Week 6

`while` loops; file I/O; introduction to lists

Special thanks to Scott Shawcroft, Ryan Tucker, and Paul Beck for their work on these slides.

while Loops

```
while test:
    statements
```

sentinel.py

```
# Sums integers entered by the user
# until -1 is entered, using a sentinel loop.
sum = 0
number = input("Enter a number (-1 to quit)? ")?

while number != -1:
    sum += number
    number = input("Enter a number (-1 to quit)? ")?

print "The total is", sum
```

python™

Random Numbers

`from random import *`

`randint(`**min**`, `**max**`)`

- returns a random integer in range [**min**, **max**] inclusive

`choice(`**sequence**`)`

- returns a randomly chosen value from the given sequence
- (the sequence can be a range, a string, an array, ...)

```
>>> from random import *
>>> randint(1, 5)
2
>>> randint(1, 5)
5
>>> choice(range(4, 20, 2))
16
>>> choice("hello")
'e'
```

Reading Files

name = `open("`**filename**`")`

– opens the given file for reading, and returns a file object

name`.read()`	-	file's entire contents as a string
name`.readline()`	-	next line from file as a string
name`.readlines()`	-	file's contents as a list of lines

– the lines from a file object can also be read using a `for` loop

```
>>> f = open("hours.txt")
>>> f.read()
'123 Susan 12.5 8.1 7.6 3.2\n
456 Brad 4.0 11.6 6.5 2.7 12\n
789 Jenn 8.0 8.0 8.0 8.0 7.5\n'
```

python™

File Input Template

- A template for reading files in Python:

```
name = open("filename")
for line in name:
    statements
```

```
>>> input = open("hours.txt")
>>> for line in input:
...     print line.strip()   # strip() removes \n

123 Susan 12.5 8.1 7.6 3.2
456 Brad 4.0 11.6 6.5 2.7 12
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

Exercise

- Write a function `input_stats` that accepts a file name as a parameter and that reports the longest line in the file.
  - example input file, `carroll.txt`:

```
Beware the Jabberwock, my son,
the jaws that bite, the claws that catch,
Beware the JubJub bird and shun
the frumious bandersnatch.
```

  - expected output:

```
>>> input_stats("carroll.txt")
longest line = 42 characters
the jaws that bite, the claws that catch,
```

Exercise Solution

```
def input_stats(filename):
    input = open(filename)
    longest = ""
    for line in input:
        if len(line) > len(longest):
            longest = line

    print "Longest line =", len(longest)
    print longest
```

Lists

- **list**: Python's equivalent to Java's array (but cooler)
  - Declaring:

    name = [**value**, **value**, **...**, **value**]   or,

    name = [**value**] * **length**

  - Accessing/modifying elements: (same as Java)

    name[**index**] = **value**

```
>>> scores = [9, 14, 18, 19, 16]
[9, 14, 18, 19, 16]
>>> counts = [0] * 4
[0, 0, 0, 0]
>>> scores[0] + scores[4]
25
```

Indexing

- Lists can be indexed using positive or negative numbers:

```
>>> scores = [9, 14, 12, 19, 16, 18, 24, 15]
[9, 14, 12, 19, 16, 18, 24, 15]
>>> scores[3]
19
>>> scores[-3]
18
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
value	9	14	12	19	16	18	24	15
index	*-8*	*-7*	*-6*	*-5*	*-4*	*-3*	*-2*	*-1*

Slicing

- **slice**: A sub-list created by specifying start/end indexes

```
name[start:end]          # end is exclusive
name[start:]             # to end of list
name[:end]               # from start of list
name[start:end:step]     # every step'th value
```

```
>>> scores = [9, 14, 12, 19, 16, 18, 24, 15]
>>> scores[2:5]
[12, 19, 16]
>>> scores[3:]
[19, 16, 18, 24, 15]
>>> scores[:3]
[9, 14, 12]
>>> scores[-3:]
[18, 24, 15]
```

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
value	9	14	12	19	16	18	24	15
index	*-8*	*-7*	*-6*	*-5*	*-4*	*-3*	*-2*	*-1*

Other List Abilities

- Lists can be printed (or converted to string with `str()`).
- Find out a list's length by passing it to the `len` function.
- Loop over the elements of a list using a `for` ... `in` loop.

```
>>> scores = [9, 14, 18, 19]
>>> print "My scores are", scores
My scores are [9, 14, 18, 19]
>>> len(scores)
4
>>> total = 0
>>> for score in scores:
...     print "next score:", score
...     total += score
assignment score: 9
assignment score: 14
assignment score: 18
assignment score: 19
>>> total
60
```

Ranges, Strings, and Lists

- The `range` function returns a list.

```
>>> nums = range(5)
>>> nums
[0, 1, 2, 3, 4]
>>> nums[-2:]
[3, 4]
>>> len(nums)
5
```

- Strings behave like lists of characters:
  - `len`
  - indexing and slicing
  - `for` ... `in` loops

String Splitting

- `split` breaks a string into a list of tokens.

  name`.split()` `# break by whitespace`

  name`.split(`**delimiter**`)` `# break by delimiter`

- `join` performs the opposite of a `split`

  delimiter`.join(`**list**`)`

```
>>> name = "Brave Sir Robin"
>>> name[-5:]
'Robin'
>>> tokens = name.split()
['Brave', 'Sir', 'Robin']
>>> name.split("r")
['B', 'ave Si', ' Robin']
>>> "||".join(tokens)
'Brave||Sir||Robin'
```

Tokenizing File Input

- Use `split` to tokenize line contents when reading files.
  - You may want to type-cast tokens: **type**(**value**)

```
>>> f = open("example.txt")
>>> line = f.readline()
>>> line
'hello world 42 3.14\n'

>>> tokens = line.split()
>>> tokens
['hello', 'world', '42', '3.14']

>>> word = tokens[0]
'hello'
>>> answer = int(tokens[2])
42
>>> pi = float(tokens[3])
3.14
```

Exercise

- Recall the `hours.txt` data:

```
123 Susan 12.5 8.1 7.6 3.2
456 Brad 4.0 11.6 6.5 2.7 12
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

- Recreate the `Hours` program from lecture in Python:

```
Susan (ID#123) worked 31.4 hours (7.85 / day)
Brad (ID#456) worked 36.8 hours (7.36 / day)
Jenn (ID#789) worked 39.5 hours (7.9 / day)
```

Exercise Answer

hours.py

```python
file = open("hours.txt")
for line in file:
    tokens = line.split()
    id = tokens[0]
    name = tokens[1]

    # cumulative sum of this employee's hours
    hours = 0.0
    days = 0
    for token in tokens[2:]:
        hours += float(token)
        days += 1

    print name, "(ID#" + str(id) + ") worked", \
          hours, "hours (" + str(hours / days), "/ day)"
```

Exercise

- Recall the Internet Movie Database (IMDb) data:

```
1 9.1 196376 The Shawshank Redemption (1994)
2 9.0 139085 The Godfather: Part II (1974)
3 8.8 81507 Casablanca (1942)
```

- Recreate the `Movies` program from lecture in Python:

```
Search word? part

Rank    Votes   Rating  Title
2       139085  9.0     The Godfather: Part II (1974)
40      129172  8.5     The Departed (2006)
95      20401   8.2     The Apartment (1960)
192     30587   8.0     Spartacus (1960)
4 matches.
```

Exercise Answer

movies.py

```
search_word = raw_input("Search word? ")
matches = 0

file = open("imdb.txt")
for line in file:
    tokens = line.split()
    rank = int(tokens[0])
    rating = float(tokens[1])
    votes = int(tokens[2])
    title = " ".join(tokens[3:])

    # does title contain searchWord?
    if search_word.lower() in title.lower():
        matches += 1
        print rank, "\t", votes, "\t", rating, "\t", title

print matches, "matches."
```

Writing Files

name = open("**filename**", "w")
name = open("**filename**", "a")

- opens file for write (deletes any previous contents) , or
- opens file for append (new data is placed after previous data)

name.write(**str**)	- writes the given string to the file
name.close()	- closes file once writing is done

```
>>> out = open("output.txt", "w")
>>> out.write("Hello, world!\n")
>>> out.write("How are you?")
>>> out.close()

>>> open("output.txt").read()
'Hello, world!\nHow are you?'
```

python™