Assignment

1. Write a function that will calculate and approximation of the definite integral of an arbitrary function. The function should be in a file called `integrate.py`, and look like this:

   ```
   i = integrate(f, a, b)
   ```

   This should return the definite integral of the function f, from a to b. You should also give the function an optional argument that allows the caller to specify the size of the intervals used to calculate the integral:

   ```
   i = integrate(f, a, v, 0.001)
   ```

2. Pick a definite integral. Plot the absolute value of the error (the difference between the actual definite integral, which you calculate by hand, and the estimate returned by your function) as a function of the size of the intervals (the optional last argument).

3. Write a function that estimates the area of arbitrary shapes, defined by an implicit function. Your code should be in a file called `area.py`, and should be called like this:

   ```
   a = area(f, p1, p2)
   ```

   Where f is the implicit function, and `p1` and `p2` define the corners of a bounding box around the area. `p1` and `p2` should be tuples, `(x, y)`. Give the function an optional fourth argument, to specify the number of samples used in the approximation:

   ```
   a = area(f, (0, 0), (2, 2), 1e6)
   ```

4. Pick a shape. Plot the absolute value of the error (the difference between the actual area, which you calculate by hand, and the estimate returned by your function) as a function of the number of samples (the optional last argument).

Thoughts

1. You should use a Riemann sum (Links to an external site.) to calculate the definite integral in the first question. There are several flavors of this, and you're free to pick any of them. We talked about this technique in class; if you're unsure of how to implement it, then get in touch with the instructor.

2. For the purposes of this assignment, an implicit function is one what evaluates to zero on the boundary of a shape, less than zero inside the shape, and greater than zero outside of the shape. For example, for a circle, the standard equation is x^2 + y^2 = r^2. The implicit form of this equation is x^2 + y^2 - r^2 = 0. For any points inside the circle, x^2 + y^2 - r^2 will evaluate to less than zero. For any points outside, it will evaluate to greater than zero. If the point is on the boundary, then it will evaluate to zero.

3. You should use Monte Carlo integration (Links to an external site.) for calculate the area in the second question. Again, there are several ways to do this, and you're free to pick the one you want. I'd advise starting with a simple one, using rejection sampling (Links to an external site.). The basic idea of this is to generate a sample point within the bounding box of the area. Count the number of times that this point lies within the area. The proportion of points that end up inside then lets you calculate the area, based on the area of the bounding box. We talked about this in class; if you're unsure of how to implement it, then get in touch with the instructor.

4. A good test case for the integration algorithm is the a circle with unit radius, centered at the origin. The return value of the function call

   ```
   area(lambda x,y:x*x + y*y -1, (-1, -1), (1, 1))
   ```

   should be an approximation of pi.

Grading

1. 40 points for a working implementation of the function to calculate a definite integral. 10 points for the graph.

2. 40 points for working implementation of the function to calculate the area of a shape. 10 points for the graph. Call the tarball `username-hw1.tgz`. When it's untarred, there should be a directory called `hw1` with all the necessary files in it.

What to Hand In

Hand in `integrate.py`, `area.py`, and any code needed to draw the graphs. Also include a file called `README` that gives the graders instructions on how to run the code that draws the graphs.

The Rules

Everything you do for this lab should be your own work. Don't look up the answers on the web, or copy them from any other source. You can look up general information about Python on the web, but no copying code you find there. Read the code, close the browser, then write your own code.