1. Python basics

1.1. A simple function. Think of a function as a black box that takes inputs and returns outputs. In Python, the function name follows the word `def`, and the input is inclosed in parentheses.

```python
def func(x):
    In this example, our function is called `func`, and the input is `x`.
    Inside of the function, we do operations on the input and then return the output. Let our function add 5 to the input and assign the result to the output, which we will call `y`.
    
    `y = x + 5`
    
    Now that we have calculated the result, we need to send it to the output, which is done using the `return` statement.
    
    `return y`
    
    Our complete function looks like this – don’t forget the colon (:) and the indentations.
```

Date: August 27, 2009.
>>> def func(x):
...     y = x + 5
...     return y

Once our function is defined in the computer’s memory, we can call the function by typing `func(5)` at the python prompt (it looks like this `>>>`). Since our input was 5, we should get 10 as the result.

```python
>>> func(5)
10
```

1.2. Making decisions. Python functions can also make decisions and perform operations that are specific to certain inputs. In the next example, we will write a function that adds ‘gy’ to the input if the input is ‘dog’, and do nothing otherwise. To do this, we need to use the if statement.

```python
if s == "dog":
    s = s + "gy"
```

The if statement checks to see if the condition is true, i.e. if the input to the function was “dog”. If the condition is true, the operations defined below the if statement are executed, otherwise nothing happens. The complete function looks like this:

```python
>>> def doggy(s):
...     if s == "dog":
...         s = s + "gy"
...     return s

>>> doggy("dog")
'doggy'
>>> doggy("cat")
'cat'
```

Often we want our program to check multiple conditions and act differently for each one. To do this, we use the else and elif statements. The elif statement is short for else if. Let’s look at the Fibonacci function introduced in lecture 1.

```python
>>> def fib(n):
...     if n == 0:
...         return 0
...     elif n == 1:
...         return 1
...     else:
...         return fib(n-1) + fib(n-2)

>>> fib(5)
10
```
In this function, the first condition checks whether the input is 0. If it is 0, we output 0. Next, the program uses an `elif` statement to check if the input is 1, and outputs 1 if the condition is true. If the output is neither 0 nor 1, the `else` statement is executed, which is a recursive call to the function, with decremented inputs. The recursive function will keep repeating until the input is either a 0 or a 1.

2. Recursion in Natural Languages

Noam Chomsky and others have argued that recursion is the property that distinguishes human languages from animal communication. It is easy to come up with examples of recursion in English. For example, take the Noun Phrase (NP) “John’s house.” We can write a phrase-structure rule to describe this NP:

\[ NP \rightarrow AP \ N \]

This rule says that an NP consists of an adjective phrase (AP) and a noun (N).

A simple rule for the AP would be:

\[ AP \rightarrow A \]

However, we can add more adjectives to our NP: think of “John’s mother’s house”, or “John’s mother’s friend’s house” or “John’s mother’s friend’s daughter’s house”, etc. It appears that we need a second rule for AP:

\[ AP \rightarrow A \ AP \]

This new rule is recursive, since it refers to itself. Recursion allows us to construct an infinite number of sentences from a finite number of words.