Mathdeck Documentation

Release 0.1.1-planning

Patrick Spencer

June 10, 2016

Contents

Mathdeck is a program suite for managing the computations involved in writing displaying, and grading mathematical based homework problems. It is written to abstract the computations so that other responsibilities such as managing users and grades can be left to more apt systems such as online course management systems.

The program is written with one central idea: problem files are plain python modules. This makes it so you can use plain python and all the scientific python libraries such as scipy, numpy, matplotlib, etc. in problem files.

CHAPTER 1

Quick tour

```
# problem-library/example1/__init__.py
from mathdeck import Answer

ans1 = Answer(value=5)
ans2 = Answer(value=101)
_answers = [ans1, ans2]
```

This makes two answers of value 5 and 101 respectively and loads into the mathdeck hook `_answers`. The answers can be accessed by the `answers` attribute in an instance of a problem file loaded by `Problem(module_path)`:

```
>>> from mathdeck import Problem
>>> problem = Problem('problem-library/example1')
>>> print(problem.answers)
[<mathdeck.Answer>, <mathdeck.Answer>]
>>> print(type.answers[0])
<mathdeck.Answer>
>>> print(type.answers[0].value)
5
>>> print(type.answers[1].value)
101
```

We can check the answer against a predefined answer. 0 means not a match and 1 means is a match.

```
>>> from mathdeck import Problem
>>> problem = Problem('problem-library/example1')
>>> print(problem.answers)
5
>>> problem.check(problem.answers[0], Answer(value=4))
0
>>> problem.check(problem.answers[0], Answer(value=5))
1
>>> problem.check(problem.answers[1], Answer(value=101))
1
```

Example problem file

```
# problem-library/example2/__init__.py
from mathdeck import Problem, Answer

ans1 = Answer(value=5)
a = (ans1.value)*3

_answers = [ans1]
_template_vars = {'temp_var': a}
```

```
>>> from mathdeck import Problem
>>> problem = Problem('problem-library/example1')
>>> print(problem.answers)
[<mathdeck.Answer>]
```

Let's make a couple templates so we know how to display the question. Mathdeck uses Jinja2 templates for its templating system.

```
<!-- problem-library/example2/templates/default.jinja2 -->
What is {{ temp_var }} divided by 3?
```

```
<!-- problem-library/example2/templates/template2.jinja2 -->
What is {{ temp_var }} times 6 divided by 3 and then divided by 2?
```

Our module file directory looks like this now:

```
# problem-library/example2/
__init__.py
templates/
-- default.jinja2
-- template2.jinja2
```

```
>>> from mathdeck import Problem

>>> problem = Problem('problem-library/example1')
>>> print(problem.answers)
[<mathdeck.Answer>]
>>> ans = problem.answers[0]
>>> print(ans.value)
5
>>> print(ans.type)
Integer
>>> problem.display()
What is 15 divided by 3?
>>> problem.display(template=template2.jinja2)
What is 15 times 6 divided by 3 and then divided by 2?
```

A more advanced problem file using sympy, the display function, the check function and seed values would look something like this:

```
# problem-library/example3/__init__.py
from mathdeck import Problem, Answer
from mathdeck.helpers import random
from sympy import var, expand, latex

# choose two random integers between 0 and 10. The second should not
# be the same as the first
root1 = random.randrange(0,10)
root2 = random.randrange(0,10).neq(root1)

# specify our variables
var(x)

# define a polynomial whose roots are root1, root2, and root3 and
# expand it
p = (x-root1)*(x-root2).expand()

ans1 = Answer({
  'value': root1,
```

```
  'type': 'number',
  'domain': 'real',
  'label': 'first_ans'
})
ans1 = Answer({
  'value': root2,
  'type': 'number',
  'domain': 'real',
  'label': 'second_ans'
})

_answers = [ans1, ans2]
_template_vars = {'poly': p}
```

```
<!-- problem-library/example3/templates/default.jinja2 -->
Find the roots of the polynomial $p(x) = {{poly | format=latex}}$:
```

When `mathdeck.Problem.loadproblem` is called it uses a seed value to make sure `mathdeck.random` is predictable by the computer. Say a seed value of 20 makes `root1 = 1` and `root2 = -1`

```
>>> from mathdeck import Problem
>>> problem = Problem('problem-library/example1')
>>> problem.display(seed=20)
Find the roots of the polynomial $p(x) = x^2-1$
>>> problem.check({'first_ans': -1, 'second_ans': 3}, seed=20)
# first_ans is correct but second_ans is incorrect
{'first_ans': 1, 'second_ans': 0}
>>> problem.check({'first_ans': -1, 'second_ans': 1}, seed=20)
```

CHAPTER 2

Contents

2.1 API

class mathdeck.**Problem**(*problem_file_path*)
this class is the main class Problem

Parameters **problem_file_path** – the location of the problem file that needs to be loaded relative to the default library location as set in mathconf.py file.

2.2 Problem Spec

This document outlines how a Mathdeck problem should be defined.

All mathdeck problems are python modules. They look like this:

```
# problem-library/example1
__init__.py
mathdeck_meta_data.py
templates/
-- default.jinja2
```

2.2.1 Meta data

All problem files should have a file called mathdeck_meta_data.py. A sample meta data file will look like this:

```
# problem-library/example1/mathdeck_meta_data.py
# ID should be unique somehow (need a way to do this)
ID = 0123456789
AUTHORS = [
{
    'name':Bob Hope,
    'institution': 'University of Missouri',
    'email': 'bhope@missou.edu',
    'edited': '2010-2012'
}
    'name': Bruce Wayne,
    'institution': University of Kentucky',
    'email': 'bwayne@uky.edu',
    'edited': '2013-2016'
}
```

```
MAJOR_CATEGORIES = ['Calculus', 'Diffeq']
MINOR_CATEGORIES = ['Slope of line', 'unique solutions']
```

All the meta data labels are capitalized because, in python, constants are capitalized.

Available meta data fields

`ID`: some id that is unique against all other problem files in the world. (maybe use an MD5 hash of the original version of the problem file to come up with this)

`AUTHORS`: a python list of dictionaries which keep tabs on the problem authors

2.2.2 Problem file hooks

These are hooks that mathdeck use to know about the problem when it loads the problem module.

_answers

`_answers`: a python list of answers. Each answer is an instance of the `mathdeck.Answer` class.

```
# problem-library/example1/__init__.py
from mathdeck import Answer

ans1 = Answer(value=5)
ans2 = Answer(value=101)
_answers = [ans1, ans2]
```

This makes two answers of value 5 and 101 respectively and loads into the mathdeck hook `_answers`. The answers can be accessed by the `answers` attribute in an instance of a problem file loaded by `Problem(module_path)`:

```
>>> from mathdeck import Problem
>>> problem = Problem('problem-library/example1')
>>> print(problem.answers)
[<mathdeck.Answer>, <mathdeck.Answer>]
>>> print(type.answers[0])
<mathdeck.Answer>
>>> print(type.answers[0].value)
5
>>> print(type.answers[1].value)
101
```

_template_vars

`_template_vars`: a python dictionary of variables that can be called in templates.

Say our problem module looks like this:

```
# problem-library/example2/
__init__.py
mathdeck_meta_data.py
templates/
-- default.jinja2
```

with

```
# problem-library/example2/__init__.py
from mathdeck import Problem, Answer

ans1 = Answer(value=5)
a = (ans1.value)*3

_answers = [ans1]
_template_vars = {'temp_var': a}
```

Mathdeck now knows that we can use a template variable called `temp_var` in templates like this:

```
<!-- problem-library/example2/templates/default.jinja2 -->
What is {{ temp_var }} divided by 3?
```

```
>>> from mathdeck import Problem

>>> problem = Problem('problem-library/example1')
>>> problem.display()
What is 15 divided by 3?
```

2.3 Install

Mathdeck is only supported on linux/unix operating systems right now.

To install mathdeck first copy `mathdeckconf.json.sample` to `/etc/mathdeck/mathdeckcong.json` and edit the attribute `prob_lib_dir` to the absolute path of where your problem library is e.g. `/var/problem_library`. Then run `python setup.py`.

CHAPTER 3

Features

- A system for writing homework problems.
- A system for displaying problem files with templates.
- A system for measuring how close a proposed solution is to a predefined solution and offering suggestions for improvement to lead a student in the right direction.

CHAPTER 4

Goals

- Clear API
- Clear documentation.
- Comprehensive test suite.
- Creating problems should be as easy as possible for people who are mathematically inclined and would be able to pick up the basics of python in a couple days.

CHAPTER 5

License

Mathdeck is licensed under the Apache 2.0 license. See LICENSE file for more details.

P