Overview and grading

In this assignment, you are provided with Python files, and you are asked to implement functions in these files. The functions are designed to give you practice with control flow and basic arithmetic in Python as well as some basic containers. You will have to edit the Python files to fill in the functionality. The file hw1.zip is located in the 'Files' tab on Canvas and contains all the files you will need. Please turn in your code (preferably in a zip file as well) on Canvas.

1 Simple functions

For this problem, we have provided three files with function empty functions: `arith.py`, `strings.py` and `simple_math.py`. Your job is to fill in the functions so that they do what we have written they do (see the comments in each file). We are also providing a file `grader_part1.py` which contains a series of 30 tests. You can test your functions with this file by using the command `python grader_part1.py`. Make sure all your files are in the same directory. Do not modify the `grader_part1.py` file in any way, it is meant to help you test your functions. Also, please do not "hard code" answers for tests, I will be grading your code with other tests.

1. **Arithmetic**
   Implement the functions `arith2()`, `arith3()`, and `arith4()` in the provided file `arith.py`. As an example, the function `arith1()` has already been implemented for you. (14 points).

2. **Strings**
   Implement the functions `str2()`, `str3()`, and `str4()` in the provided file `strings.py`. As an example, the function `str1()` has already been implemented for you. Hint: the `len()` function may be useful for implementing `str4()`. See: http://docs.python.org/2/library/functions.html#len. (8 points).

3. **Simple math functions**
   Implement the functions `seq_add()` and `fact()` in the provided file `simple_math.py`. You are not allowed to use the Python math library. (8 points).

2 Converting coordinate systems

In this part, we will write code that converts data types between different three-dimensional coordinates systems. First, we will review three coordinate systems.

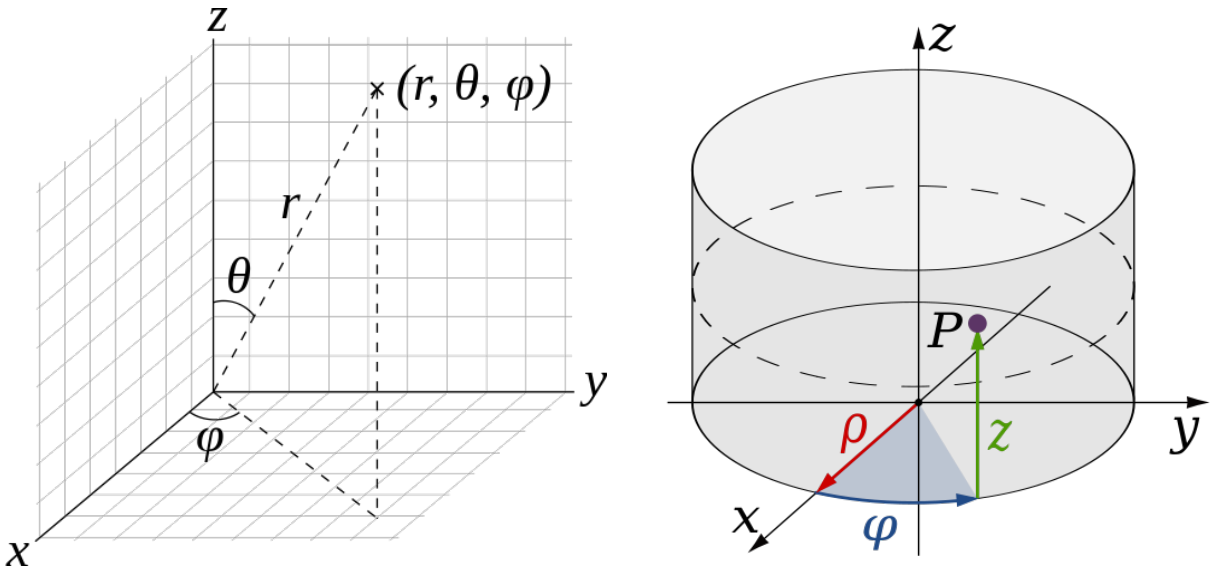
The first system is the familiar cartesian coordinate system. Three points, call them x, y, and z, are used to represent the point on three orthogonal coordinate axes.

The second system is spherical coordinates, which represents a point by a nonnegative radial distance r, an an inclination angle θ, and an azimuthal angle ϕ. (r, θ, ϕ) represent a point on a sphere centered at the origin. The two-dimensional analog to spherical coordinates is polar coordinates, and hence spherical coordinates are sometimes referred to as polar coordinates. See Fig. 1a for the relationship between spherical and cartesian coordinates.

The third system is cylindrical coordinates, which represents a point by a nonnegative radial distance ρ, an angle ϕ, and a height z. (ρ, ϕ, z) represents a point on a cylinder centered at the origin. See Fig. 1b for the relationship between cylindrical and cartesian coordinates.

We will use the following formulas to convert between coordinate systems in this assignment:

- cartesian $\rightarrow$ spherical: $(r = \sqrt{x^2 + y^2 + z^2}, \theta = \cos^{-1}\left(\frac{z}{r}\right), \phi = \tan^{-1}\left(\frac{y}{x}\right))$



(a) Spherical coordinates

(b) Cylindrical coordinates

Figure 1

- spherical → cartesian: $(x = r\sin\theta\cos\phi,\ y = r\sin\theta\sin\phi,\ z = r\cos\theta)$
- cartesian → cylindrical: $(\rho = \sqrt{x^2 + y^2},\ \phi = \tan^{-1}\left(\frac{y}{x}\right),\ z = z)$
- cylindrical → cartesian: $(x = \rho\cos\phi,\ y = \rho\sin\phi,\ z = z)$
- spherical → cylindrical: spherical → cartesian → cylindrical
- cylindrical → spherical: cylindrical → cartesian → spherical

where the last two conversions are compositions of the other conversions.

1. Finish the implementation of the functions `cart2cyl()`, `sphere2cart()`, `sphere2cyl()`, `cyl2cart()`, and `cyl2sphere()` in the file `coordinates_tuples.py`. The function `cart2sphere()` has already been implemented for you. These functions convert between the coordinate systems by representing points as Python tuples in (x, y, z), (r, θ, ϕ), or (ρ, θ, ϕ) form. Remember to use your other functions to implement `sphere2cyl()` and `cyl2sphere()`.

   For cos^{-1}, use `math.acos()`; for $\sin^{-1}$, use `math.asin()`; and for $\tan^{-1}$, use `math.atan2()`. `math.atan2()` maintains the quadrant information of its two input parameters, while `math.atan()` does not. See http://docs.python.org/2/library/math.html#trigonometric-functions for more information. (10 points)

2. Finish the implementation of `convert_points()` in the file `coordinates_tuples.py`. This function converts a list of points in one coordinate system to a list of points in another coordinate system. The implementation has been started for you. (5 points)

3. Repeat part (a) in the file `coordinates_dicts.py`. These functions use dictionaries instead of tuples to represent points. Again, `cart2sphere()` has been implemented for you. For cos^{-1}, use `math.acos()`; for $\sin^{-1}$, use `math.asin()`; and for $\tan^{-1}$, use `math.atan2()`. (10 points)

4. Implement `detect_type()` in the file `coordinates_dicts.py`. This function determines what type of coordinate system is being used based on the keys in the dictionary. (5 points)

We have included a file `grader_part2.py` to help you test your code. See file for instructions on use. Again, do not modify this file in any way or attempt to "hard code" answers to these tests in your functions.