Week Three

Conditionals, Loops, and Functions

Conditionals

- A conditional defines a certain “condition” in your code, which the computer determines as being “met”, or “not met.”
- The primary conditional is the “IF” statement.
- When you begin a line with “if” and indent a block of code, python will only run that code if the condition you define is met.

Conditionals

- For example:

```
if x > 5:
    print ‘You win the game.’
```

- The statement will print only if some variable ‘x’ happens to be greater than five when python arrives at this line of code.

- Remember to include the colon at the end of the if statement line, and to indent any lines of code you want to include in the conditional statement.

Conditionals

- You are not limited to using only one condition- you can use the special word ‘and’ to require multiple conditions to be met, or you can use ‘or’ to require only one of several possible conditions be met. You can also use parenthesis to group any combination of these together, and use contractions like <= and >=.
- Example

Conditionals

- What happens if the condition is not met? We can include an else statement:

```
if x > 5:
        print ‘Yay!’
else:
        print ‘Aw’
```

Conditionals

- If you want to include cases for multiple possible conditions, use the ‘elif’ statement:

```
if (x > 5) and (x>1):
    print ‘case 1’
elif (x<10) and (x>5):
    print ‘case 2’
```

Loops

- Loops allow us to set off a block of code to be repeated under various conditions.
- Two main kinds of loops: While-loops, and For-loops

While-Loops

- A while loop is a block of indented code that will continue to run over and over so long as the conditional statement at the top is evaluated as true.
- Example
- Note: Be careful defining while loops! If the code doesn't have some mechanism within the loop to make sure the conditional will eventually be broken, then your code will hang in the while-loop forever!

For-Loops

- For-Loops allow us to *iterate* over a certain range of values within a block of code.
- We can iterate over integer numbers (such as a range), or over elements in a list or array.
- We define some variable (anything we want, “i”, ‘j’, and ‘k’ are common) to be the iterator, and include that variable in our block of code somewhere so that it runs slightly differently for each ‘i’ it plugs in.
- Example
- Always keep track of whether you are iterating over elements or indices (via range)

Functions

- So far, the functions we have used are built into python or part of a package (e.g., numpy, matplotlib). Python also allows you to define your own functions
- You can specify which arguments your function takes, and what it returns
- Unlike a script (just code in a file), your function doesn't "run" when your code is run; instead you have to actively call your function (and if your function returns something, you have to set a variable).

Script vs. Function

Example script:

```
x=5
y=7
print x+y
```

Example function:

```
def add(num1, num2):
    x=num1
    y=num2
    sum = x+y
    return sum

print add(5,7)
var = add(12,15)
```

Functional Programming

- It is easy to make functions to do small things, but often times it becomes really helpful to use them on big chunks of your code (like a pipeline).
- Say your job was to pull in some images, do some corrections, do an analysis, and then plot the output. It would be wise to write separate functions to load images/perform corrections, do the analysis, and then plot nicely.