Introduction To Python

Dr.Hajialiasgari

Tehran University
Of
Medical Science

November 2, 2024

TEHRAN UNIVERSITY
OF
MEDICAL SCIENCES

1 Loops and Iteration

Repeated Steps

Loops (repeated steps) have iteration variables that change each time through a loop. Often these iteration variables go through a sequence of numbers.

An Infinite Loop

```
n = 5
while n > 0 :
    print('Lather')
    print('Rinse')
print('Dry off!')
```

What is wrong with this loop?

Breaking Out of the Loop (Code)

The `break` statement ends the current loop and jumps to the statement immediately following the loop.

```
while True:
    line = input('> ')
    if line == 'done' :
        break
    print(line)
print('Done!')
```

Breaking Out of the Loop (Output)

Output:

```
> hello there
hello there
> finished
finished
> done
Done!
```

Finishing an Iteration with continue (Code)

The `continue` statement ends the current iteration and jumps to the top of the loop to start the next iteration.

```
while True:
    line = input('> ')
    if line[0] == '#' :
        continue
    if line == 'done' :
        break
    print(line)
print('Done!')
```

Finishing an Iteration with continue (Output)

Output:

```
> hello there
hello there
> # don't print this
> print this!
print this!
> done
Done!
```

Definite Loops

Iterating over a set of items

- Loops that iterate over a finite set of things are called definite loops.
- Example:

```
1 for i in range(1,10) :
2     print(i)
3 print('Your first loop!')
```

Finding the Average in a Loop (Code)

```
count = 0
sum = 0
print('Before', count, sum)
for value in [9, 41, 12, 3, 74, 15] :
    count = count + 1
    sum = sum + value
    print(count, sum, value)
```

Filtering in a Loop (Code)

```
print('Before')
for value in [9, 41, 12, 3, 74, 15] :
    if value > 20:
        print('Large number', value)
print('After')
```

We use an if statement in the loop to catch/filter the values we are looking for.

Finding the Smallest Value (Code)

```
1 smallest = None
2 for value in [9, 41, 12, 3, 74, 15] :
3     if smallest is None :
4         smallest = value
5     elif value < smallest :
6         smallest = value
7     print(smallest, value)
8 print(smallest)
```

We still have a variable that is the smallest so far. The first time through the loop, smallest is `None`, so we take the first value to be the smallest.

The is and is not Operators

```
1 smallest = None
2 for value in [3, 41, 12, 9, 74, 15] :
3     if smallest is None :
4         smallest = value
5     elif value < smallest :
6         smallest = value
7     print(smallest, value)
```

Python has an `is` operator that can be used in logical expressions.
Implies is the same as
Similar to, but stronger than, ==.
`is not` also is a logical operator.

End of Loops and Iteration