A gradual set of exercises to get up and running with Python

Juan David González Cobas

July 15, 2010

This is a set of graded exercises (from trivial to overly complex) to harden your teeth on Python. I would advise you to provide for each a separate source file that executes in this fashion

```
\$ python name_of_exercise.py
```

An example of this being the very easy solution to exercise 1, named `sum_numbers.py`. To ease things a bit, I also provide the solution of exercise 4 in a couple of ways, from naïve to sophisticated.

Ok, let's go.

1. Write a program that asks the user for a number repeatedly and stops when a negative number is input; then print the sum of all the numbers put in by the user

2. Write a program that asks repeatedly for a number as in the previous exercise, but produces at the end the complete list of numbers the user provided. Hint: you'll need to build a list and `append` to it the successive inputs.

3. Write a program that asks the user for a file name, then opens the file and writes its contents on screen. Hints: look at `file`, and its `read()`, `readlines()` and similar methods

4. Write a program that prints the values of the sine of each number between 0 and 2π, with a step of $2\pi/100$.

5. Same program, but the number of divisions is provided as input by the user

6. Same program, but now the number of divisions is given as a command-line argument, in this fashion

   ```
   \$ python sinewave.py 2000
   ```

   for 2000 divisions, and so on

7. Create a *function* that takes as arguments the number of samples (divisions) and returns a vector with the sine wave values as done in the previous examples.

8. Do the above using the `numpy` package `sin` function (if you didn't already do so)

9. Write a program that asks the user for a number of samples, creates a vector with that many samples of a sine wave period, and shows the values of the FFT of it

10. Same idea, but now the program has to show *only* the modulus and argument of the first and last harmonic (this means the only ones that are not zero)

11. Write a program that takes as a command-line argument a filename, in this fashion

    ```
    $ python fft.py datafile.txt
    ```

    The file will contain a list of floating-point values, separated by whitespaces (in normal cases, it will be one number per line). Construct a vector with that, find out its FFT and produce the modulus of the FFT as output. Hint: you'll need to use `file()`, `file.read()`, `string.split()`, `numpy` and `fft()`, among others.

12. Take a look at the documentation of the (enormous) `matplotlib` package, and extend the program above by displaying a plot of the modulus of the FFT that you got as numerical values.