Programming Lessons 4 – 6:
If/Else,
Loops,
and
Functions

If/Else

- Used to make decisions based on whether a certain condition is true or false
  - The condition is in the parentheses
- Indenting matters (use tab)

```
3  if (a > 3):
4      print "Hello"
5  else:
6      print "Goodbye"
```

```
9  if (first_name == "Mary"):
10     print "Mary had a little lamb"
```

Logic Symbols

Equal To	==
Greater Than	>
Greater Than or Equal To	>=
Less Than	<
Less Than or Equal To	<=
Not Equal To	!=
And	and
Or	or

Using Elif

- If you want to provide the computer with more than two options, use **elif** as many times as needed

```
13  if (a == 3):
14      print "Apples"
15  elif (a == 4):
16      print "Bannas"
17  elif (a > 4):
18      print "Oranges"
19  else:
20      a = a + 4
```

Loops: For Loop

- Useful for accomplishing a task that is done repeatedly for a set number of times

```
1  for i in range(1,5):
2      print "Hello! My favorite number is", i
```

- This is what is printed to the terminal window:

```
Michael-Zitolos-MacBook-Air:Desktop user$ python test.py
Hello! My favorite number is 1
Hello! My favorite number is 2
Hello! My favorite number is 3
Hello! My favorite number is 4
```

Loops: While Loops

- Useful for doing something repeatedly until a certain condition is met

```
1   i = 1
2   while (i < 4):
3       print "Hello!  My favorite number is", i
4       i = i + 1
```

- This is what is printed to the terminal window:

```
Michael-Zitolos-MacBook-Air:Desktop user$ python test.py
Hello!  My favorite number is 1
Hello!  My favorite number is 2
Hello!  My favorite number is 3
```

Loops: Infinite While Loops

- If the condition next to while doesn't ever become false, the while loop will continue forever

```
1
2  while (i < 4):
3      print "Hello!  My favorite number is", i
4
```

- Since i will never become greater than 4, the loop will continue forever.

Lists

- Useful for holding a number of items that you may want to keep track of
- Defined using brackets []

```
2    shopping_list = ["apples", "oranges", "pears", "prunes"]
3
4    favoriteNumbers = [4, 8, 15, 16, 23, 42]
```

Functions

- Functions are things in programs that "do stuff"
- Functions you are already familiar with:

```
meal_order = raw_input("What would you like?")
range(2, 8)
```

- **Blue** = Function, **Green** = Arguments
- Benefits:
  - Can significantly reduce length of code
  - Can add "readability" to the code

Functions: Example

```
1   print "What is your favorite color?"
2   fav_color = raw_input("> ")
3   print "That's so cool, I love %s too!" % fav_color
4
5   print "What is your favorite animal?"
6   fav_animal = raw_input("> ")
7   print "That's so cool, I love %s too!" % fav_animal
8
9   print "What is your favorite food?"
10  fav_food = raw_input("> ")
11  print "That's so cool, I love %s too!" % fav_food
12
13  print "What is your favorite subject?"
14  fav_subject = raw_input("> ")
15  print "That's so cool, I love %s too!" % fav_subject
```

Functions: Example

- You can create your own function using **def**

```
18  def favs(category, user_fav):
19      print "What is your favorite %s?" % category
20      user_fav = raw_input("> ")
21      print "That's so cool, I love %s too!" % user_fav
22
23  favs("color", fav_color)
24  favs("animal", fav_animal)
25  favs("food", fav_food)
26  favs("subject", fav_subject)
```

Functions that Return Something

- Some functions return a value that needs to be stored in a variable (like raw_input( ))
- This is done using **return**

```
29  def adding_machine():
30      a = raw_input("> ")
31      b = raw_input("> ")
32      c = raw_input("> ")
33      sum = int(a) + int(b) + int(c)
34      return sum
35
36
37  print "Give me three numbers"
38
39  x = adding_machine()
40
41  print "Okay, the sum of your numbers is %d" % x
```

Conventions for Functions

- Normally named to give the reader an idea of what the function does
  - Adding a comment above it to explain what the function does (instructions) is usually helpful
- All functions are defined at the start or end of the program, not in the middle of a program
- For Python, indenting is required for functions