`for` loops

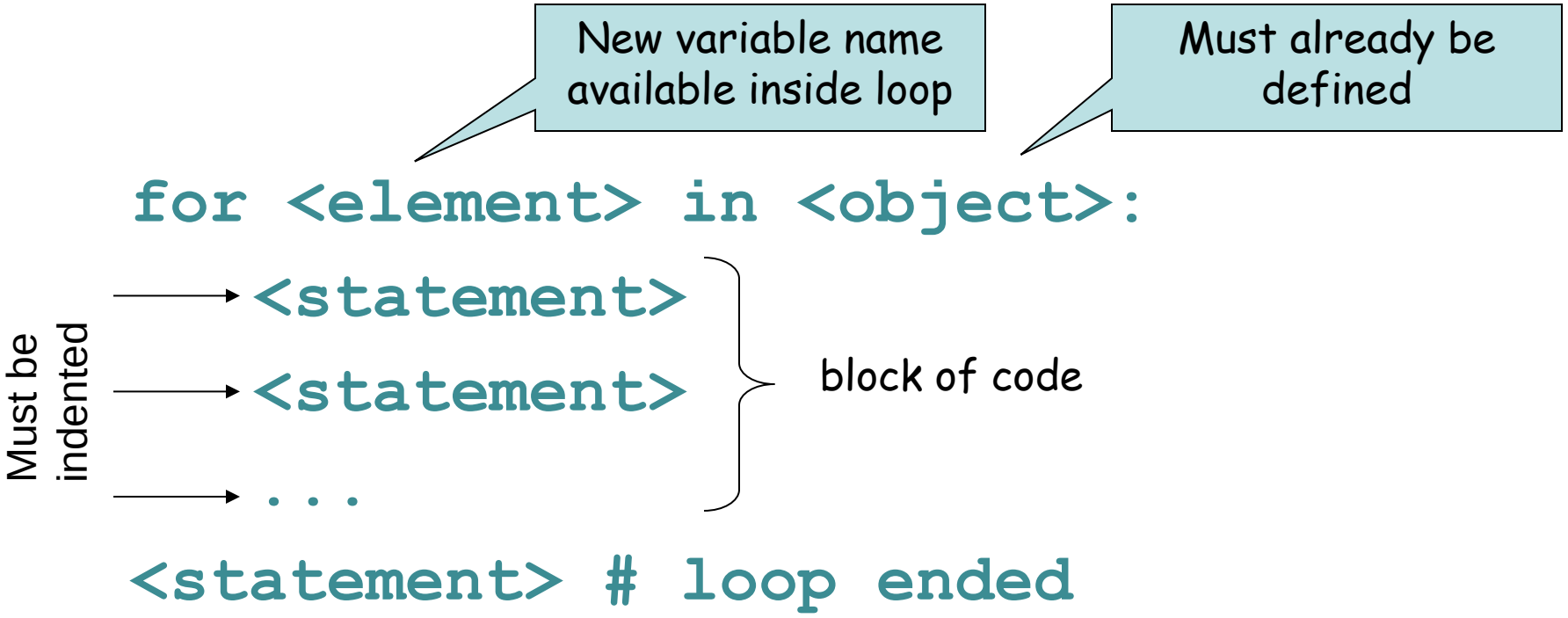
Genome 373

Genomic Informatics

Elhanan Borenstein

`for` loop

- Allows you to perform an operation on each element in a list (or character in a string).



Try it …

```
>>> for name in ["Andrew", "Teboho", "Xian"]:
...     print "Hello", name
...
Hello Andrew
Hello Teboho
Hello Xian
>>>
```

Another example

- Reminder - each line in block must have the same indentation.

```
>>> for integer in [0, 1, 2]:
...     print integer
...     print integer * integer
...
0
0
1
1
2
4
```

Looping on a string

```
>>> DNA = 'AGTCGA'
>>> for base in DNA:
...     print "base =", base
...
base = A
base = G
base = T
base = C
base = G
base = A
>>>
```

Indexing

- If needed, use an integer variable to keep track of a numeric index during looping.

```
>>> index = 0   # initialize index
>>> for base in DNA:
...    index = index + 1  # increment index
...    print "base", index, "is", base
...
base 1 is A
base 2 is G
base 3 is T
base 4 is C
base 5 is G
base 6 is A
>>> print "The sequence has", index, "bases"
The sequence has 6 bases
>>>
```

the increment operation is so common there is a shorthand: **`index += 1`**

The `range()` function

- The `range()` function returns a list of integers covering a specified range.

```
range([start,] stop [,step])
```

[optional arguments], default to 0 and 1

```
>>>range(5)
[0, 1, 2, 3, 4]
>>>range(2,8)
[2, 3, 4, 5, 6, 7]
>>> range(-1, 2)
[-1, 0, 1]
```

```
>>> range(0, 8, 2)
[0, 2, 4, 6]
>>> range(0, 8, 3)
[0, 3, 6]
>>> range(6, 0, -1)
[6, 5, 4, 3, 2, 1]
```

Using `range()` in a `for` loop

```
>>> for index in range(0,4):
...     print index, "squared is", index * index
...
0 squared is 0
1 squared is 1
2 squared is 4
3 squared is 9
```

`range()` produces a list of integers (so this is really just like looping over a list)

Nested loops

```
>>> matrix = [[0.5, 1.3], [1.7, -3.4], [2.4, 5.4]]
>>> for row in range(0, 3):
...     print "row =", row
...     for column in range(0, 2):
...         print matrix[row][column]
...
row = 0
0.5
1.3
row = 1
1.7
-3.4
row = 2
2.4
5.4
>>>
```

Terminating a loop

- **break** jumps out of the closest enclosing loop

```
>>> for index in range(0,3):
...     if (index == 2):
...         break
...     print index
...
0
1
```

Terminating a loop

- `continue` jumps to the top of the closest enclosing loop

```
>>> for index in range(0, 3):
...     if (index == 1):
...         continue
...     print index
...
0
2
```

Summary

```
for <element> in <object>:
   <block>
```

Perform `<block>` for each element in `<object>`.

```
range(<start>, <stop>, <increment>)
```

Define a list of numbers. `<start>` and `<increment>` are optional, default to 0 and 1. Increment can be negative (go backwards with start > stop)

`break` – jump out of a loop

`continue` – jump to the top of the loop

`while` loop

Similar to a `for` loop

```
while (conditional test):
      <statement1>
      <statement2>
      . . .
      <last statement>
```

While something is `True` keep running the loop, exit as soon as the test is `False`. The conditional test syntax is the same as for `if` and `elif` statements.

What does this program do?

```
sum = 0
count = 1
while (count < 10):
    sum = sum + count
    count = count + 1
print count                 # should be 10
print sum                   # should be 45
```

`for` vs. `while`

- you will probably use `for` loops more

- `for` is natural to loop through a list, characters in a string, etc. (anything of determinate size).

- `while` is natural to loop an indeterminate number of times until some condition is met.

Examples of `for` loops

```
for base in sequence:
    <do something with each base>

for sequence in database:
    <do something with each sequence>

for index in range(5,200):
    <do something with each index>
```

Examples of `while` loops

```
while (error > 0.05):
     <do something that will reduce error>
```

```
while (score > 0):
     <traceback through a DP matrix, each
     time setting the current score>
```

You now know everything you need to know to write quite complex programs.

There's lots more to learn, but you could now (for example) write a sequence alignment program.