Metadata

```
Course:  DS 5100
Module:  09 Python Packages
Topic:   Challenge -- Create a Package
Author:  R.C. Alvarado
Date:    10 July 2022
```

Challenge

Create and publish on GitHub a package containing a single module with a single function.

Recipe

1. **Create a new git repo for your package**.
2. **Write a module file** that contains at least one function.
3. Create and edit the **required files and directories** for your package and move your module there.
4. **Stage, commit, and push** all the files you've created.
5. **Install** your package with `pip`.
6. Outside of your package dir, write a script to **test your method**.

Create Repo

Create a single new repo for your package on GitHib.

Be sure to choose a license, include a readme, and a select a `.gitignore` file for Python.

If working in pairs, Invite your partner to the repo. Both members clone the repo locally -- outside of your course repo.

The root of the repo will be the root of your package.

Create the Module

Your module just needs to have one function.

It's a good idea to make it print something.

It should also have a docstring.

Name the file something that reflects the purpose of the function, and any other code that the file will contain.

in the file `jabber.py`:

```
def tell_me_something():
    """Wise cracking function."""
    print("Something.")
```

Create Package Directory and Files

Create your directory and file structure for your package.

Include essential files.

Move your module into it.

The initial directory structure your package should look like this:

```
your_package/
    your_module.py
    __init__.py
setup.py
```

My example looks like this (I named the package `demo`):

```
demo/
    jabber.py
    __init__.py
setup.py
```

Edit `__init__.py`

Put a print statement welcoming the user to your module.

Inside of `__init__.py`:

```
print("Welcome to my module!")
```

Create a setup file

Create the `setup.py` file.

Import what needs to be imported.

Add basic metadata and configuration info about your package.

Inside of `setup.py` I put:

```
from setuptools import setup

setup(name='Demo',
      version='0.1',
      description='A simple demonstration package',
      url='http://github.com/ontoligent/m09-demo',
```

```
      author='R.C. Alvarado',
      author_email='ontoligent@gmail.com',
      license='MIT',
      packages=['demo'])
```

Install your package

Run a command from bash to install the script to Python on your system.

We normally run this during development:

```
pip install -e .
```

Here are the results:

```
rca2t@rivanna$ pip install -e .
Defaulting to user installation because normal site-packages is not writeable
Obtaining file:///sfs/qumulo/qhome/rca2t/Documents/MSDS/DS5100/m14-demo
Installing collected packages: Demo
  Running setup.py develop for Demo
Successfully installed Demo
```

We might also have done this:

```
pip install --user -e .
```

Test it out

Run the file somewhere on your system outside of the repo.

You can use a python file, a notebook, or the command-line.

Here is a sample script:

```
#! /usr/env python3

from demo.jabber import tell_me_something as tms

tms()
```

And here is the output:

```
rca2t@rivanna$ python ./m14-demo-test.py
Welcome to Jabber!
Something.
```

Really Test It

Clone the repo on another computer.

Install your package.

Test it out.

Try importing the function in `__init__.py`

Add an import line to `__init__.py` to preload the function.

In my case, I put this at the top of `__init__.py`:

```
from .jabber import tell_me_something
```

I could have also put:

```
from demo.jabber import tell_me_something
```