Exercise 6: Arguments, unpacking, variables, passing

By: Rahul Sharma 20 April, 2018

In the last lecture you saw how to take inputs from a user which can be stored in a variable and used anywhere in the program. However, there is another way to take inputs from the user to the program.
Every time you ran the script in the command prompt or terminal using `python script_name.py`, the word 'python' was the command and the 'script_name.py' was an "argument" to the command. For now, you can also say these arguments by another name which is parameters. You can also use arguments to provide inputs to the python script by writing in you command prompt something like:

```
python script.py argument1 argument2 argument3
```

So where does these arguments that you have "passed" to the code get stored so that you can use them into variables inside the code? Well, let's understand it with the help of an example.

```
from sys import argv
script, first, second, third = argv
print "The script is called:", script
print "Your first variable is:", first
print "Your second variable is:", second
print "Your third variable is:", third
```

In the first line we have an "import". The import is used to include "features" into your program. We call them in programming terms, modules or libraries. But, I'm going to stick to the modules for the rest of the course. Think of modules as a friend who is good at something. Well, we all are good at something in our own way. Okay, so this friend of yours knows something which you are not experienced in and you really need the work to be done. Therefore, in programming terms you're going to "import" this friend to your code and do the work for you. There are a number of friends in our lives, similarly, there are a number of modules or libraries in the python language.

Here we are using 'argv' (short for argument variable) functionality of the 'sys' module. In short, the argv stores the arguments that we are passing during the program run in the command prompt and lets us use them later in the code by "unpacking" them into variables. It will be better to use it in an exercise to understand better.

Exercise 1: Arguments, unpacking and variables

1. Create a file called **ex6_1.py** and write the code discussed above in it
2. Run the file like this: python ex6_1.py pentagram research centre ltd
3. Write a comment above each line describing what is it doing
4. Write a script that has fewer arguments and one that has more. Make sure you give the unpacked variables good names. Name the scripts **ex6_1_fewer.py** and **ex6_1_more.py**

5. Combine raw_input with argv to make a script that gets more input from a user.

Let's do a script that uses raw_input and argv together to ask the user something specific. Notice that I'm using a variable inside the raw_input() to avoid writing the same thing again and again.

Exercise 2: Prompting and passing

1. Create a script called ex6_2.py and write the following code in it

```
from sys import argv
script, user_name = argv
prompt = '> '
print "Hi %s, I'm the %s script." % (user_name, script)
print "I'd like to ask you a few questions."
print "Do you like me %s?" % user_name
likes = raw_input(prompt)
print "Where do you live %s?" % user_name
lives = raw_input(prompt)
print "What kind of computer do you have?"
computer = raw_input(prompt)
print """
Alright, so you said %r about liking me.
You live in %r. Not sure where that is.
And you have a %r computer. Nice.
""" % (likes, lives, computer)
```

2. Change the prompt variable to something else entirely.
3. Add another argument and use it in your script.
4. Make sure you understand how I combined a """ style multiline string with the % format activator as the last print.

Note: Submit your exercises before April 21, 2018 11:59 PM

Next Lecture: Reading files