PyCodeJam Documentation

Release 1.0.0

Jon Eisen

April 06, 2014

CHAPTER 1

Introduction

This is PyCodeJam, a CodeJam runner in python!

Essentially, this package will save you time vital to CodeJam problems.

Can I used this in the contests? Yes, as long as this library is open source (it is) and is openly available online (it is), it can be used without submission on each problem. Just make a comment in your code to that effect. See the Code Jam FAQ

License: pycodejam is licensed with the MIT license found in the LICENSE file.

1.1 Installation

```
pip install pycodejam
```

Or you can install from source:

```
git clone https://github.com/yanatan16/pycodejam
cd pycodejam
python setup.py test && python setup.py install
```

pycodejam is now compatible with python 2.6+ and 3.x!

1.2 Examples

There are also some examples in the [examples folder](http://github.com/yanatan16/pycodejam/tree/master/examples).

A simple example:

```
def solve(line1, line2):
  return sum(line1) - sum(line2) # This is where you put your solution

if __name__ == "__main__":
  from codejam import CodeJam, parsers
  CodeJam(parser.ints, solve).main()
```

A complex example:

```
def solve(*lines):
  return sum((sum(line) for line in lines)) # This is where you put your solution
```

```
@parsers.iter_parser
def parse(nxtline):
  n = int(nxtline())
  return [int(nxtline()) for unused in range(n)]

if __name__ == "__main__":
  from codejam import CodeJam
  CodeJam(parse, solve).main()
```

The command line *-h* option:

```
usage: problem_solver.py [-h] [-o FILE] [-d] [-q] [-m] [-w N] FILE

Run a Generic CodeJam Problem.

positional arguments:
  FILE                  input file (A-small-practice.in for example)

optional arguments:
  -h, --help            show this help message and exit
  -o FILE, --output FILE
                        output file (defaults to input_file.out)
  -d, --debug           Add some debug output
  -q, --quiet           Quiet all output to stdout
  -m, --multiproc       Enable multiprocessing
  -w N, --workers N     Number of workers to use if multiprocessing
```

CHAPTER 2

In Depth API

2.1 codejam Module - Main Class

class codejam.codejam.**CodeJam**(*parser*, *solver*, *name='Generic CodeJam Problem'*, *floating_accuracy=6*, *include_case=True*)

A class that provides ways to easily run a code jam problem

The class requires two parameters to instantiate: parser - A generator function of one parameter (file_obj) that yields each case in a tuple. There are predominant parsers and helpful decorators in the parsers module solver - A solver that takes the case tuple expanded and returns a str()-able object to print as the answer

The usual way to use this class is to call the main() function which will interpret command line arguments for the input file and options for debugging (-d)

name - Something to add to the help statement floating_accuracy - digits of precision for floating point numbers include_case - Include the "Case #n: " for each case

main(*argv=None*)

Run the CodeJam runner utility from the command line

run(*inf*, *outf*, *debug=False*, *silent=False*)

Run the CodeJam runner with file objects inf and outf and options for debug and silent

run_multiproc(*inf*, *outf*, *debug=False*, *silent=False*, *workers=4*)

Run the CodeJam runner utility as multiprocessing with a default of 4 workers

2.2 parsers Module - Input File Parsers

Code Jam Problem Parsers

Author: Jon Eisen Dec 2012

codejam.parsers.**floats**(*f*)

Simple parser that returns all floats as an array of arrays

codejam.parsers.**ints**(*f*)

Simple parser that returns all integers as an array of arrays

codejam.parsers.**iter_parser**(*fn*)

A decorator that will pass a function to return the next line of input. The decorated function will be called for each case and should return (not yield) the case tuple

codejam.parsers.**lines**(*f*)
Simple parser that returns each line as an array

codejam.parsers.**simple_parser**(*parse*)
Simple parsers have a constant number of lines per case. This decorator implies that number and calls the decorated function with an expanded list of lines

codejam.parsers.**split_cast_parser**(*cast*)
A split_cast_parser is a simple_parser that splits each line into words then casts them

codejam.parsers.**words**(*f*)
Simple parser that returns all words as an array of arrays

2.3 helpers Module

Code Jam Problem Helpers

Author: Jon Eisen Dec 2012

codejam.helpers.**memoize**(*fn*)
Memoize a function

C