lasp_reu_python_tutorial_day1_exercises

June 11, 2016

1 LASP REU Tutorial Day 1 Instructions

1.1 Write a simple non-interactive script

In a terminal, type the following to open a text editor:

```
~$ gedit &
```

Type in a program that * prints out the numbers from 10 to 30 * with a step of 2,
using a `for` loop and the `range` function.
Save the program as "simple_loop.py" into your "Home" directory.
[The terminal might show some `gedit` related errors, ignore them by pressing return and getting a new input prompt.]
Go back to the terminal, and execute your program using:

```
~$ python simple_loop.py
```

1.2 Work with Jupyter notebook

Close gedit.
Launch a jupyter notebook with

```
~$ jupyter notebook &
```

(The `&` symbol means "Launch an independent process and give me back the terminal prompt")
A webbrowser should open. (If asked which one, pick Chromium, which has a good support for Jupyter notebook)
In the upper right, click on the button named "New" and in the dropdown menu, pick the item in the "notebooks" subsection that is most similar to Python3.

1.2.1 Write a function to calculate the factorial of a number

There are several ways to implement this.
As a reminder, the factorial of 3 is `1*2*3`, the factorial of 5 is `1*2*3*4*5`. You should have learned enough to be able to come up with a combo of some kind of loop, either `for` or `while`.
As a reminder about functions, your skeleton function should look like this:

```
def calc_factorial(n):
    <do the right stuff>
    print("The factorial of",n,"is",result)  # if you store the result in `result`
```

1.2.2 Dictionaries

The last very important data container in Python that we did not discuss yet are called `dictionaries`.

They work by storing a value with a certain key, so that that key can be used later to get that value back.

Similar to how a list is initialised and recognized via the square brackets `[]`, dictionaries are recognized and initialized by curly brackets `{}`:

```
In [10]: mydict = {}
         mydict['a'] = 5
         mydict['b'] = 10
         mydict[4] = 'mystring'
         mydict

Out[10]: {4: 'mystring', 'b': 10, 'a': 5}
```

Note, that anything that is an unambigously identifiable Python object can be used as a key.

Which is why the only simple certain choices are integers and strings.

One can loop over the dictionary values and get both key and value at once:

```
In [11]: for k, v in mydict.items():
             print(k, v)

4 mystring
b 10
a 5
```

Note, that dictionaries don't store things in the order they have received it, for maximum performance (optimzing memory). (there are sorted versions if required).

1.2.3 Task: create function that combines 2 lists into dictionary

Program a function that takes a list of first_names and last_names and stores them into a dictionary.

As you might have guessed, keys have to be unique, but values don't have to. So don't try this with a list of first_names that is non-unique (unless you want to learn about error handling. ;)

So, take the list ['john', 'james', 'jacob'] as first_names for example, and the list ['smith', 'baker', 'anderson'] as last_names and give them to the function.

Your function skeleton would look like this:

```
def my_dict_creator(list1, list2):
    <do the right stuff and create mydict>
    return mydict
```