ECS10

10/10

Overview
- Announcements / Homework 2
- ‘While’ loops
- Operators
- Functions

Announcements
- Homework 2
  - Should now be possible to submit homework 2
  - May use either nested ‘if’ statements or state variables
  - Ignore the ‘drop’ option for the homework assignment
- Assignment 3 should be up tonight
- Can’t add more than 198 students because of room requirements
- Check your position again; may have been added
- People will likely drop today, so keep checking

Loops: Repeating blocks
- Basic program will run through the program once
- ‘if’ statement changes what happens depending on the current program state, but still runs its block only once
- Loops repeat a block 0 or more times

‘While’ loops
- ‘while’ loops are one type of loop
- Repeats a block while some condition is True
- Once condition is False, block under while loop doesn’t run
  - May never run the block!

if/else statement

while loop
### 'While' loop dissected

- 'while' is a Python keyword which causes a block to repeat
- 'Boolean is True' is the condition. Replace with any Boolean expression or variable
- Used when you don’t know how many times you need to repeat

```python
while Boolean is True:
    do something
    ... 
    do something
condition.
Replace with any Boolean expression or variable.
```

### Side note: DON’T use break

- You may see the 'break' keyword being mentioned with loops
- DON’T use it for this course
- break ends the while loop early
- Can’t make any assumptions about the condition used by the while loop

```python
while x < 64:
    print x
    x = x * 2
# Boolean no longer True
print 'With x = ', x
print 'While loop done:',
print 'x >= 64,','with x = ', x
```

### Coin Flipping

- Last time, wrote program to flip a single coin
- Big topic in probability
- Is it possible to flip 10 heads in a row?
- How long do you have to flip before getting 3 heads total?
- Use 'while' loop to flip coins until three heads get flipped

```python
while x < 64:
    print x
    x = x * 2
# Boolean no longer True
print 'With x = ', x
```

```python
while x < 64:
    print x
    x = x * 2
# Boolean no longer True
print 'With x = ', x
```
**Coin flipping: In English**

Set number of heads = 0

While number of heads < 3:
  Flip a coin
  If the coin is heads:
    Print “Heads”
    Increment number of heads
  Else (the coin had to be tails):
    Print “Tails”

**Coin flipping: In Python**

Gain access to the random module

Set number of heads = 0

While number of heads < 3:
  Flip a coin
  If the coin is heads:
    Print “Heads”
    Increment head count
  Else (the coin had to be tails):
    Print “Tails”

```python
import random
numHeads = 0
while numHeads < 3:
    coin = random.randint(1,2)
    if coin == 1:
        print "Heads!
        numHeads = numHeads+1
    else: # expect coin == 2
        print "Tails!
```

**Coin flipping: As a flowchart**

**Coin flipping dissected**

**Getting valid input**

choosing = True
while choosing:
    input = raw_input("Choose h or t: ")
    if input == "h" or input == "t":
        choosing = False
“State” variable, redux

- choosing is a Boolean variable
- choosing == True while the user still has to choose ‘h’ or ‘t’
- Becomes False as soon as a valid choice is made
- It is a data item that never appears as input or output of the program; it is just there to keep track of how things are going internally.

Infinite loop

- One of the classic programming bugs
- Get out of it using CTRL-c (hold down control key and type c)
- Repeat after me: CTRL-c

```
while x < 5:
    print "I'm stuck!"
# Never reaches here
print 'I'm out!'```

Next assignment: compound interest program

- Original debt is $10,000
- While you still owe money, make a payment and pay interest
- Program ends when debt is paid off

Operators

- ‘Operators’ perform basic operations
- Typically needs 1 or 2 items of the same data type
- Usually used to combine items

<table>
<thead>
<tr>
<th>Data type</th>
<th>Operators</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>+ - / *</td>
<td>Arithmetic</td>
</tr>
<tr>
<td>String</td>
<td>+</td>
<td>Concatenates strings</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable 1</th>
<th>Variable 2</th>
<th>Result</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>Integer</td>
</tr>
<tr>
<td>1.0</td>
<td>2.0</td>
<td>3.0</td>
<td>Float</td>
</tr>
<tr>
<td>&quot;Spidey&quot;</td>
<td>&quot;Pig&quot;</td>
<td>&quot;Spidey-Pig&quot;</td>
<td>String</td>
</tr>
<tr>
<td>&quot;Johnny&quot;</td>
<td>5</td>
<td>FAILS</td>
<td></td>
</tr>
</tbody>
</table>

Operators

- The same symbol can be used for different operations
- + means either ‘addition’ or ‘concatenation’, depending on the context
  - Given two numbers, adds them together
  - Given two strings, concatenates the second to the first
  - Given a string and a number, will fail to work
- Fails because “+” is given two meanings
  - “Johnny” implies that + should concatenate two strings
  - 5 implies that + should add two numbers
- Since “Johnny 5” was expected, turn 5 into a string
  - Need str(): str(5) gives “5”
Functions

- `str()` is a function (purple in IDLE)
- Like functions in Algebra:
  - `x` is input
  - `f(x)` evaluates to output
- Functions are like commands that are “added in” to the basic Python language.
- The value of `str(10)` is the string “10”.

```
print "ECS " + str(10)
print "ECS " + "10"
print "ECS 10"
```

raw_input()

- `raw_input()` is a function
- It prints out its string, waits for a response.
- Its value is the user’s response (what the user types).
- Here, the value gets stored in the variable `reply`.

```
raw_input("Press enter to exit:")
reply = raw_input("Enter the answer:")
```