Assignment 2 (Due COB Monday 12th September):

Background:

The *gradient* is a vector that points always in the direction of a function's maximum increase. For a three-dimensional function, the gradient is

$$\nabla f = \langle \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \rangle \tag{1}$$

A *unit vector* that points always "downhill" is therefore

$$\boldsymbol{d} = -\frac{\nabla f}{|\nabla f|} \quad . \tag{2}$$

Assignment Task

Build a *Direction Set Search* method called **gradsearch** which finds the minimum of a function of three variables by always searching downhill. The code should have the following characteristics:

1. At each step of the iteration the *numerical* derivative is used to compute the "downhill direction vector" **d** in (2) above (This means you will have to compute each of the partial derivatives numerically, using the code that you already have).
2. The search direction at each step is given by the vector **d**. (This means that you only have to keep one direction vector in the direction set at a time).

The user should be able to call the function in the following way:

```
import [your initials]_search as [your initials]

x = [your initials].gradsearch(f,x0,tol)
```

where:

f(x): any function that accepts a 3D numpy array x = np.array([x[1],x[2],x[3]) as its input and returns a real number

x0: an initial estimate for the solution (also a 3D numpy array)

tol: the desired tolerance in x, defined as the square root of the summed squares of the components of the array x. That is, for the n^{th} step:

$$\text{tol} = \sum_{i=1}^{3} \sqrt{(x[1]_n - x[1]_{n-1})^2 + (x[2]_n - x[2]_{n-1})^2 + (x[3]_n - x[3]_{n-1})^2}$$

The solver should return an error if it does not converge within 1000 iterations, or if the partial derivatives become too small.

Submission: IMPORTANT

It is anticipated that your code will contain more than one python script. You should put all the scripts necessary to run your search function into a single ".zip" file and upload this to canvas.

Grading

The code will be graded according to the following scale:

Effectiveness (i.e. whether it works on some standard tests):	60%
Overall "elegance" (i.e. how easy the code is to read):	10%
Comments (whether they are comprehensible):	30%

BONUS POINTS will be awarded if your code deals with N-variable functions, rather than just 3. (This will be up to 15%, but you can't get more than 100% for the assignment).