Lecture Agenda

• Last lecture:
  - `bool`
  - `if`
  - Conditionals
• This lecture:
  - Iteration

Iteration

One final, essential tool to make the computer do something over and over again.

• Repeat something forever
• Repeat something until something happens
• Repeat something a fixed number of times
  - move Mario forward 10 pixels
  - print 7 copies
  - play Nyan Cat 15 times

Iteration: while Loops I

A conditional loop.

• The general idea is that we continue repeating a block of code as long as a given condition holds
• Basic form:
  ```python
  while <condition>:
      statement block
  ```
• We use the notion of “block” as in `if` statements, but here, potentially the code block is repeated
  ```python
text = ''
while len(text) != 3:
text = input('Enter 3-digit code: ')
if len(text) == 3:
    break
print('Sorry, invalid code. ')
  ```

Iteration: while Loops II

• Another way to end `while` loops (and bypass the condition in the `while` statement) is via a `break` in the block of code
  ```python
text = ''
while True:
    text = input('Enter 3-digit code: ')
    if len(text) == 3:
        break
print('Sorry, invalid code. ')
  ```

This prematurely and unconditionally exits from the loop

Iteration: for Loops I

A loop over sequences.

• The general idea is that we work our way through (all of) an “iterable” (e.g. `str`, `tuple`) of items one item at a time, in sequence
• Basic form:
  ```python
  for <var> in <iterable>:
      statement block
  ```
• Note: `in` here is not (quite) the same as the comparison operator of the same name
• It is called for...in... because of the `\forall` quantifier
Iteration: for Loops II

• Simple example:

```python
sum = 0
for i in "abc":
    sum = sum + ord(i)
print(sum)
```

is equivalent to:

```python
sum = 0
sum = sum + ord('a')
sum = sum + ord('b')
sum = sum + ord('c')
print(sum)
```

Iteration: for Loops III

• More interesting example:

```python
vowels = 0
for char in "rhythm":
    if char in "aeiou":
        vowels = vowels + 1
print(vowels)
```

A Useful Function for creating a sequence

• `range(start=0,end,end,step=1)`: generate a sequence of `int` values from `start` (inclusive) to `end` (non-inclusive), counting `step` at a time

```python
>>> for i in range(5):
...     print(i, end=" ")
0 1 2 3 4
>>> for i in range(0,10,2):
...     print(i, end=" ")
0 2 4 6 8
>>> for i in range(10,0,-1):
...     print(i, end=" ")
10 9 8 7 6 5 4 3 2 1
```

Choosing between for and while

• If you need to iterate over all items of an iterable, use a `for` loop
• If there is a well defined end-point to the iteration which doesn’t involve iterating over all items, use a `while` loop
• With a `for` loop, avoid modifying the object you are iterating over within the block of code
• Given a choice between the two, `for` loops are generally more elegant/safer/easier to understand

for Loop Practice: Mexican Wave

• Given the string `wave` made up of a "Y" and `width-1` repeats of "x", how can we use a `for` loop to move the "Y" across one position to the right at a time?

Class Exercise

• Assuming an unlimited number of coins of each of the following denominations: (1, 2, 5, 10, 20) calculate the number of distinct coin combinations which make up a given amount `N` (in cents).
Lecture Summary

• What is an “iteration”, and what are the two basic forms of iteration in python?
• What is the basic syntax of a for loop, and what function is commonly used to iterate a fixed number of times?
• What is the basic syntax of the while loop, and what is the role of break?
• When should you use for vs. while?