Lecture Notes on
Sequence and Loops

15-112: Introduction to Programming and Computer Science
Saquib Razak

Lecture 1
August 26, 2019

1 Sequence of instructions

We will start our discussion of Python by talking about algorithms. An Algorithm is a sequence of instructions that in the end solves a specific problem. While specifying the instructions, it is extremely important that we apply the following principles:

- We have to make sure we have all the instructions - no missing parts
- We have to make sure we have the instructions in the right order
- There should not be any unwanted/extra/useless instructions

Let's understand this with a simple example. Lets say you have two commands:

- Forward n - Moves forward by n meters and draws a line on the ground
- Left d - turns left by d degrees

Our simple task is to use the above commands to draw a square of length 10 meters on the ground. Its not too hard to imagine that the following sequence of instructions will achieve this task:

```
Forward 10
Left 90
Forward 10
Left 90
Forward 10
Left 90
Forward 10
Left 90
```

You can argue that the last "Left 90" is not necessary and you would probably be correct. The above example points out two things: One, that we need to know what are the basic commands are we are able to execute (in this case Forward and Left) and two, we need to put those commands in the right sequence to achieve our final results. In section 3, we look at a library in Python called Turtle graphics to practice writing simple algorithms. But before we can get there, let's look at how we can get started with Python.

2 Getting Started with Python

You can use the following instructions to write a small program that prints some text on the screen using Python.

- Go to start menu and look for $Python2.7 -> IDLE$
- Click on the file menu and click on New - this will create a new window
- write the following text in this new window

```
print "Hello World!!"
```

- Press F5 to run the program, it will ask you to save the file, Click on yes and then save the file in a folder inside the Documents folder. Make sure you give your file an appropriate name and a .py extension.
- When you are done, you should see "Hello World!" printed on your IDLE window.

3 Turtle Library

Python comes with a pre-installed library called Turtle. Turtle is a simple graphics library that allows you to draw on the screen using simple primitives. Two of these primitives are: a function called forward(n) and a function called left(d). Similar to our example above, forward(n) moves the turtle in the direction it is facing by n centimeters while drawing a line on the screen as it moves. The function left(d), makes the turtle turn left by d degrees. We can draw a square on the screen by following the given instructions:

- Start IDLE
- Go to file menu and click New to start a new Python code editor
- type the following code:

```
from turtle import *   # this line allows us to use the Turtle library

forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)

done()
```

- press F5 to save and run the program
- You should see the turtle move around on the screen and draw a square

To learn more about the turtle library, please refer to the following website:
https://docs.python.org/2/library/turtle.html

4 Repetition

You might have noticed in the above example that we had to repeat a set of instructions (forward(100),left(90)) four times to draw the square. This repetition is a common occurrence in programming so there is an easy construct for this in the python language.

You can use the for loop to repeat a sequence of instructions a set number of times. For example, we can write the square program the following way:

```
from turtle import *   # this line allows us to use the Turtle library

for i in range(4):
   forward(100)
   left(90)

done()
```

The statements that are indented after the "for i in range(4)" instruction are executed four times in sequence. In a generic case, if we have a sequence of n instructions that we need repeated x number of times, we can write the following code:

```
for i in range(x)
   statement 1
   statement 2
   statement 3
   .....
   statement n
```

5 Functions

Recall the following code that draws a square:

```
forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)
```

or

```
for i in range(4):
   forward(100)
   left(90)
```

Both pieces of code perform the same task - drawing a square. Both code assume that the turtle library has already been imported. It woule be beneficial if I can name this code as ”drawSquare“ and then ask that this code be executed whenever I want to draw a square. We can give a name to a set of instructions by using the def (short for define) command. In the following code, we use the shorter version of drawing a square and give it the name ”drawSquare“.

```
def drawSquare():
   for i in range(4):
     forward(100)
     left(90)
```

Note that you need the open and close parenthesis followed by a colon ":" right after the function name. The function body (the statements that are part of this function) are indented by one tab space. This is called "defining a function". Note that when you define a function, it does NOT cause the function to be executed. If we want to run the function, we would call the function as follows:

```
drawSquare()
```

The whole code will look something like the following:

```
from turtle import *

def drawSquare():
    for i in range(4):
      forward(100)
      left(90)

drawSquare()
```

Now I can take advantage of this new function and use it to draw several squares:

```
from turtle import *

def drawSquare():
    for i in range(4):
      forward(100)
      left(90)


drawSquare()
forward(200)
drawSquare()
forward(200)
drawSquare()
forward(200)
```

Usually, and we will follow this style very closely, your program starts out with all the import statements, followed by all function definitions, followed by your code:

```
 #all import statements go here
 import xyz
```

```
from abc import *
import pqr
...


# all function definitions
def funcOne():
 function body
 ...

def funcTwo():
 function body
 ...


# your code
statement 1
statement 2
...
```