Assignment 2 – comp30670 – Daniel O'Byrne - 17205389

Q1

Cookicutter

- Once I successfully installed cookiecutter, it was a simple set up – the cookiecutter command with a git repository as arguement, prompts the user to enter project and personal details. Command prompt entries make it easy and efficient.

```
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~$ cookiecutter$ cookiecutter https://github.com/audreyr/cookiecutter-pypackage
No command 'cookiecutter$' found, did you mean:
 Command 'cookiecutter' from package 'cookiecutter' (universe)
cookiecutter$: command not found
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~$ cookiecutter https://github.com/audreyr/cookiecutter-pypackage
full_name [Audrey Roy Greenfeld]: Daniel O'Byrne
email [aroy@alum.mit.edu]: daniel.obyrne@ucdconnect.ie
github_username [audreyr]: obyrned1
project_name [Python Boilerplate]: Assignment2CC
project_slug [assignment2cc]:
project_short_description [Python Boilerplate contains all the boilerplate you need to create a Python package.]: Project for a
ssignment 2 in comp30670
```

- The resulting tree structure of the project:

```
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670/assignment2cc$ tree
.
├── assignment2cc
│   ├── assignment2cc.py
│   ├── cli.py
│   └── __init__.py
├── AUTHORS.rst
├── CONTRIBUTING.rst
├── docs
│   ├── authors.rst
│   ├── conf.py
│   ├── contributing.rst
│   ├── history.rst
│   ├── index.rst
│   ├── installation.rst
│   ├── make.bat
│   ├── Makefile
│   ├── readme.rst
│   └── usage.rst
├── HISTORY.rst
├── LICENSE
├── Makefile
├── MANIFEST.in
├── README.rst
├── requirements_dev.txt
├── setup.cfg
├── setup.py
├── tests
│   ├── __init__.py
│   └── test_assignment2cc.py
└── tox.ini
```

- I came across no issues in configuring my project with cookiecutter. It provides an easy and efficient way of structuring projects with ease of use, as it prompts the user for details.
- I would certainly use this tool for future python projects. A very straight-forward tool, which automatically establishings such things as a README and a setup.py file.

Python Project Template

- Firstly I had to clone the Python Project Template from the GitHub repository provided, to the desired folder (assignment2):

```
1264  git clone https://github.com/seanfisk/python-project-template.git assignment2
```

- Within this assignment2 folder, a metadata.py file was created, where I opened this and entered data describing my project:

```python
# -*- coding: utf-8 -*-
"""Project metadata

This project is for assignment 2, software engineering, comp30670
"""

# The package name, which is also the "UNIX name" for the project.
package = 'my_module'
project = "Assignment2"
project_no_spaces = project.replace(' ', '')
version = '0.1'
description = 'It does cool things'
authors = ['Daniel O Byrne']
authors_string = ', '.join(authors)
emails = ['daniel.obyrne@ucdconnect.ie']
license = 'MIT'
copyright = '2018 ' + authors_string
url = 'http://ucd.ie/'
```

- When the command ‘python internal/generate.py’ was run, it had generated files based on the metadata I had just entered.
- I subsequently deleted the old git repository and initialized the new one.

```
rm -rf .git # or `ri -recurse -force .git' for PowerShell
git init
```

- I came across this error when running ‘paver test_all’ (I renamed the folder from assignment2 to assignment2PPT):

```
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670/assignment2PPT$ paver test_all
/home/obyrned1/anaconda3/envs/comp30670/lib/python3.6/distutils/dist.py:245: UserWarning: 'licence' distribution option is deprecated; use 'license'
  warnings.warn(msg)
usage: paver [global_opts] cmd1 [cmd1_opts] [cmd2 [cmd2_opts] ...]
   or: paver --help [cmd1 cmd2 ...]
   or: paver --help-commands
   or: paver cmd --help

error: invalid command 'test_all'
```

```
UserWarning: 'licence' distribution option is deprecated; use 'license'
```

- I replaced the ‘licence’ with ‘license’ in the file that the command points at:

```
/home/obyrned1/anaconda3/envs/comp30670/lib/python3.6/distutils/dist.py:245:
```

  - However, a similar error presided, thus I never ran the ‘paver test_all’ successfully.
- Regardless of failing this test, the PPT set up the following file structure:

```
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670/assignment2PPT$ tree
.
├── docs
│   ├── make.bat
│   ├── Makefile
│   └── source
│       ├── conf.py
│       ├── index.rst
│       ├── README
│       └── _static
├── LICENSE
├── MANIFEST.in
├── my_module
│   ├── __init__.py
│   ├── main.py
│   ├── metadata.py
│   └── __pycache__
│       └── metadata.cpython-36.pyc
├── pavement.py
├── README.rst
├── requirements-dev.txt
├── requirements.txt
├── setup.py
├── tests
│   └── test_main.py
└── tox.ini
```

- The only issue I found in the installation of my project using PPT was realted to the paver test_all command, even after adjusting the licence issue.
- Besides this it was quite an easy set up, and I would use the tool again. Entering details into the metadata.py file and creating the files from this data made sense. However this process was arguably less convenient then the command prompt equivalent used by Cookiecutter.

Pyscaffold

- Firstly I had to install Pyscafold:

```
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670$ pip install pyscaffold
```

- I then created a file for the contents of the pyscaffold project to be put into, using putup:

```
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670$ putup assignment2pyscaffold
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670$ ls
assignment2cc  assignment2PPT  assignment2pyscaffold  README.md
```

- This created the following file structure (project template) in this folder:

```
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670/assignment2pyscaffold$ tree
.
├── AUTHORS.rst
├── CHANGELOG.rst
├── docs
│   ├── authors.rst
│   ├── changelog.rst
│   ├── conf.py
│   ├── index.rst
│   ├── license.rst
│   ├── Makefile
│   └── _static
├── LICENSE.txt
├── README.rst
├── requirements.txt
├── setup.cfg
├── setup.py
├── src
│   └── assignment2pyscaffold
│       ├── __init__.py
│       └── skeleton.py
└── tests
    ├── conftest.py
    └── test_skeleton.py
```

- This putup automatically synced my details of name and author in the setup.cfg file, with space to enter a project description

```
# This file is used to configure your project.
# Read more about the various options under:
# http://setuptools.readthedocs.io/en/latest/setuptools.html#configuring-setup-
using-setup-cfg-files

[metadata]
name = assignment2pyscaffold
description = Add a short description here!
author = Daniel O Byrne
author-email = daniel.obyrne@ucdconnect.ie
license = mit
url = http://...
long-description = file: README.rst
# Change if running only on Windows, Mac or Linux (comma-separated)
platforms = any
# Add here all kinds of additional classifiers as defined under
# https://pypi.python.org/pypi?%3Aaction=list_classifiers
classifiers =
    Development Status :: 4 - Beta
    Programming Language :: Python
```

- Again I found setting up a project on pyscaffold very efasy and effective, coming across no issues.
- I would use this tool again for creating python projects in the future. The projects details are stored in the setup.cfg file, thus no tampering with the setup.py file is necessary. An efficient system

SampleMod

- I found the article by Kenneth Reitz very intersting and helpful in understanding the importance of file structure when creating python projects.

```
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670/assignment2SM$ tree
.
└── samplemod
    ├── docs
    │   ├── conf.py
    │   ├── index.rst
    │   ├── make.bat
    │   └── Makefile
    ├── LICENSE
    ├── Makefile
    ├── MANIFEST.in
    ├── README.rst
    ├── requirements.txt
    ├── sample
    │   ├── core.py
    │   ├── helpers.py
    │   └── __init__.py
    ├── setup.py
    └── tests
        ├── context.py
        ├── __init__.py
        ├── test_advanced.py
        └── test_basic.py

4 directories, 17 files
```

- I used git clone and install the project template on a local folder and the above shows the file structure. Porject and authr details are then entered in the setup.py file, a similar process to the PPT structure entering details in the metadata.py.

```
# -*- coding: utf-8 -*-

# Learn more: https://github.com/kennethreitz/setup.py

from setuptools import setup, find_packages


with open('README.rst') as f:
    readme = f.read()

with open('LICENSE') as f:
    license = f.read()

setup(
    name='Daniel O Byrne',
    version='0.1.0',
    description='Sample package for Python-Guide.org',
    long_description=readme,
    author='Daniel O Byrne',
    author_email='daniel.obyrne@ucdconnect.ie',
    url='https://github.com/kennethreitz/samplemod',
    license=license,
    packages=find_packages(exclude=('tests', 'docs'))
)
```

- It is a simple, clean structure appropriate for python projects. It is an effective structure as it does not contain numerous nested repositories, instead three folders (sample, docs and tests) , with relevant python files inside of them.
- I would use this tool again and found no issues when configuring.

- Overall however, I found the cookicutter tool to be most effective. It's command line interface asking for project details and general ease of use made it the clear choice for me

Q2 (a)

(Note: In order to get tflask_platform folder i created, the user must first clone the repository from GitHub; https://github.com/obyrned1/flask_platform)

- I used CookieCutter to set up the project named systeminfo. It created the resulting file structure:

```
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670/systeminfo$ tree
.
├── AUTHORS.rst
├── CONTRIBUTING.rst
├── docs
│   ├── authors.rst
│   ├── conf.py
│   ├── contributing.rst
│   ├── history.rst
│   ├── index.rst
│   ├── installation.rst
│   ├── make.bat
│   ├── Makefile
│   ├── readme.rst
│   └── usage.rst
├── HISTORY.rst
├── LICENSE
├── Makefile
├── MANIFEST.in
├── README.rst
├── requirements_dev.txt
├── setup.cfg
├── setup.py
├── systeminfo
│   ├── cli.py
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-36.pyc
│   │   └── systeminfo.cpython-36.pyc
│   └── systeminfo.py
├── tests
│   ├── __init__.py
│   ├── __pycache__
│   │   └── __init__.cpython-36.pyc
│   └── test_systeminfo.py
└── tox.ini
```

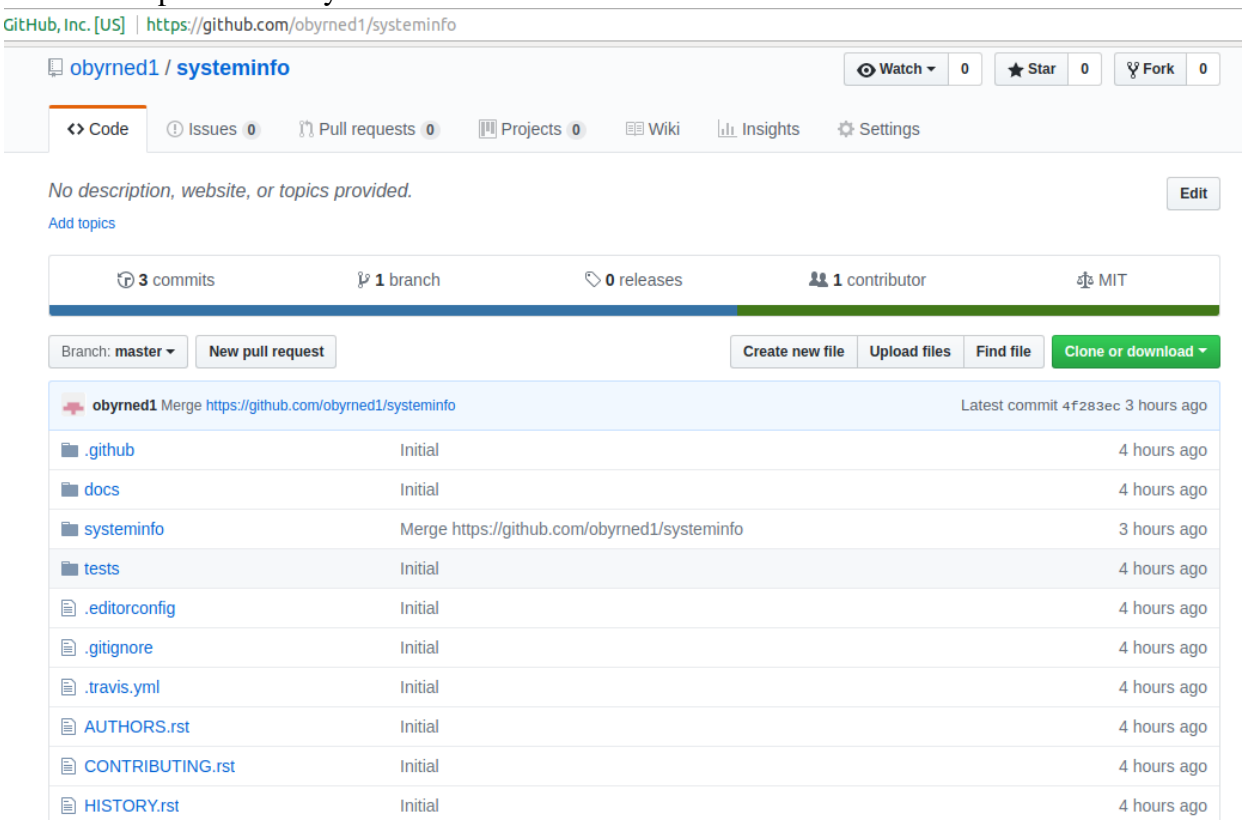
- In the system folder, within this system info project folder, a python file called systeminfo.py file is created

```
├── systeminfo
│   ├── cli.py
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-36.pyc
│   │   └── systeminfo.cpython-36.pyc
│   └── systeminfo.py
```

- I transferred my systeminfo script from previous practicals into this systeminfo.py file. I adjusted this slightly by just having a return statement, rather than a print

```
obyrned1@obyrned1-Aspire-V5-122P: ~/compsci/comp30670/systeminfo/systeminfo
  GNU nano 2.5.3                                    File: systeminfo.py

# -*- coding: utf-8 -*-

"""Main module."""

import platform
def main():
        return platform.platform()


if __name__ == '__main__':
    main()
```

- I then pushed this systeminfo folder to GitHub



- I installed this systeminfo python module from the GitHub repository it was located in:

```
pip install git+https://github.com/obyrned1/systeminfo.git
```

- To check this was properly installed as a module, I used pythont to get a list of the installed modules:

```
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670/systeminfo/systeminfo$ python
Python 3.6.4 | packaged by conda-forge | (default, Dec 23 2017, 16:31:06)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-15)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> help('modules')

Please wait a moment while I gather a list of all available modules...
```

- it's existence in this list confirmed proper installation:

```
sys
sysconfig
syslog
systeminfo
tabnanny
tarfile
telnetlib
tempfile
```

- Subsequently, I opened a python interpreter to test the module and as seen, it ran as required

```
>>> from systeminfo import systeminfo
>>> systeminfo.main()
Linux-4.13.0-32-generic-x86_64-with-debian-stretch-sid
>>>
```

Q2(b)

- As per slide 51 in Software Necessities, I created the following file structures:

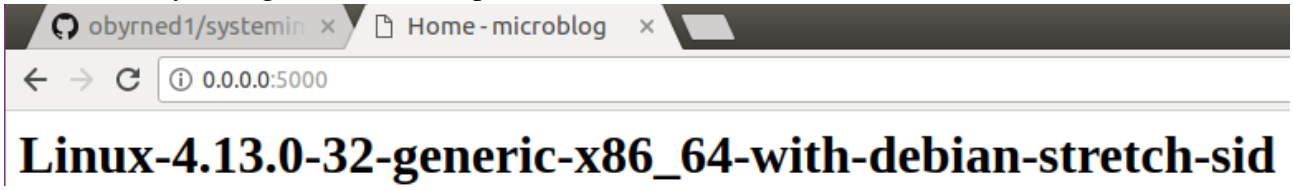
```
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670/flask_platform/flask_platform/src/dummyapp$ ls
app  __pycache__  run.py
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670/flask_platform/flask_platform/src/dummyapp$ cd app/
(comp30670) obyrned1@obyrned1-Aspire-V5-122P:~/compsci/comp30670/flask_platform/flask_platform/src/dummyapp/app$ ls
__init__.py  __pycache__  static  templates  views.py  views.save
```

- Instead of the views.py file printing out Hello as per slide 51, I imported my systeminfo module and instead printed the type of my machine to the local host. I made the following adjustments to views.py and index.html:

```
obyrned1@obyrned1-Aspire-V5-122P: ~/compsci/comp30670/flask_platform/flask_platform/src/dummyapp/app
  GNU nano 2.5.3                                    File: views.py

# dummyapp/app/views.py
from flask import render_template
from app import app

from systeminfo import systeminfo


@app.route('/')
def index():
        returnDict = {}
        returnDict['sysinfo'] = systeminfo.main()
        returnDict['title'] = 'Home'
        return render_template("index.html", **returnDict)
```

```
obyrned1@obyrned1-Aspire-V5-122P: ~/compsci/comp30670/flask_platform/flask_platform/src/dummyapp/app/templates
  GNU nano 2.5.3                                    File: index.html

<!-- dummyapp/app/templates/index.html -->
<html>
<head>
<title>{{ title }} - microblog</title>
</head>
<body>
<h1>{{ sysinfo }}</h1>
</body>
</html>
```

- Thus yielding the desired output:



(c) Bonus part

- The README files in both the flask_platform and systeminfo folders give an explanation of how to run the system. In order to get the flask_platform folder, the user must first clone the repository from GitHub. I will show how to do this below as part of the advanced section:

```
ubuntu@ip-172-31-45-34:~$ git clone https://github.com/obyrned1/flask_platform
Cloning into 'flask_platform'...
remote: Counting objects: 51, done.
remote: Compressing objects: 100% (40/40), done.
remote: Total 51 (delta 3), reused 51 (delta 3), pack-reused 0
Unpacking objects: 100% (51/51), done.
Checking connectivity... done.
ubuntu@ip-172-31-45-34:~$ ls
flask_platform
```

- I installed systeminfo as a module, the exact same way as before:

```
ubuntu@ip-172-31-45-34:~$ pip install git+https://github.com/obyrned1/systeminfo
Collecting git+https://github.com/obyrned1/systeminfo
  Cloning https://github.com/obyrned1/systeminfo to /tmp/pip-7Y_o4u-build
Collecting Click>=6.0 (from systeminfo===0.1.0)
  Downloading click-6.7-py2.py3-none-any.whl (71kB)
    100% |                                        | 71kB 2.1MB/s
Installing collected packages: Click, systeminfo
  Running setup.py install for systeminfo ... done
Successfully installed Click systeminfo
```

- I had to change some security group settings on the aws website, then running run.py as before, my machine's model/platform was returned