**Lecture 6 (week 4.1)**

Tuesday 27th 2016

Functions, Conditionals and Recursion

Another fundamental concept in programming is the one of *conditionals*. Often a program has to take a decision based on some condition. From now on, you will start using all the concepts learned so far and mix them! The combination of functions, conditionals and recursion will allow you start real programming.

As a start, I want you to read section 5.1 up to 5.12 in <http://greenteapress.com/thinkpython2/html/thinkpython2006.html>

The input statement

Before looking at the conditional statements, let me introduce the **input()** function. **input()** is a built-in function (<https://docs.python.org/2/library/functions.html>). It allows the user to enter information in a program while it is executing. The **input()** function works also from the Python prompt, but it is most useful inside a program. On the prompt, this is how you would use it:

**In [1]: x=input("Enter value: ")**

**Enter value: stringvalue**

**In [2]: x**

**Out[2]: 'stringvalue'**

**In [3]: type(x)**

**Out[3]: str**

**In [4]: x=input("Enter value: ")**

**Enter value: 3.4**

**In [5]: x**

**Out[5]: '3.4'**

**In [6]: type(x)**

**Out[6]: str**

Note that i**nput()** returns a string, you want to convert to a numerical type (float, int, etc…) to use it as a number.

The **if-else** statement

This conditional statement will allow the program to take some specific action if the condition is **True**, or a different action if it is not. The generic syntax is:

**previous statements**

**if (condition) :**

**statements if condition is True**

**else :**

**statements if condition is False**

**subsequent statements**

Note that **if-else** is written as a block (and remember, *indentation* is mandatory). Note also that the column “**:**” is mandatory (same as for functions). Imagine you want to calculate **sin(x)/x** but you want to avoid a runtime error when x is zero. The following script would just do this:

**import math**

**x = input("Enter x value: ")**

**x=float(x)**

**if (x != 0) :**

**y = math.sin(x)/x**

**else :**

**y = 1.0**

**print("sinc(", x, ")=", y)**

I created a script with these commands and called this file *sinc.py*. When executing it, you get the following output:

**In [3]: run sinc.py**

**Enter x value: 0.**

**sinc( 0.0 )= 1.0**

**In [4]: run sinc.py**

**Enter x value: 1.3**

**sinc( 1.3 )= 0.7411986041670715**

What happens when you run it with no input?

It is also possible to run an **if-else** statement without the **else**. As an interesting twist to the above example, consider the following script:

**import math**

**x = input("Enter x value: ")**

**x=float(x)**

**if (x != 0) :**

**y = math.sin(x)/x**

**if 'y' in locals():**

**print("sinc(", x, ")=", y)**

**if ('y' in locals()) == False:**

**print("I do not calculate this”)**

It calculates sin(x)/x, but now it will not do it if x=0, and without using the **if-else** statement, only the **if** part. Experiment with it. What is **locals()**? what are the **if** statements doing?

The **while** loop statement

This conditional statement keeps running internally as long as the condition is **True**. The generic syntax is:

**previous statements**

**while (condition) :**

**statements if condition is True**

**returns to the while loop until the condition is False**

**subsequent statements**

Note the indentation and the “**:**”, both are mandatory again! Imagine you want to calculate all values of sin(x)/x for x going from xmin to xmax with an incremental value dx, you would write this script:

**import math**

**xmin=0.**

**xmax=100.**

**dx=1.**

**x=xmin**

**print("Begins here")**

**while (x<=xmax):**

**if (x != 0) :**

**y = math.sin(x)/x**

**else :**

**y = 1.0**

**x=x+dx**

**print("sinc(", x, ")=", y)**

**print("Ends here”)**

Explore the following: when does the loop end? what are the blocks structure of this program? What happens if I eliminate the **x=x+dx** line? what happens if I define **xmax** a value smaller than **xmin**?

Now I want you to read Section 6.1 to 6.9 in <http://greenteapress.com/thinkpython2/html/thinkpython2007.html>

Note that for the **if-else** and **while** statements, the condition **(condition)** does not have to be necessarily between parenthesis, both syntaxes are accepted.

Scipy fitting routine

This last paragraph is an example of fitting data points with a model in order to find out what the parameters of the model are. It is a bit advanced because we have not covered half of the material in it, however we will do it very soon. I give this example now to give you an early exposure on the Python scientific library **scipy**. We will use the **scipy** fitting routine **curve\_fit()**, here is where you can read more about it:

<http://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html>

The data points are

**# x y**

**0.00 0.00**

**1.00 1.84**

**2.00 2.91**

**3.00 3.14**

**4.00 3.24**

**5.00 4.04**

**6.00 5.72**

**7.00 7.66**

**8.00 8.99**

**9.00 9.41**

copy them to a file called **data.txt**. Now write the following fitting script to a file and run it:

**import numpy as np**

**import matplotlib.pyplot as plt**

**import scipy.optimize**

**def fct(x,A,B):**

**return A\*np.sin(x)+B\*x**

**z=np.loadtxt('data.txt')**

**x=z[:,0]**

**y=z[:,1]**

**fitparams,fitcov=scipy.optimize.curve\_fit(fct,x,y,p0=(0.,0.))**

**plt.plot(x,fct(x,\*fitparams))**

**plt.plot(x,y,'ro')**

**plt.show()**

As a home exercise, can you figure out what the script is doing? it is not easy, but you should be able to recognize some of the things we have used before (**numpy** and **def**). What is new is the call to the **scipy** library, the plotting, and the format of some variables.