ESE 3.08: Programming for Geoscientists — Introduction to Python

Class test: 10th December 2012

Introduction

In each of the following questions you will find an explicit specification for a program. Each of your programs **must fulfill all** of those instructions. Please follow the instructions carefully and double check that your program fulfills all of the given instructions.

Question 1: Convert from meters to British length units

Write a program where the user specifies a length given in meters on the command line and then compute and write out the corresponding length measured in inches, in feet, in yards, and in miles. Use the conversion factors: one inch is 2.54 cm; one foot is 12 inches; one yard is 3 feet; and one British mile is 1760 yards.

Instructions for question 1

- Name of program file: `lengths_conversion.py`
- Use `sys.argv` to read in the length from the command line.
- Check that a single command line argument of the correct type is provided by the user. If it is not present, or of the wrong type, print the usage message `"Usage: %s meters"% sys.argv[0]` to the screen and exit the program with a return code of `1`.
- Make sure your program output matches exactly the format given in the listing below.

Example usage:

```
$ python lengths_conversion.py 640
25196.85 inches
2099.74 feet
699.91 yards
0.3977 miles
```

Question 2: Compute the area of an arbitrary triangle

A triangle can be described by the coordinates of its three vertices: $(x_0, y_0), (x_1, y_1), (x_2, y_2)$. The area of the triangle is given by the formula

$$A = \frac{1}{2}[x_1y_2 - x_2y_1 - x_0y_2 + x_2y_0 + x_0y_1 - x_1y_0].$$

Write a function `area(vertices)` that returns the area of a triangle as a single floating point number. The triangle vertices are specified by the single argument `vertices`, which is a nested list. For example, `vertices = [[0, 0], [1, 0], [0, 2]]` if the three corners of the triangle have coordinates $(0, 0), (1, 0)$, and $(0, 2)$. Test the area function by adding the following main program at the very end of your Python script:

```
if __name__ == '__main__':
    print area([[0, 0], [1, 0], [0, 2]])
```

The condition `if __name__ == '__main__'` guards the main program such that it is not executed when importing your module from another module.

Instructions for question 2

- Name of program file: `area_triangle.py`
- Your function **must** be called `area` and take a single parameter called `values` to accept the nested list.
- Do not use any `print` statements apart from the one given in the listing.

Question 3: Plot exact and inexact Fahrenheit-Celsius formulas

Here is a rule of thumb to quickly compute the Celsius temperature from the Fahrenheit degrees: $C = (F - 30)/2$.
Compare the curve obtained using this formula against the exact curve $C = (F - 32)\ 5/9$ in a plot. Let F vary between -20 and 120 with a spacing of 1.

Instructions for question 3

- Name of program file: `f2c_shortcut_plot.py`
- Define a function `C_approx` implementing the approximation and a function `C_exact` implementing the exact conversion. Both functions take a single parameter `F`, which is expected to be a NumPy `array` (*i.e.* your function performs a vector computation on the input array and returns an array).
- Use `linspace` to obtain a NumPy `array` of values for F and store it as a variable called `F`. Make sure they have a spacing of 1.
- Print your results by using the following print statements:
  - `print 'C_approx = ', C_approx(F)`
  - `print 'C_exact = ', C_exact(F)`
- Plot both curves in a single figure.
- Label the x axis `'F'` and the y axis `'C'`.
- Use the plot title `'Comparison of exact and inexact Fahrenheit-Celsius formulas'`.
- Display a legend with the labels `'C_approx(F)'` for the approximation and `'C_exact(F)'` for the exact curve.
- Save the plot using `savefig`, the filename must be `f-c-plot.svg`
- Inspect the saved file by running `display f-c-plot.svg` in a terminal.
- Do **not** use `show` to interactively show a plot window.
- Do **not** use any other `print` statements in your code.

Question 4: Write a function for numerical differentiation

The formula

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h},$$

can be used to find an approximate derivative of a mathematical function $f(x)$ if h is small. Write a function `diff(f, x, h=1E-6)` that returns this approximation of the derivative of a mathematical function represented by a Python function `f` passed in as an argument.

Use `diff` to differentiate $f_1(x) = e^x$ at $x = 0$, $f_2(x) = e^{-2x}$ at $x = 0$ and $f_3(x) = cos(x)$ at $x = 2\pi$. Use $h = 0.01$ in all cases.

Instructions for question 4

- Name of program file: `diff_f.py`
- Your differentiation function **must** be called `diff`, take 3 arguments `f`, `x` and `h` and return a floating point number with the approximate derivative computed using the formula above. Note that `h` is an optional argument with a default value.
- Define a separate function taking a single parameter `x` for each of the 3 expressions given above: name them `f1`, `f2` and `f3` respectively.
- Print out the 3 differentiation values using the following print statements:
  - `print "f1'(0.0)= ", diff(f1, 0.0, h)`
  - `print "f2'(0.0)= ", diff(f2, 0.0, h)`
  - `print "f3'(2*pi)= ", diff(f3, 2*pi, h)`
- Do not use any other `print` statements in your code.

Question 5: Star luminosities

Based on the star data in the file `/python-course/python-book-examples/src/files/stars.dat`, make a dictionary where the keys are the names of the stars and the values are a tuple containing the distance, apparent brightness and luminosity.

Instructions for question 5

- Name of program file: `stars_data_dict.py`
- Use a Python dictionary to store the data, **excluding the Sun**.
- All numerical values have to be of type float.
- Print out the dictionary **without any formatting**.
- Do not use any other `print` statements in your code.