Notes & Summary For Section 2

Section 3, Lecture 27

**Notes and summary for section 2**

**If statements in Python:**

In order to write useful programs, we almost always need the ability to check conditions to change the behavior of the program accordingly. Conditional statements allow us to do so. The simplest form is the if statement, which has the general form:

if BOOLEAN EXPRESSION:  
    STATEMENTS

**A few important things to note about if statements:**

The colon (:) is significant and required. It separates the header of the compound statement from the body.

The line after the colon must be indented. It is standard in Python to use four spaces for indenting.  
All lines indented the same amount after the colon will be executed whenever the BOOLEAN\_EXPRESSION is true.

The boolean expression after the if statement is called the condition. If it is true, then all the indented statements get executed. What happens if the condition is false,  In a simple if statement like this, nothing happens, and the program continues on to the next statement.

**Elif statement in Python:**

Sometimes there are more than two conditions and we need more than two branches. In such cases we use the elif statement.elif is an abbreviation of else if. Again, exactly one branch will be executed. There is no limit of the number of elif statements but only a single (and optional) final else statement is allowed and it must be the last branch in the statement:

 if choice == 'a':  
    print("You chose 'a'.")  
elif choice == 'b':  
    print("You chose 'b'.")  
elif choice == 'c':  
    print("You chose 'c'.")  
else:  
    print("Invalid choice.")  
      
Each condition is checked in order. If the first is false, the next is checked, and so on. If one of them is true, the corresponding branch executes, and the statement ends.  
   
   
**Lists in Python:**  
Python has six built-in types of sequences, but the most common ones are lists and tuples.

There are certain things you can do with all sequence types. These operations include indexing, slicing, adding, multiplying, and checking for membership. In addition, Python has built-in functions for finding the length of a sequence and for finding its largest and smallest elements. The list is a most versatile datatype available in Python which can be written as a  list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.  
   
   
 Creating a list is as simple as putting different comma-separated values between square brackets

For example

listone = ['physics', 'chemistry', 1997, 2000];

**List functions in Python**

**append() :** to append an item to the list.  
to append to a list named fruits, type in fruits.append("Banana")

**function to calculate the length of the list:**  
print(len((fruits))  
This line of code will print out the length of the list named fruits

**Insert function:**  
This function allows you to insert some item to the list, this function is similar to append but   
it allows you to insert item at a particular position.  
example:  
fruits.insert(1,"banana")   
This line of code places the item banana at position 1 in the list fruits.

**The index function.**  
it returns the index value / position of particular item in the list.  
Example:  
print(fruits.index("Peach"))  
This line of code will return the index position of item Peach.

**Range function in Python**  
The built-in range function in Python is very useful to generate sequences of   
numbers in the form of a list.

The given end point is never part of the generated list;

range(10) generates a list of 10 values, the legal indices for items of a   
sequence of length 10.

It is possible to let the range start at another number, or to specify a  
different increment (even negative;

Sometimes this is called the ‘step’):

example:

>>> range(1,10)  
[1, 2, 3, 4, 5, 6, 7, 8, 9]

# You can use range() wherever you would use a list.

a = range(1, 10)   
for i in a:   
    print i

for a in range(21,-1,-2):  
   print a,

#output>> 21 19 17 15 13 11 9 7 5 3 1

# We can use any size of step (here 2)  
>>> range(0,20,2)  
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

>>> range(20,0,-2)  
[20, 18, 16, 14, 12, 10, 8, 6, 4, 2]

# The sequence will start at 0 by default.   
#If we only give one number for a range this replaces the end of range value.  
>>> range(10)  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

# If we give floats these will first be reduced to integers.   
>>> range(-3.5,9.8)  
[-3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8]

**Functions in Python:**  
A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

As you already know, Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called user-defined  functions.  
   
**Defining a function:**  
You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses ( ( ) ).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.

The first statement of a function can be an optional statement - the documentation string of the function or docstring.The code block within every function starts with a colon (:) and is indented.The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

**Calling a function:**  
 Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.Once the basic structure of a function is finalized, you can execute it by   
calling it from another function or directly from the Python prompt.

**For loop in Python:**  
The for loop processes each item in a sequence, so it is used with Python’s sequence   
data types - strings, lists, and tuples.Each item in turn is (re-)assigned to the loop variable,  
 and the body of the loop is executed.

The general form of a for loop is:

for LOOP\_VARIABLE in SEQUENCE:  
    STATEMENTS

This is another example of a compound statement in Python, and like the branching statements, it has a header terminated by a colon (:)  and a body consisting of a sequence of one or more statements indented the same  
 amount from the header.The loop variable is created when the for statement runs, so you do not need to   
create the variable before then. Each iteration assigns the the loop variable to the next element in the sequence, and then executes the statements in the body. The statement finishes when the last element in the sequence is reached.This type of flow is called a loop because it loops back around to the top after each   
iteration.

**While loop in python**  
The general syntax for the while statement looks like this:

while BOOLEAN\_EXPRESSION:  
    STATEMENTS

Like the branching statements and the for loop, the while statement is a compound statement consisting of a header and a body. A while loop executes an unknown number of times, as long at the BOOLEAN EXPRESSION is true.