Exercise 8: More Files and Introduction to Functions

By: Rahul Sharma 26 April, 2018

Let's play a little more with the file reading and writing. Here, we will try to copy the contents of one file to another. This lecture will also introduce you with python functions which will allow you to create your own chunks of code which you can pack inside a unique name and use it anywhere in the code.

Here we are importing yet another module called `exists` from os.path.

Exercise 1: Copying contents of one file to another

1. Write the steps that you would follow to copy the contents of one file to another?
2. Create a file called **ex8_1.py** and write the following code in it:

```
from sys import argv
from os.path import exists
script, from_file, to_file = argv
print "Copying from %s to %s" % (from_file, to_file)
# we could do these two on one line too, how?
in_file = open(from_file)
indata = in_file.read()
print "The input file is %d bytes long" % len(indata)
print "Does the output file exist? %r" % exists(to_file)
print "Ready, hit RETURN to continue, CTRL- C to abort."
raw_input()
out_file = open(to_file, 'w')
out_file.write(indata)
print "Alright, all done."
out_file.close()
in_file.close()
```

3. What do you think the exists() function is doing?
4. Write a comment above each of the line telling what is it doing?
5. To do a simple copying task we are using too much of prints and unnecessary commands. Consider that you want to copy a 1000 files, will you be willing to see all of these print statements a thousand times? Try to make this script shorter in a file called **ex8_1_short_version.py**
6. Write in a comment at the end of the file ex8_1.py about what the following two are doing?
    6.1 len(indata)
    6.2 exists(to_file)

Let's move on to one of the most important and fundamental parts of any project that involves a lot of code: FUNCTIONS. Functions are very important in day to day life. Basically, functions do three things:

1. They name pieces of code the way variables name strings and numbers.
2. They take arguments the way your scripts take argv.

3. Using #1 and #2, they let you make your own “mini- scripts” or “tiny commands.”

While all languages have their way of creating a function, in Python we use the *def* word. Def really is short for 'define'. There is a syntax associated with the functions. We will talk about this in just a moment. Before that, let's take a look at four functions that do what we have already done in the previous exercises.

Exercise 2: Introduction to functions

1. Create a file called **ex8_2.py** and add the following code to it:

```
# this one is like your scripts with argv
def print_two(*args):
      arg1, arg2 = args
      print "arg1: %r, arg2: %r" % (arg1, arg2)
# ok, that *args is actually pointless, we can just do this
def print_two_again(arg1, arg2):
      print "arg1: %r, arg2: %r" % (arg1, arg2)
# this just takes one argument
def print_one(arg1):
      print "arg1: %r" % arg1
# this one takes no arguments
def print_none():
      print "I got nothin'."
print_two("Zed","Shaw")
print_two_again("Zed","Shaw")
print_one("First!")
print_none()
```

Let's understand the steps in creating a function:

1. Write the *def* keyword
2. Write the name of the function (it can be set as any word of your choice as long as the name makes sense of what that function does)
3. We open the parenthesis using (
4. We place the arguments/parameters inside the parenthesis
5. Close the parenthesis using)
6. Place : (colon) character
7. Press enter and start indenting your code to make it a part of the function using the tab space.
8. Dedent (opposite of indent) to complete your function.

The first function in the code works like your argv module. The only difference is that we are using *args in functions. Alternatively, we can give the number of arguments that we want inside a function like the second function in the code. Third one takes only one argument and fourth one takes no arguments at all.

Note: This lecture has associated checklist for creati