while loop

while loop

A while loop allows general repetition based upon the repeated testing of a Boolean condition

The syntax for a while loop in Python is as follows:

```
while condition:
    body
```

: Colon Must

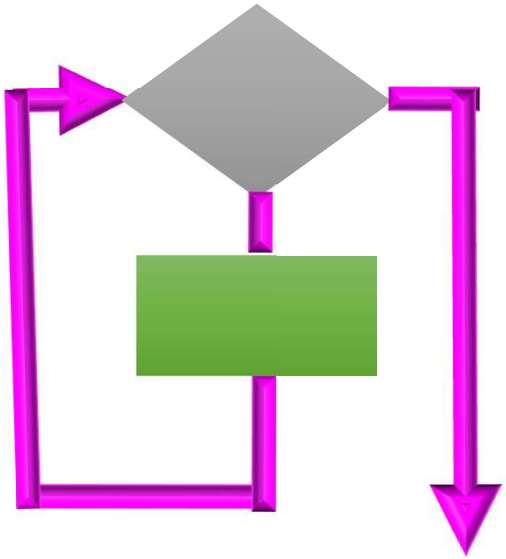
Where, loop body contain the single statement or set of statements (compound statement) or an empty statement.

Contd..

while loop

The loop iterates while the expression evaluates to true, when expression becomes false the loop terminates.

FLOW CHART



while loop

while loop – Programming example

while loop - programs

Natural Numbers generation

*Python 3.4.0: nat_while.py - C:/Python34/nat_wh...

File Edit Format Run Options Windows Help

```
#Program to generate natural nos
def gen_nat_no_while():
    i=1
    n=int(input("Enter the limit : "))
    while(i<=n):
        print(i)
        i+=1
gen_nat_no_while()
```

Ln: 11 Col: 0

OUTPUT

Python 3.4.0 S...

File Edit Shell Debug Options Windows Help

```
Enter the limit : 5
1
2
3
4
5
```

Ln: 26 Col: 4

while loop - programs

Calculating Sum of Natural Numbers

*Python 3.4.0: while_llop.py - C:/Python34/w...

File Edit Format Run Options Windows Help

```
#sum of Natural numbers
def while_loop_example():
    sum1 = 0
    count = 1
    while (count < 10):
        sum1 = sum1 + count
        count = count + 1
    print (count) # should be 10
    print (sum1) # should be 45
while_loop_example()
```

Ln: 12 Col: 0

OUTPUT

Python 3.4.0 ...

File Edit Shell Debug Options Windows Help

```
10
45
```

Ln: 17 Col: 4

while loop - programs

#Generating Fibonacci numbers

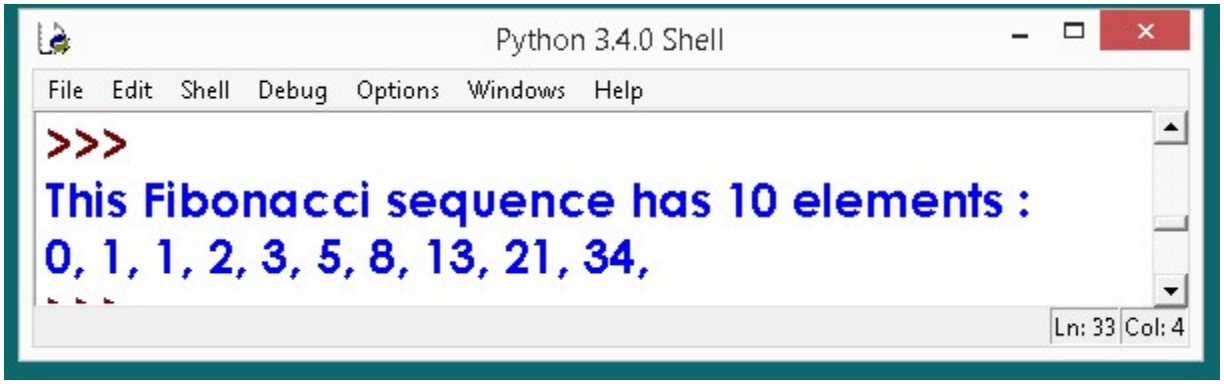
File Edit Format Run Options Windows Help

```
def fibo_numbers():
    length = 10
# The first two values
    x = 0
    y = 1
    iteration = 0
# Condition to check if the length has a valid input
    if length <= 0:
        print("Please provide a number greater than zero")
    elif length == 1:
        print("This Fibonacci sequence has {} element".format(length), ":")
        print(x)
    else:
        print("This Fibonacci sequence has {} elements".format(length), ":")
    while (iteration < length):
        print(x, end=', ')
        z = x + y
        # Modify values
        x = y
        y = z
        iteration += 1
fibo_numbers()
```

while loop - programs

#Generating Fibonacci numbers

OUTPUT



4. BRANCHING OR JUMPING STATEMENTS

4. BRANCHING OR JUMPING STATEMENTS

Python has an unconditional branching statements and they are,

1. break STATEMENT

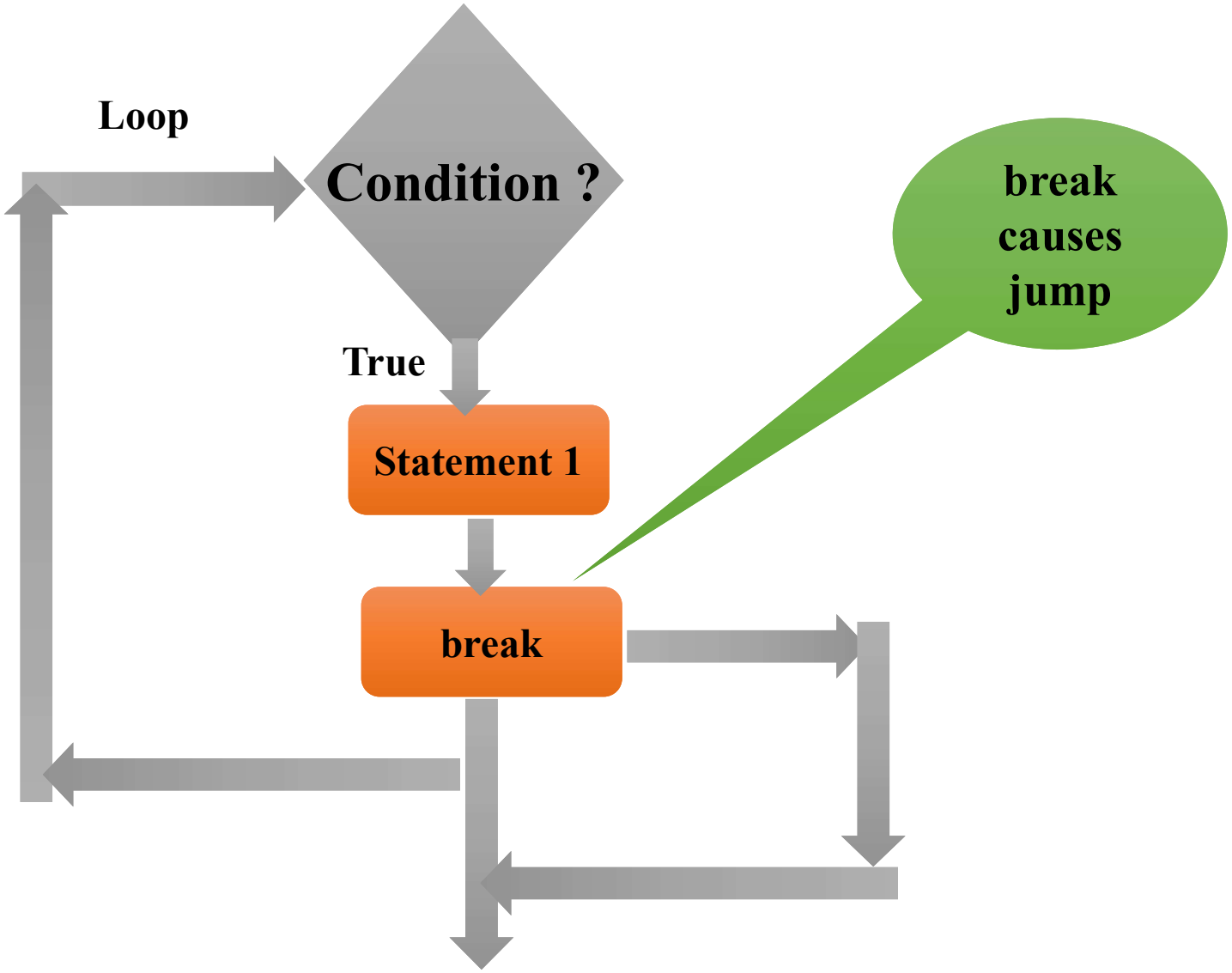
2. continue STATEMENT

4. BRANCHING OR JUMPING STATEMENTS

1. break STATEMENT

Break can be used to unconditionally jump out of the loop. It terminates the execution of the loop. Break can be used in while loop and for loop. Break is mostly required, when because of some external condition, we need to exit from a loop.

1. break STATEMENT



1. break STATEMENT

Python 3.4.0: breakexample.py - C:\Python34\bre...

File Edit Format Run Options Windows Help

```
def break_example():
    y=5
    for i in range(0,y+1):
        if i == y:
            print("Thank you!")
            break
        else:
            print(i)
    print("End of Prg")
break_example()
```

Ln: 11 Col: 0

OUT PUT

Python ...

File Edit Shell Debug Options Windows Help

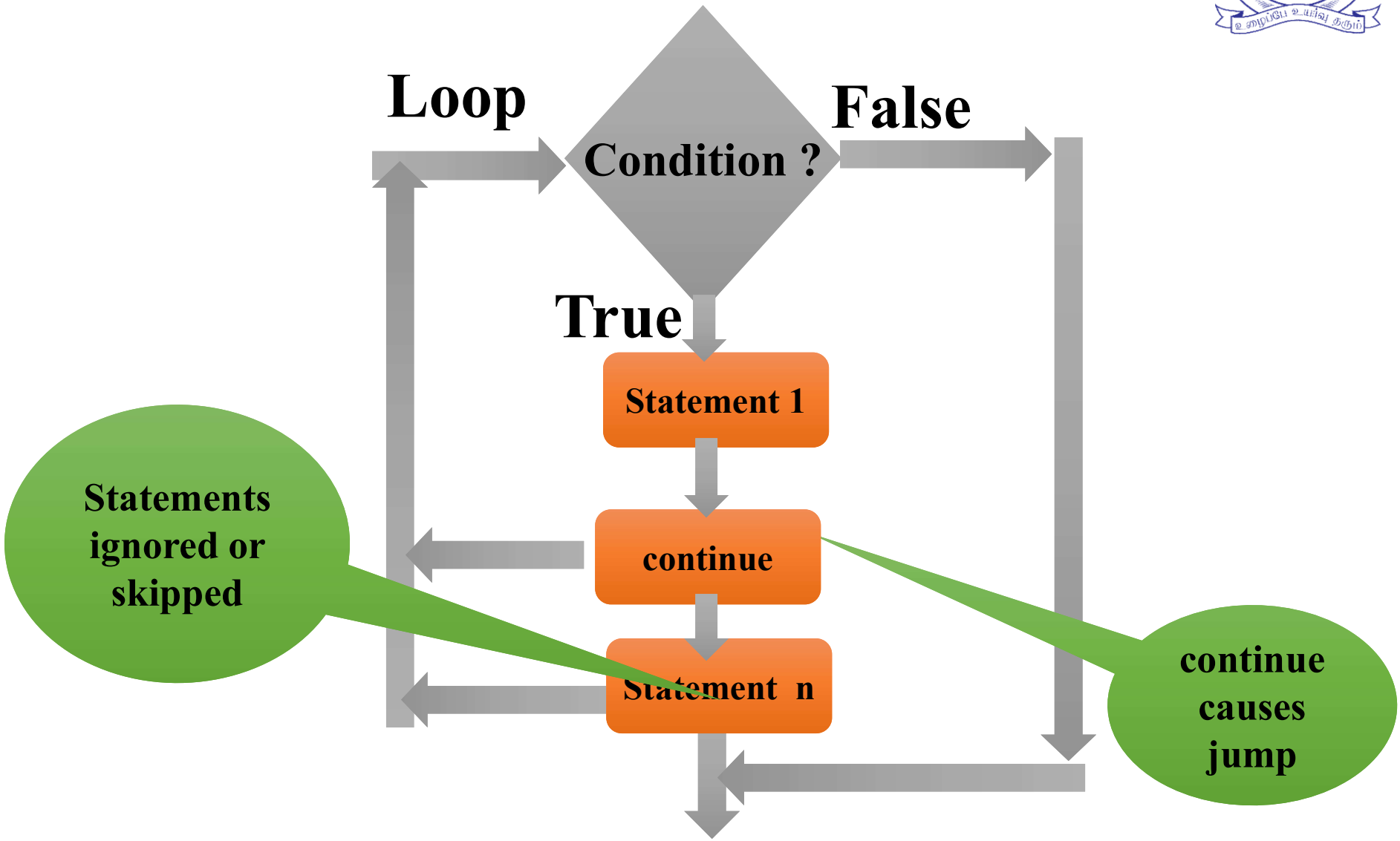
```
>>>
0
1
2
3
4
Thank you!
End of Prg
>>>
```

Ln: 12 Col: 4

2. continue STATEMENT

The continue statement in Python returns the control to the beginning of the while loop. The continue statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop. The continue statement can be used in both while and for loops.

2. continue STATEMENT



2. continue STATEMENT

Python 3.4.0: continueexa.py - C:/Pyth...

File Edit Format Run Options Windows Help

```
def continue_in_loop():
    for i in range(0,5):
        if i==2:
            continue
        print (i)

continue_in_loop()
```

Ln: 8 Col: 0

File Edit Shell Debug Options Windows Help

```
>>>
0
1
3
4
>>>
```

Ln: 9 Col: 4

when i value becomes 2 the print statement gets skipped, continue statement goes for next iteration, hence in the out put 2 is not printed

pass STATEMENT

pass STATEMENT

The pass statement in Python 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 is also useful in places where your code will eventually go, but has not been written yet (e.g., in stubs for example):

pass STATEMENT

Python 3.4.0: passexampl...

File Edit Format Run Options Windows Help

```
def pass_example():
    for i in range(0,10):
        pass
    print("Good Bye!")
pass_example()
```

Ln: 6 Col: 0

pass in loop

Python 3.4.0 ...

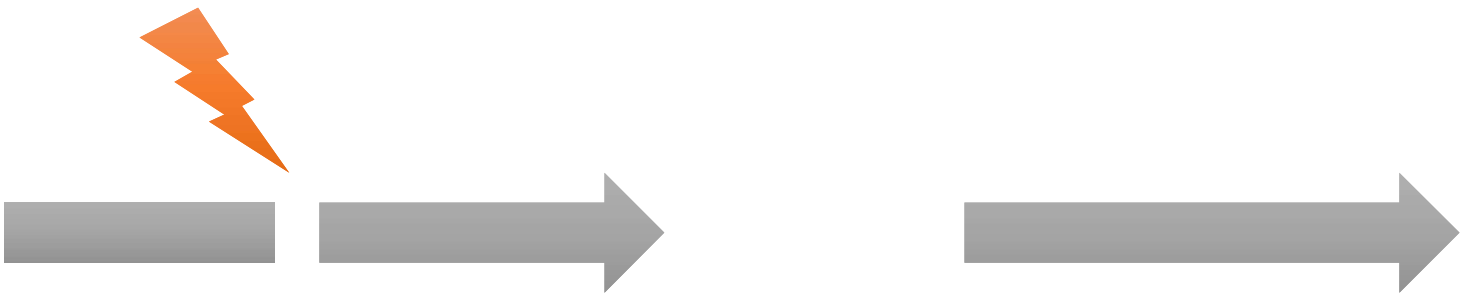
File Edit Shell Debug Options Windows Help

```
>>>
Good Bye!
>>>
>>>
```

Ln: 7 Col: 4

pass in loop has no output

Difference Between break and continue



break **continue**

Difference Between break and continue

BREAK	CONTINUE
It terminates the execution of remaining iteration of the loop.	It terminates only the current iteration of the loop.
'break' resumes the control of the program to the end of loop enclosing that 'break'.	'continue' resumes the control of the program to the next iteration of that loop enclosing 'continue'.
It causes early termination of loop.	It causes early execution of the next iteration.
'break' stops the continuation of loop.	'continue' do not stops the continuation of loop, it only stops the current iteration.

Thank You