Loops

Darin Brezeale

The University of Texas at Arlington

Basic Concepts – Loops

Python has the following loop constructs:

- `while`
- `for`

Basic Concepts – Loops

Loops allow us to repeat a task. We need some way to determine when the loop should terminate. This could be

- when some condition has been met
- after a predetermined number of iterations

`while` loop

The basic form of the `while` loop is

```
while test:
    do_something
```

As long as `test` is true, the loop will repeat.

Example: If we begin summing the positive integers starting at 1, how many will it take to get a sum of at least 100?

Output

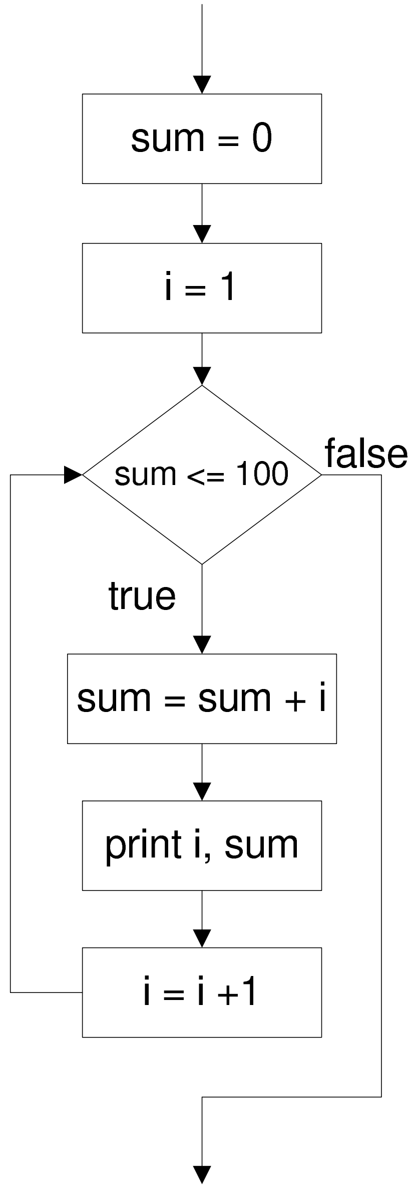
If we did it by hand, we would determine the following:

```
1 1
2 3
3 6
4 10
5 15
6 21
7 28
8 36
9 45
10 55
11 66
12 78
13 91
14 105
```

`while` loop flow chart

```
sum = 0
i = 1

while sum <= 100:
    sum = sum + i
    print i, sum
    i = i + 1
```



Basic Concepts – Loops

When the loop will terminate after a predetermined number of iterations, we need:

- a counting variable
- a test of that variable
- to increment/decrement that variable

`while` loop

Example: What is the sum of the integers from 1 to 5?

```
sum = 0
i = 1

while i < 6:
    sum = sum + i
    i = i + 1

print "the sum from 1 to 5 is", sum
```

`for` loop

The `for` loop has the following form:

```
for element_of_object in object:
    do_something
```

`for` loop example

We could have done the last example with a `for` loop.

```
sum = 0

for i in range(1, 6):
    sum = sum + i

print "the sum from 1 to 5 is", sum
```

`range(a, b)` is a function that generates the integers `a` to `b-1` (in this example).

Changing loop behavior

Sometimes we want to end a loop early or move on to the next value. We have two ways of doing this:

1. `continue` – jump to the very end of the current loop
2. `break` – get out of the current loop completely

`continue` Statement

```
i = 0

while i < 10:
    i = i + 1
    if i == 5:
        continue  # skip 5
    print i
```

produces

```
1
2
3
4
6
7
8
9
10
```

`continue` Statement cont.

In the previous example, we could have avoided the `continue` statement:

```
i = 0

while i < 10:
    i = i + 1
    if i != 5:
        print i
```

`break` Statement

Keep going until a negative number is provided:

```
while True :
    value = input("Enter a number (negative to stop):  ")
    if value < 0 :
        break
    print value

print "All done"
```

`True` is always true, so this creates an infinite loop. Since we have replaced the test condition with something that can never evaluate as false, the `break` is the way of eventually stopping the loop.

`break` Statement cont.

Output from previous program:

```
Enter a number (negative to stop):  3453
3453
Enter a number (negative to stop):  0
0
Enter a number (negative to stop):  12
12
Enter a number (negative to stop):  -5
All done
```