Ling 555 — Programming for Linguists

Python — Conditionals and Loops

Robert Albert Felty

Speech Research Laboratory
Indiana University

Oct. 1, 2008

Outline

1. Homework questions
2. Miscellaneous tips from chapter 5
3. Conditionals
   - Blocks and indenting
   - Truth values
   - if
   - booleans
4. Loops
   - While
   - For
   - Break and continue
   - More on iteration

Something completely different!

Pop Quiz

1. Create three variables, *foo,bar,foobar*, with the values `1,500, "you"` in one line (1 point)
2. Create three variables *x,y,z* all with the value `0` with as few keystrokes as possible (1 point)
3. Print out all six variables you just created in as few keystrokes as possible (2 points)

Blocks and indenting

Definition

In python, blocks are created by the use of a colon, followed by an indented section of text

Example

```
if foo==bar:
    do something
    do another thing
    a final thing
do this regardless
```

Truth values

Definition

Nothing is false. That is:

`False`

`None`

`0`

`[]` (empty list)

`{}` (empty dict)

`''` (empty string)

`()` (empty tuple)

if

Practice

```
# Initialize two variables x,y to 3,4
# if x and y are equal to each other,
# print "equal", otherwise "unequal"
```

if

Practice

```
# Initialize two variables x,y to 3,4
# if x and y are equal to each other,
# print "equal", otherwise "unequal"
if x ==y:
    print "equal"
else:
    print "unequal"
```

if

Practice

```
# Initialize two variables x,y to 3,4
# if x and y are equal to each other,
# print "equal", otherwise "unequal"
if x ==y:
    print "equal"
else:
    print "unequal"

# Now add a condition that if
#x is divisible by y, print "divisible"
```

if

Practice

```
# Now add a condition that if
#x is divisible by y, print "divisible"
if x ==y:
    print "equal"
elif x % y == 0:
    print "divisible"
else:
    print "unequal"
```

Booleans

Definition

You can combine conditions with *and* and *or*, and negate with *not*

Example

```
if 5 < x < 10 and x not in y:
    print "x is between 5 and 10, \
        and is not in the list y"
```

While

Beware of infinite loops!

It is easy to create infinite loops with `while`. Make sure that your while condition will return false at some point.

practice

1. Create a list *mylist* from 1:10
2. Choose a random item from this list until the item you choose is your lucky number (your choice of number from 1 to 10 (inclusive)). Count how many times did your program have to choose a number? (Use the random.choice function)

While

practice

1. Create a list *mylist* from 1:10
2. Choose a random item from this list until the item you choose is your lucky number (your choice of number from 1 to 10 (inclusive)). Count how many times did your program have to choose a number? (Use the random.choice function)

```
import random
mycount=0
myList=range(1,11)
myNumber=3
while random.choice(myList) != myNumber:
    mycount+=1
print "it took", mycount, "times"
```

for

Definition

Iterate: to do (something) over again or repeatedly.
It easy to iterate over lists with for loops

practice

1. Multiply each item in *mylist* by 2, and print the result (don't change the values of mylist)
2. Multiply each item in *mylist* by 2, (*do* change the values of mylist)

for

Definition

Iterate: to do (something) over again or repeatedly.
It easy to iterate over lists with for loops

practice

1. Multiply each item in *mylist* by 2, and print the result (don't change the values of mylist)
   `for item in mylist:  print (item*2)`
2. Multiply each item in *mylist* by 2, (*do* change the values of mylist)

for

Definition

Iterate: to do (something) over again or repeatedly.
It easy to iterate over lists with for loops

practice

1. Multiply each item in *mylist* by 2, and print the result (don't change the values of mylist)
   ```
   for item in mylist:  print (item*2)
   ```
2. Multiply each item in *mylist* by 2, (*do* change the values of mylist)
   ```
   mylist=[x*2 for x in mylist]
   ```

Break and Continue

Definition

break stops execution of the current loop
continue skips the rest of one iteration of the current loop

practice

Loop over the *mylist* list until you find a number divisible by 3

Parallel iteration

Definition

Sometimes you want to iterate over two lists simultaneously `for name,age in zip(names,ages):`

Numbered iteration

Definition

Sometimes when iterating over a list, you want the indices and the values.

```
for index, string in enumerate(strings):
```

OR

```
for index in range(len(strings)):
```

practice

Print the index and the value for all the items in *mylist*