hw3.sagews

April 14, 2014

Contents

1 Homework 3- Math 480b - Spring 2014

Due Friday, April 18, 2014 by 6pm

Turn it in by creating a folder called homework3 in your project, with this worksheet in it. NOTE that youll find this worksheet in homework3 in the first place so just leave it there. It will be automatically collected sometime after 6pm on Friday, April 18.

1.1 Problem 1:

- Create two lists v and w that each consist of 60 distinct randomly chosen (in a way you decide) integers between 1 and 100, inclusive.
- Sort the two lists v and w.
- Intersection: Compute the list of integers that are in both v and w
- Union: Compute the list of integers that are in either v or w.
- Difference: Compute the list of integers that are in v but not in w.
- Evens: Compute the list of even integers that are in v.
- Odds: Compute the list of odd integers that are in w.
- Plot: Draw a plot that has a 60 dots in it, one at each point (x,y)=(v[i],w[i]).

1.2 Problem 2: Start working on your project

Come up with at least three plausible ideas for a project and describe each by giving a title and 3-sentence description. These ideas could involve adding some functionality to Sage, fixing bugs, or just applying Sage to solve a problem. You can find inspiration at

- past courses `http://wstein.org/courses/`
- the trac tickets `http://trac.sagemath.org/wiki/TicketReports`
- the Sage documentation `http://sagemath.org/help.html`

1.3 Problem 3: Create a Python Class

Create your own math-related Python class that models something interesting to you. It should at least have the following methods:

__repr__, __add__, __sub__, __neg__, __init__

In your solution show how to create an instance of your class.

1.4 Problem 4: Implement a Python Decorator Class

This is a two-part problem, to design a class and use it as a decorator.

First, well make a new class called file_logger. We want to be able to use this as a decorator, as follows: my_logger = file_logger("some_filename") @my_logger def f(n, m): do_stuff() return something_interesting

Now every call to f will act just like normal except that it will log all inputs and outputs to the file some_filename.

This should be implemented as a Python class with at least two methods: __init__ and __call__. The __init__ method should take two arguments: self (which methods on a class almost always take), and filename, which is the file where it should log inputs and outputs. The __call__ method should take a single argument, a function, and should return a new function that takes the same arguments (hint: *args, **kwds) and logs all inputs and outputs to the file filename.