Homework 7
Unit Testing and Python

Due: Wednesday, March 7th, 11:59PM (Hard Deadline)

Submission Instructions

When you are done, submit a link to your GitHub repository here: https://goo.gl/forms/56zbjqh1YnnregSF3

For this assignment, we will build on the RPN calculator from lecture.

1 Continuous Integration

One of the neat elements of modern software development is *continuous integration*. CI technologies automatically run test suites throughout the development process and help prevent bugs from creeping in.

1.1 The Setup

While GitLab does support CI, it's not quite set up and working here at Michigan yet, so we'll do this next bit using GitHub instead. We'll use Travis CI as our CI platform.

1. As a first step, create accounts on both of these platforms – do GitHub first, Travis CI will use your GitHub account.
2. Next, create a new repository named **c4cs-w18-rpn** on GitHub.
3. Like submitting for attendance, we'll follow the directions from GitHub for "...or push an existing repository from the command line". *However*, we already have a remote named "origin", so we'll need to change the commands just a little:
   - `git remote add` **`github`** `https://github.com/`**`your-github-username`**`/c4cs-w18-rpn.git`
   - `git push -u` **`github`** `master`
4. Refresh the GitHub page in your browser – you should see your code!
5. Now that we have a repository, we need to enable Travis CI for this repository. Visit https://travis-ci.org/profile and enable Travis for the repository you just made.
   - You may need to click the "Sync Account" button in the top right if it doesn't immediately show up

Note: Final versions of `rpn.py` and `test_rpn.py` from lecture are on the course homepage (https://c4cs.github.io/).

1.2 The Part You Have to Figure Out

This is a complete `.travis.yml` file:

```
$ cat .travis.yml
language: python

python:
  - 3.5

script:
  - make test
```

You can also see a complete example live at https://github.com/ppannuto/c4cs-f16-rpn

Travis uses a file named `.travis.yml`[1] to figure out what it needs to do. You will need to create this file, commit it, and then run `git push` to push that commit to GitHub. Once you push, Travis will automatically start running.

It is very likely that Travis will error on your first try. That's fine. Just make some changes to `.travis.yml`, commit, and push again.

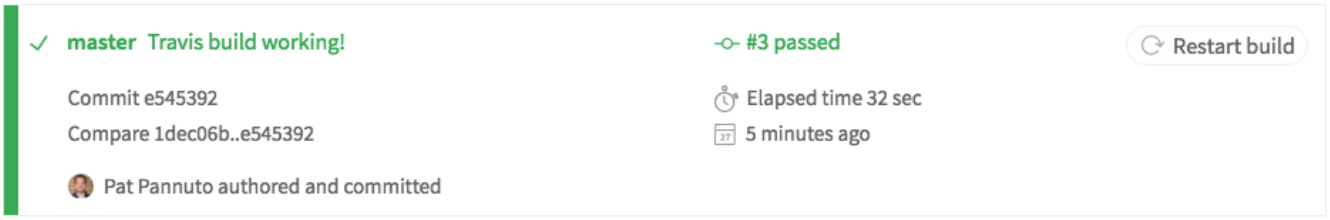
The things you'll have to tell Travis in your configuration file:

1. This is a Python project
2. This project requires Python3 (any version of Python3, but 3.5 is a good choice)
3. How to run your tests

A few links that may help:

- Wikipedia on YAML for a syntax primer: https://en.wikipedia.org/wiki/YAML
- Travis CI's getting started guide: https://docs.travis-ci.com/user/getting-started/
- Travis CI's guide for Python: https://docs.travis-ci.com/user/languages/python/
- Travis Lint: http://lint.travis-ci.org/
  - A *linter* is a tool that checks code for syntax and style problems. It validates that things are correct, make sense, and gives feedback on how to make your code better.
  - I'm really bad at writing YAML, the syntax has just never made sense to me, so I find this really useful

When you've got everything working, `https://travis-ci.org/`**`your-github-username`**`/c4cs-w18-rpn` should look something like this:



For this class, we're happy to do everything publicly. If you use GitHub and Travis CI for other class projects ***You MUST create PRIVATE repositories***.

You can do this for free as a student, grab a copy of the GitHub Student Developer Pack to get started.

For most EECS classes at Michigan, posting your code online (e.g. a public repository) is an honor code violation. You may be subject to failing the project, failing the course, or other penalties.

[1] Note the leading period for this file. Remember that makes it a *hidden file*. This means it won't show up if you just run `ls`. You have to run `ls -a` (think "all") to see it. Other than this little quirk, hidden files are like any other. You can edit them with any text editor, no fanciness.

DO NOT PUT CODE FOR OTHER EECS CLASSES IN PUBLIC REPOSITORIES

2 Let's do some test-driven-development

2.1 Make the test

Add a test to your test suite for the exponentiation operator (the carat: ^). Once you have finished your test, commit and push your test. Verify that your Travis build **fails** (you have not implemented carat support yet!). Fix your Travis setup if you need to.

```
commit 666df1ae474e743a6ea344de563eb024060181a0
Author: Pat Pannuto <pat.pannuto(at)gmail.com>
Date:   Wed Oct 19 23:26:47 2016 -0400

    Add exponentiation test (warn: not implemented)

diff --git a/test_rpn.py b/test_rpn.py
index 9f8e6b9..72f688b 100644
--- a/test_rpn.py
+++ b/test_rpn.py
 -15,6 +15,9  class TestBasics(unittest.TestCase):
        def test_divide(self):
                result = rpn.calculate("6 3 /")
                self.assertEqual(2, result)
+       def test_exponentiation(self):
+               result = rpn.calculate("3 4 ^")
+               self.assertEqual(81, result)
        def test_badstring(self):
                with self.assertRaises(TypeError):
                        rpn.calculate("1 2 3 +")
```

2.2 Add the implementation

Add support to your RPN calculator for exponentiation. Commit, push, and verify that your CI build is "green" (the tests pass).

```
commit 42d8fb112b179b48dafcfc6e83049cd9ccc1ba8f
Author: Pat Pannuto <pat.pannuto(at)gmail.com>
Date:   Wed Oct 19 23:28:42 2016 -0400

    Implement exponentiation (tests clear)

diff --git a/rpn.py b/rpn.py
index 5b7bbe5..90dba8d 100644
--- a/rpn.py
+++ b/rpn.py
 -8,6 +8,7  operators = {
        '-': operator.sub,
        '*': operator.mul,
        '/': operator.truediv,
+       '^': operator.pow,
 }

 def calculate(myarg):
```

Submission Instructions

When you are done, submit a link to your GitHub repository to the link given at the beginning of the homework.

The solution repository also includes these changes: https://github.com/ppannuto/c4cs-f16-rpn