ESE 2.18: Programming for Geoscientists — Introduction to Python

Class test: 11th 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 temperature units of Fahrenheit to Celsius

Write a program where the user specifies a temperature in Fahrenheit on the command line and then compute and write out the corresponding temperature in degrees Celsius. Use the conversion formula

$$C = \frac{5}{9}(F - 32).$$

Instructions for question 1

- Name of program file: `f2c.py`
- Use `sys.argv` to read in the temperature 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 f2c.py 100
100 degrees F corresponds to 37.7778 degrees C
```

Question 2: Store values in a nested list

Write a program, that creates a list t with 6 values, $0.1, 0.2, \ldots, 0.6$. Compute a corresponding list y of $y(t)$ values using the formula:

$$y(t) = v_0 t - g t^2,$$

where $v_0 = 6.0$ and $g = 9.8$. Store these two lists, t and y, in a new list t1. Write out a table with a column of t and a column of y values by traversing the data in the nested t1 list.

Instructions for question 2

- Name of program file: ball_table.py
- You may use list or NumPy array for t and y
- Print out a table header with the column names 't' and 'y'
- For printing the table, iterate the nested list t1, do **not** access the previously computed t and y lists directly.
- Print out the table t1 using format specifiers for floating point values such that the decimal points line up.
- Do not use any additional print statements.

Question 3: Implement the factorial function

The factorial of n, written as $n!$, is defined as:

$$n! = n(n-1)(n-2)\cdots 2 \cdot 1,$$

with the special cases

$$1! = 1, 0! = 1.$$

For example, $4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$, and $2! = 2 \cdot 1 = 2$. Write a function `fact(n)` that returns $n!$. Return 1 immediately if x is 1 or 0, otherwise use a loop to compute $n!$. Test the factorial function with the following main program:

```
if __name__ == '__main__':
    for n in range(7):
        print fact(n)
```

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 3

- Name of program file: `fact.py`
- The function **must** be called `fact` and take a single argument called `n`.
- The software should check that the supplied value is a non-negative integer. If it is not, raise a `ValueError` exception as follows:
  `raise ValueError("n must be a non-negative integer")`
- Do not use any `print` statements apart from the one given in the listing.

Question 4: Plot density of air at different temperatures

A table of temperatures and densities, in units of degrees (C) and kg/m^3, are given in the file `/python-course/python-book-examples/src/files/density_of_air.dat`
Write a program that reads in the data from file into a list `temperature` (first column) and `density` (second column) and plots the variation of `density` against `temperature`.

Instructions for question 4

- Name of program file: `hot_air.py`
- The input file contains blank lines and lines starting with a `'#'`, which you must ignore when reading in the data.
- You may use `list` or NumPy `array` for `temperature` and `density`
- Print the data read from the file using the following print statements:
  - `print 'temperature = ', temperature`
  - `print 'density = ', density`
- Plot the variation of `density` against `temperature`.
- Label the x axis `'Temperature (Celsius)'` and the y axis `'Density (kg/m^3)'`
- Use the plot title `'Density of air at different temperatures, at 1 atm pressure'`
- Display a legend with the label `'Air'`
- Save the plot using `savefig`, the filename must be `density_of_air.svg`
- Inspect the saved file by running `display density_of_air.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 5: Physical constants

Based on the data in the file `/python-course/python-book-examples/src/files/constants.txt`, make a dictionary where the keys are the names of the physical constant and the values are a tuple containing the numerical value and the units.

Instructions for question 5

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