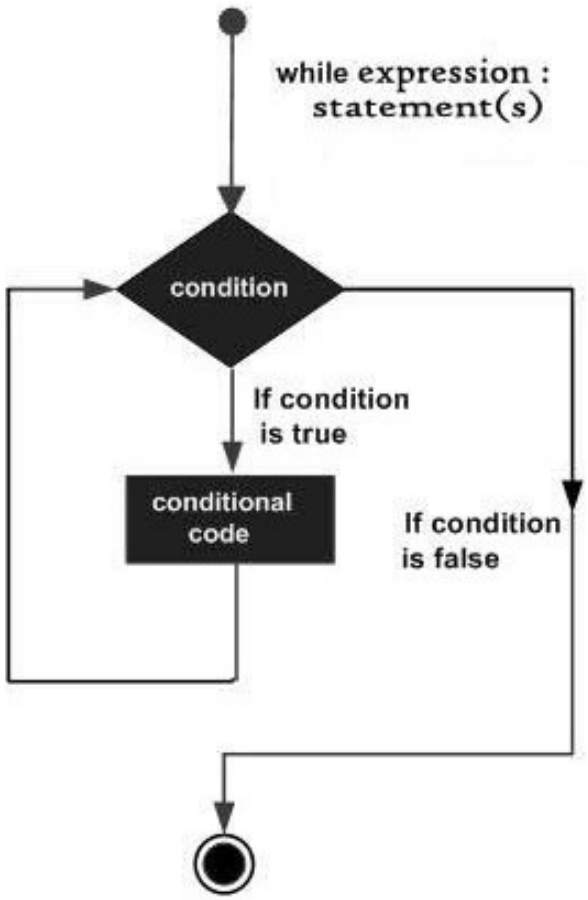
Iteration	sum	counter	counter<=3	sum=sum+counter	counter=counter+1
1	0	1	True	sum=0+1 ->(1)	counter=1+1 ->(2)
2	1	2	True	sum=1+2 ->(3)	counter=2+1 ->(3)
3	3	3	True	sum=3+3 ->(6)	counter=3+1 ->(4)
4	4	4	False	loop termination	

While statement

- Repeatedly executes a set of statements based on a provided Boolean expression (condition).
- All iterative control needed in a program can be achieved by use of the while statement.

Syntax of While in Python

```
while test:                    # Loop test
    statements                 # Loop body
else:                          # Optional else
    statements
# Run if didn't exit loop with break
```



Example use

Sum of first 'n' numbers

```
sum =0
current =1
n=3
while current <= n:
    sum=sum + current
    current = current + 1
```

Iteration	sum	current	current <= 3	sum = sum + current	current = current + 1
1	0	1	True	sum = 0 + 1 (1)	current = 1 + 1 (2)
2	1	2	True	sum = 1 + 2 (3)	current = 2 + 1 (3)
3	3	3	True	sum = 3 + 3 (6)	current = 3 + 1 (4)
4	6	4	False	loop termination	

Print values from 0 to 9 in a line

```
a=0; b=10
while a < b:            # One way to code counter loops
        print(a, end=' ')
        a += 1                  # Or, a = a + 1
```

Output:

0 1 2 3 4 5 6 7 8 9

Include end=‘ ‘ in print statement to suppress default move to new line

Break, continue, pass, and the Loop else

- ***break***
  - Jumps out of the closest enclosing loop
- ***continue***
  - Jumps to the top of the closest enclosing loop
- ***pass***
  - Does nothing at all: it's an empty statement placeholder
- ***Loop else***
  - Runs if and only if the loop is exited normally (i.e., without hitting a break)

break statement – Example

- Read and print the name and age of a person until 'stop' is read.
  - There is no fixed number of iteration as we do not know when stop would be entered by the user
  - break statement is used to jump out of the loop when 'stop' is read

break statement

```
while True:
        name = input('Enter name:')
        if name == 'stop':
                break
         age  = input('Enter age: ')
        print('Hello', name, '=>', age)
```

Output:

```
Enter name:bob
Enter age: 40
Hello bob => 1600
```

Try it

- Write a python code to find the sum of integers read until 0 is read.

pass statement

- Infinite loop
- while True: pass #empty placeholder

  # Type Ctrl-C to stop me!

continue statement

- Jumps to the top of the loop (condition) and skips the current iteration
- Example:
  - Print only the even numbers between 1 to N

```
Let N=3
counter =1
while  counter  <= N:
        if N%2!=0:
            continue        #skips the current iteration
        print(counter)
        counter = counter + 1
```

Print all even numbers less than 10 and greater than or equal to 0

```
x = 10
while x:
        x = x-1                          # Or, x -= 1
        if x % 2 != 0: continue          # Odd? -- skip print
        print(x, end=' ')
```

```python
y = int(input())
if not isinstance(y,int):
    print("Prime number check can be done only for integers")
else:
    if y==0:
        print("Zero is neither prime nor composite")
    elif y<0:
        print("Prime is checked only for positive integer")
    else:
        x = y // 2
        while x > 1:
            if y % x == 0:
                break
            x -= 1
        else:
            print(y, 'is prime')
```

Class Average

```
File Edit Format Run Options Window Help
count =0
total = 0
n=int(input('enter how many mark you want to read: '))
while count < n:
    mark=int(input('enter mark :'))
    if mark<0:
            print ("mark should be greater than 0, terminates.
            break
    total = total + mark
    count = count + 1
else:
    average=total/n
    print("average mark is" , format(average,"0.2f"))
```

Pattern Generation

- Your teacher has given you the task to draw the structure of a staircase. Being an expert programmer, you decided to make a program for the same. You are given the height of the staircase. Given the height of the staircase, write a program to print a staircase as shown in the example. For example, Staircase of height 6:

```
#
##
###
####
#####
######
```

Boundary Conditions: height >0

Pattern Generation

Input	Processing	Output
Staircase height	Create steps one by one To create a step print character equal to length of step	Pattern

Pseudocode

```
READ staircase_height
if staircase_height > 0
x = 1
Repeat
y = 1
Repeat
    print #
    y = y + 1
Until y <= x
x = x + 1
Until x <= staircase_height
End if
Else
Print "Invalid input"
```

Test Cases

Input

3

Output

```
#
# #
# # #
```

Processing Involved

Print step by step

Test Cases

Input

-1

Output

Invalid input

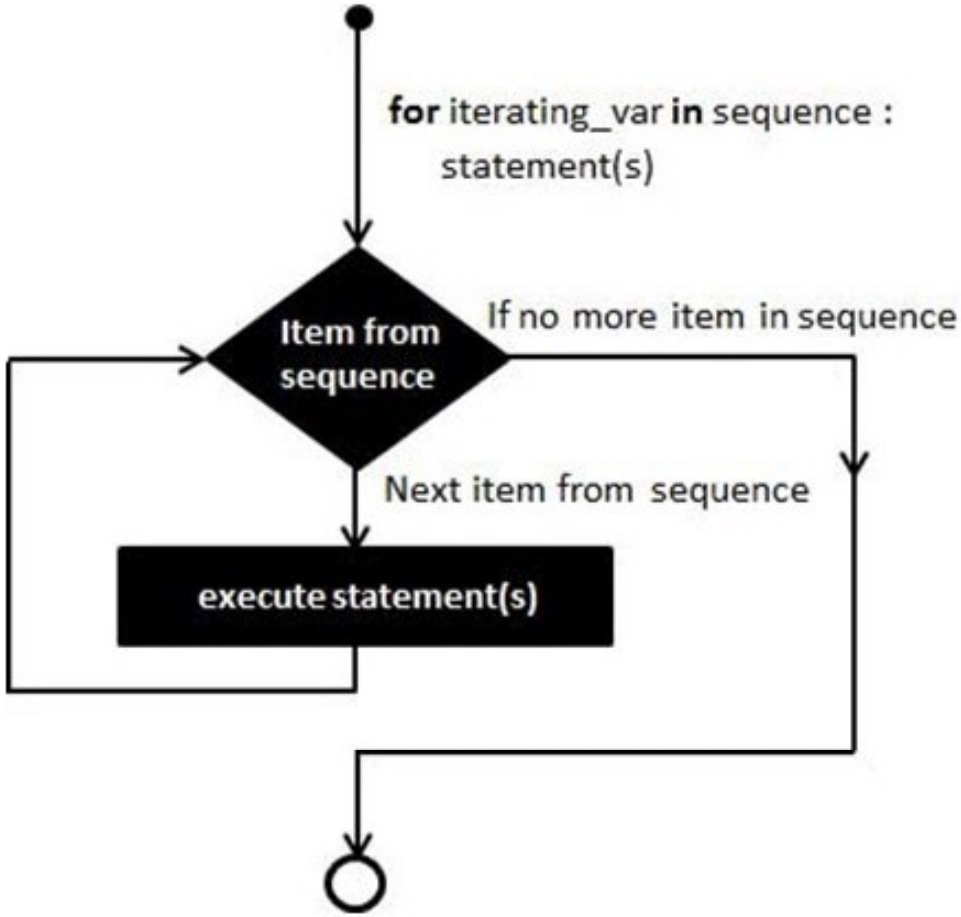
Processing Involved

Boundary condition check fails

For iteration

- In while loop, we cannot predict how many times the loop will repeat
- The number of iterations depends on the input or until the conditional expression remains true
- While loop is ideal when stop criteria is not explicit

Control flow of for statement



Syntax of for Statement

```
for target in object:
    # Assign object items to target
    statements
    if test: break          # Exit loop now, skip else
    if test: continue       # Go to top of loop now
else:   statements          # If we didn't hit a
    'break'
```

For and Strings

for iterating_var in sequence or range:

    statement(s)

Example:

```
for letter in 'Python':
   print 'Current Letter :', letter
```

For and Strings

When the above code is executed:

```
Current Letter : P
 Current Letter : y
 Current Letter : t
 Current Letter : h
 Current Letter : o
 Current Letter : n
```

For and Range

```
for n in range(1, 6):
    print(n)
```

When the above code is executed:

```
1
2
3
4
5
```

range function call

Syntax - range(begin,end,step)

where

Begin - first value in the range; if omitted, then default value is 0

end - one past the last value in the range; end value may not be omitted

Step - amount to increment or decrement; if this parameter is omitted, it defaults to 1 and counts up by ones

begin, end, and step must all be integer values;

floating-point values and other types are not allowed

Example for Range

range(10) → 0,1,2,3,4,5,6,7,8,9

range(1, 10) → 1,2,3,4,5,6,7,8,9

range(1, 10, 2) → 1,3,5,7,9

range(10, 0, -1) → 10,9,8,7,6,5,4,3,2,1

range(10, 0, -2) → 10,8,6,4,2

range(2, 11, 2) → 2,4,6,8,10

range(-5, 5) → −5,−4,−3,−2,−1,0,1,2,3,4

range(1, 2) → 1

range(1, 1) → (empty)

range(1, -1) → (empty)

range(1, -1, -1) → 1,0

range(0) → (empty)

Print Even Numbers Using Range

```
>>> for i in range(2,10,2):
        print(i)
```

Output:

2

4

6

8

```
print("Enter number of steps")
n = int(input())
for i in range(0,n):
    for j in range(0,i+1):
        print('#',end = ' ')
    print()
```

Try it

- Print even numbers from N to 1
- Print only odd numbers from 1 to N

Pattern Generation - Code

```
print("Enter number of steps")
n = int(input())
for i in range(0,n):
    for j in range(0,i+1):
        print('#',end = ' ')
    print()
```

Test Cases

Input

-1

Output

Invalid input

Processing Involved

Boundary condition check fails

Test Cases

Input

3

Output

```
#
# #
# # #
```

Processing Involved

Print step by step

Try it

- Write pseudocode and python code to print the pattern structure given the value of n.
  - For example, the following structure should be displayed for n=4.

```
####
###
##
#
```

Try it

- Write pseudocode and python code to print the pattern structure given the value of n.
  - For example, the following structure should be displayed for n=4.

1

2 2

3 3 3

4 4 4 4