Assignment 1: Salutations

Create Python program called `saluations.py` that will print a friendly greeting.

Be sure you are in your virtual environment

Open the be434-spring-2023 directory in VS Code (in your GitHub repository) and then activate the virtual machine:

For a Mac, you will need to enter the following commands in the terminal:

```
source .venv/bin/activate
```

For a PC, you will need to enter the following commands:

```
.venv\scripts\activate
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process
```

Installing new.py Into Your PATH

I hate writing code from scratch! This week you learned about using a program called `new.py` that will create a program for you to start from. Now, we need to add this program to our PATH, so we can just use it without having to figure out where it is!

In the *bin* directory of your repo, you should find a program called `new.py` that will help you make a new Python program. From this directory, you can provide the full path using `..` to indicate the parent directory:

```
$ cd ./assignments/01_salutations
$ ../../bin/new.py -h
usage: new.py [-h] [-n NAME] [-e EMAIL] [-p PURPOSE] [-f] program

Create Python argparse program

positional arguments:
  program               Program name

optional arguments:
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Name for docstring (default: Ken Youens-Clark)
  -e EMAIL, --email EMAIL
                        Email for docstring (default: kyclark@gmail.com)
  -p PURPOSE, --purpose PURPOSE
                        Purpose for docstring (default: Rock the Casbah)
  -f, --force           Overwrite existing (default: False)
```

It will be unpleasant to always indicate the full path to `new.py` as you will use it often. I suggest you create a directory in your `$HOME` (which is often written using the tilde `~` AKA "twiddle") to put useful programs you'll write. It's common to create a *~/local* or *~/.local* (so it's hidden) to install software, and inside of that a *bin* directory:

```
mkdir ~/.local
mkdir ~/.local/bin
```

You will need to ensure that this directory is included in your `$PATH`. First check what Unix shell you are using:

```
echo $SHELL
```

If are using the `bash` shell, you can edit *~/.bashrc*. If are using the `zsh` shell, you can edit *~/.zshrc*. For instance, you can use `nano`:

```
nano ~/.bashrc
```

Add this line to the end:

```
export PATH=~/.local/bin:$PATH
```

Then use the `source` command to read this file and alter your `$PATH`:

```
source ~/.bashrc
```

You can view your `$PATH` to ensure this directory is included:

```
echo $PATH
```

Then you can copy the `new.py` program to that location:

```
cp ../../bin/new.py ~/.local/bin
```

Verify that the program can be found using `which`:

```
$ which new.py
```

My new.py file is located here: /Users/bhurwitz/.local/bin/new.py

Getting Started with new.py

Here is how you can create the `salutations.py` using `new.py`:

```
$ new.py -p 'Print greeting' salutations.py
Done, see new script "salutations.py."
```

Open the new `salutations.py` program and modify it to accept three optional arguments:

- `-g|--greeting`: A greeting, defaults to "Howdy"
- `-n|--name`: A name to greeting, defaults to "Stranger"
- `-e|--excited`: A flag to terminate the greeting with an exclamation point

The program should respond to `-h|--help` to print the following usage:

```
$ ./salutations.py -h
usage: salutations.py [-h] [-g str] [-n str] [-e]

Greetings and salutations

optional arguments:
  -h, --help            show this help message and exit
  -g str, --greeting str
                        The greeting (default: Howdy)
  -n str, --name str    Whom to greet (default: Stranger)
  -e, --excited         Include an exclamation point (default: False)
```

When run with no arguments, it should use the default values to print the following:

```
$ ./salutations.py
Howdy, Stranger.
```

The `-g|--greeting` option should cause it to use the provided greeting:

```
$ ./salutations.py -g Sup
Sup, Stranger.
```

The `-n|--name` option should cause it to use the provided name:

```
$ ./salutations.py -n Amanda
Howdy, Amanda.
```

The `-e|--excited` flag should cause the greeting to end with a bang:

```
$ ./salutations.py -e
Howdy, Stranger!
```

The program should accept any combination of the short or long names of the arguments:

```
$ ./salutations.py --greeting Sup --name Dude --excited
Sup, Dude!
```

Testing

The test suite will require the modules pytest, flake8, and pylint which you can install with the following command (if you have not already installed these using the requirements.txt during setup):

```
$ python3 -m pip install pytest flake8 pylint
```

You can run the test suite with the following command:

```
$ pytest -xv test.py salutations.py
$ flake8 salutations.py
$ pylint salutations.py
```

You can also use the Makefile shortcut:

```
$ make test
```

The tests include linting with `pylint` and `flake8`, so be sure that you format your code with something like `yapf` or `black` (which may need to be installed using the `pip` module above.

A passing test suite looks like this:

```
============================= test session starts
==============================
...
---------------------------------------------------------------------------
```

```
------
Linting files
.
--------------------------------------------------------------------------
------

test.py::PYLINT PASSED                                                   [
11%]
test.py::FLAKE8 PASSED                                                   [
22%]
test.py::test_exists PASSED                                              [
33%]
test.py::test_usage PASSED                                               [
44%]
test.py::test_defaults PASSED                                            [
55%]
test.py::test_greeting PASSED                                            [
66%]
test.py::test_name PASSED                                                [
77%]
test.py::test_excited PASSED                                             [
88%]
test.py::test_all_options PASSED
[100%]

============================== 9 passed in 0.51s
===============================
```

Your grade is whatever percentage of tests your code passes.

Authors

Bonnie Hurwitz bhurwitz@arizona.edu and Ken Youens-Clark kyclark@gmail.com