30 points total. 70+% correctness (21+ points) is needed to pass. Remember: you must pass all assignments to pass the class.

The starter code for this assignment is available in the zip file `hw2.zip`, available on on the course web site.

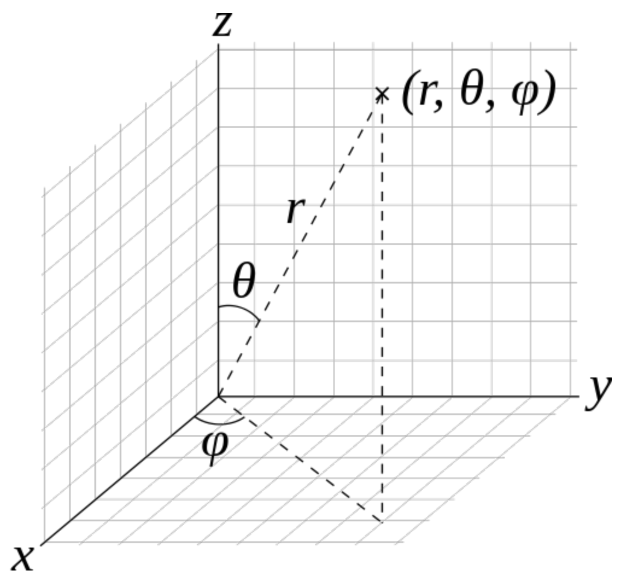
1. **Coordinate systems in three-dimensions**

   In class, we saw how to use a dictionary to store cartesian and spherical coordinates. In this homework assignment, we will write code that converts data types between different three-dimensional coordinates systems. First, we will review three coordinate systems.

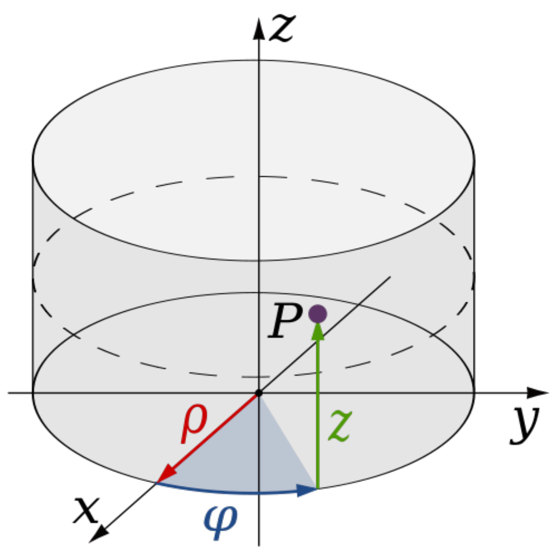
   The first system is the familiar cartesian coordinate system. Three points, call them x, y, and z, are used to represent the point on three orthogonal coordinate axes.

   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.



(a) Spherical coordinates



(b) Cylindrical coordinates

Figure 1

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

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

(a) 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. (9 points)

(b) 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)

(c) 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()`. (9 points)

(d) 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)

Grading:

To grade this question, tests will be conducted by calling the functions you implement. Each test is worth 0 points (do not pass test) or 1 point (pass test). There are 30 total tests.

Gaming the autograder by hard-coding the answers of the provided test functions is considered cheating and a violation of the Stanford honor code. I will be testing your code with additional (unreleased) tests to ensure that there is no cheating.

2. **NumPy and SciPy**
   In the next lecture, we will start with NumPy and SciPy, the two main Python libraries for scientific computing. For this part of the assignment, you will get introduced to these libraries. See course website for instructions on how to install/access these libraries.

   (a) Run the `dot()` example from http://wiki.scipy.org/Numpy_Example_List_With_Doc. (0 points).

   (b) Run the `integrate.quad()` example from http://wiki.scipy.org/scipy_Example_List. (0 points).

3. **Generator power (bonus: instructional only - not graded)**
   In this bonus question, we explore generator functions. Consider the following function, which takes as input a set of x coordinates, y coordinates, and z coordinates and returns all possible (x, y, z) points from the sets:

```
def triples(xpoints, ypoints, zpoints):
    return [(x, y, z) for x in xpoints for y in ypoints for z in zpoints]
```

   (a) Describe a potential memory hazard with this function.

   In Python, we can alternatively choose to "yield" a value. This allows us to iterate over objects without constructing the entire object. These types of functions are called generating functions. For example, the following function is the generator analog to `triples()`:

```
def triples_gen(xpoints, ypoints, zpoints):
    for x in xpoints:
        for y in ypoints:
            for z in zpoints:
                yield (x, y, z)
```

   (b) What gets printed with the following code? Explain.

```
for triple in triples_gen([1, 2], [2, 3], [4]):
    print triple

print triples_gen([3, 4], [5], [7, 8])
```

   (c) Write a function `keyvals`, which takes as input a dictionary and returns a list of (key, value) tuples of all key-value pairs in the dictionary.

   (d) Write a function `keyvals_gen`, which is the generator function analog of `keyvals` from part (c).