Name: Evan Leglar	**Lab Time** T 12:00 PM

Names of people you worked with:
• N/A

Websites you used:
http://code.activestate.com/recipes/577263-numerical-integration-using-monte-carlo-method/ https://matplotlib.org/ https://stackoverflow.com/

Approximately how many hours did it take you to complete this assignment (to nearest whole number)?	6

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.

By writing or typing your name below you affirm that all of the work contained herein is your own, and was not copied or copied and altered.

Evan Leglar

Note: Failure to sign this page will result in a 50 percent penalty. Failure to list people you worked with may result in no grade for this lab. Failure to fill out hours approximation will result in a 10-percent penalty.

Turn .zip files of Python code to Canvas or your assignment will not be graded

BEFORE YOU BEGIN

1) Make a new project for this lab
2) Make integrate.py
3) Read up on Riemann integration
4) Read up on Monte Carlo sampling for integration. There are several sources out there; I suggest googling and finding one that matches your level of understanding of statistics. There are multiple ways to use Monte Carlo sampling to do integration; the simplest is to randomly sample x and take the mean of the f(x) values. More complicated versions (rejection sampling) put a box around the function then randomly generate both x and y values, then add up the number of x,y pairs that were under the curve.

Learning Objectives:

You should be able to do the following:

- Pass functions to functions
- Use tuples as a data structure
- Use numpy
- Use random numbers

Thoughts (come back and read these after you've read the problems):

1. There is a function to do integration in numpy. You should not use it in your function, but you should consider using it for your tests.

2. You should use a Riemann sum 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 briefly in class.

3. You should use Monte Carlo integration for calculate the definite integral in the second question. You'll need to do what's called rejection sampling (putting a box around the function and calculating inside versus outside). Start with a function that is all positive, then try it with a function that is both positive and negative.

Grading Checkpoints

1. `integrate()` produces the correct values (2 points) and implements all other specifications (a > b, and default arguments) (1 point)*. [3 points total]

2. `integrate_mc()` produces plausibly correct values for a positive function [2 points]*, a more complicated function [1 point]*, and implements bounds correctly [1 point]*. [4 points total]

3. `estimate_pi()` produces a value near pi (1 point)* and exhibits increased accuracy as we increase n (1 point)*. [2 points total]

4. Your plot exhibits the expected properties for the Riemann and Monte Carlo lines, as well as showing the ground truth (2 point). Also has labelled axes and a title (1 point). [3 points total]

All parts except for number 4 will be graded automatically by Gradescope with immediate feedback. 4 will be graded manually.

Hand in all files to Gradescope.

Problem 1 Riemann sum

Put all code in a file called `integrate.py`.

Write a function called `integrate` which takes in 4 values: `f`, `a`, `b` and an optional `n` value (which defaults to 100). `a` and `b` are floats representing the starting and ending bounds of integration, while `f` is a *function reference* which takes in a single argument and returns a number.

It should return the definite integral of the function `f` from `a` to `b`, computed by using a *left-sided* Riemann sum with `n` intervals.

Your function should also be able to handle the case when a >= b, in which case it should return the negative of the integral from b to a.

Comments for grader/additional information (if any)

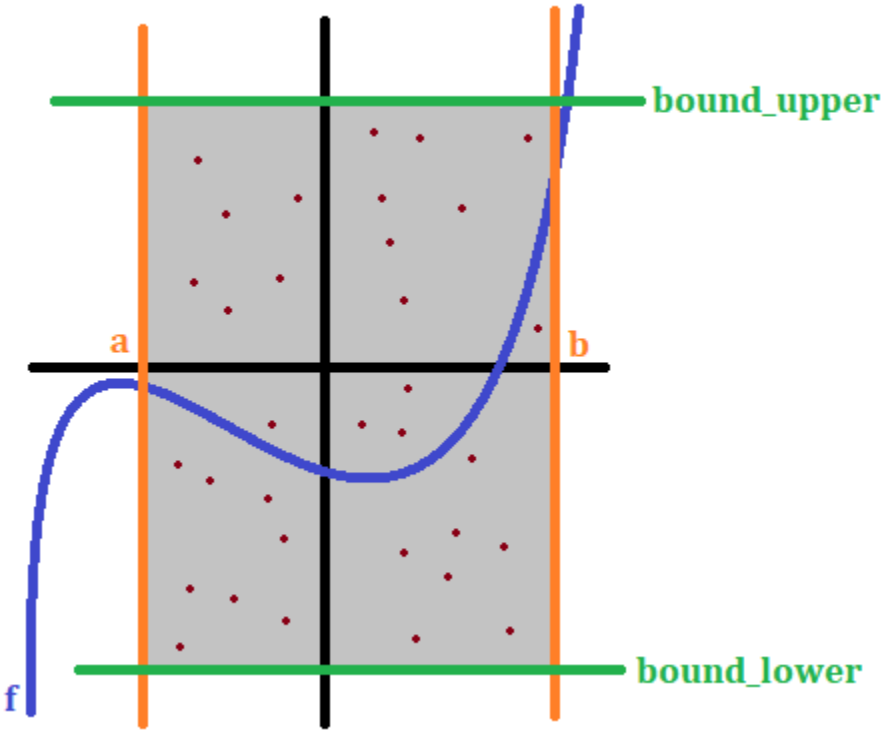
Problem 2 Monte Carlo integration

Write a function called `integrate_mc` which takes in 5 arguments: `f`, `a`, `b`, `bound_lower`, `bound_upper`, and an optional argument `n` which defaults to 100.

It should compute an approximation of the function `f` from `a` to `b` by using *Monte Carlo integration*, i.e. by sampling `n` points in the range [a, b] x [bound_lower, bound_upper] and using the function value at the corresponding x-value to determine if that point is "contained" by the function.

As before, it should handle the case of a > b. If bound_lower > bound_upper, raise a ValueError. You can assume that bound_lower <= 0 and bound_upper >= 0.

A quick visual showing the bounds and the process of sampling points:



Hint: First, think about how to make your code work on functions that are positive valued. Then think about how you would modify it to make it work on a negative-valued function. Then think about how to handle both at the same time.

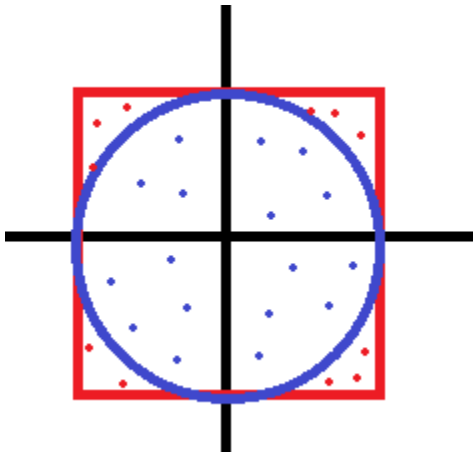
Note: It's possible for your bound_upper and bound_lower to "cut off" your function. This is perfectly fine; essentially then we're simply estimating the integral of a bounded version of f.

Comments for grader/additional information (if any)

Problem 3 Probabilistic estimate of pi

Write a function `estimate_pi` which takes in a single argument, `n`. It should then return an estimate of pi by uniformly sampling `n` points on the square [-1, 1] x [-1, 1] and returning the area of the square, normalized by the percentage of points which fall inside the unit circle.

See below for a visualization. Out of all the points, the blue points are the ones which "positively" contribute to the area of the circle.



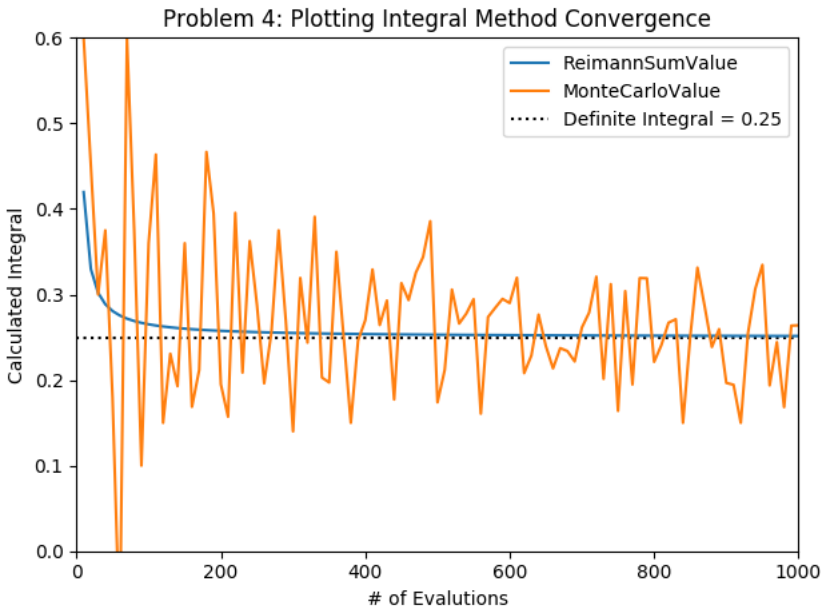
Comments for grader/additional information (if any)

Problem 4 Plot integral convergence

Consider the function f(x) = x^3 + 2x – 1. **Write some code to create a plot** where the x-axis represents the number of evaluations and the y-value shows the prediction of the integral of f(x) from 0 to 1 for both your Riemann sum function and your Monte Carlo function. Do this for up to n=1000 in steps of 10.

You should also compute the true value of the definite integral by hand and plot it as a **dashed horizontal line** on your plot. Be sure to add a title and labelled axes. Paste your plot here.

Plot



Comments for grader/additional information (if any)

MonteCarlo integral values vary greatly with each iteration of the calculation. General trend for montecarlo method is to incrementally calculate integral closer to definite answer. The reimann sum method approaches the definite answer much quicker and consistently.