3 — PHY 494: Homework assignment (50 points total)

Due Friday, Feb 9, 2018, 5pm.

Submission is now to your **private GitHub repository**. Follow the link provided to you by the instructor in order for the repository to be set up: It will have the name *ASU-CompMethodsPhysics-PHY494/assignments-2018-*YourGitHubUsername and will only be visible to you and the instructor/TA. Follow the instructions below to submit this (and all future) homework.

Read the following instructions carefully. Ask if anything is unclear.

1. `git clone` your assignment repository (change *YourGitHubUsername* to your GitHub username)

   ```
   repo="assignments-2018-YourGitHubUsername.git"
   git clone https://github.com/ASU-CompMethodsPhysics-PHY494/${repo}
   ```

2. run the script `./scripts/update.sh` (replace *YourGitHubUsername* with your GitHub username):

   ```
   cd ${repo}
   bash ./scripts/update.sh
   ```

   It should create three subdirectories[1] `assignment_03/Submission`, `assignment_03/Grade`, and `assignment_03/Work`.

3. You can try out code in the `assignment_03/Work` directory but you don't have to use it if you don't want to. Your grade with comments will appear in `assignment_03/Grade`.

4. Create your solution in `assignment_03/Submission`. Use Git to `git add` files and `git commit` changes.

   You can create a PDF, a text file or Jupyter notebook inside the `assignment_03/Submission` directory as well as Python code (if required). **Name your files `hw03.pdf` or `hw03.txt` or `hw03.ipynb`**, depending on how you format your work. Files with code (if requested) should be named exactly as required in the assignment.

5. When you are ready to submit your solution, do a final `git status` to check that you haven't forgotten anything, commit any uncommited changes, and `git push` to your GitHub repository. Check on *your* GitHub repository web page[2] that your files were properly submitted.

   You can push more updates up until the deadline. Changes after the deadline will not be taken into account for grading.

Homeworks must be legible and intelligible and on-time or may be returned ungraded with 0 points.

[1] If the script fails, file an issue in the Issue Tracker for PHY494-assignments-skeleton and just create the directories manually.

[2] `https://github.com/ASU-CompMethodsPhysics-PHY494/assignments-2018-`*`YourGitHubUsername`*

3.1 Version control with Git (9 points)

Keep your answers short, one or two sentences should be sufficient for questions (b)–(d).

(a) *Briefly* explain what a version control system such as Git does and how it can help you. (For your answer, it is sufficient to focus on three different aspects out of many — choose the ones that *you* find most important.) [**3 points**]

(b) Explain the difference between `git init` and `git clone`. [**2 points**]

(c) Explain the difference between `git add` and `git commit`. [**2 points**]

(d) Explain the difference between `git commit` and `git push` [**2 points**]

3.2 Be the Git (10 points)

In this problem you should state various outcomes when a number of git commands are run on the directory structure in Figure 1 (*Documents* is the top-level directory, sub-directories are indicated by blue folder icons, files are text):[3]

User dvader decides (quite sensibly) to use version control, namely **git**, for his Documents. The *Documents* directory was not under version control before. The first set of commands that he performs is:

```
cd Documents
git init
git add shopping.txt
git add planets/alderaan.jpg
```

To answer the following question, create a table where you list the command that is being executed, the *files in the git staging area* and the *files in the git repository.* After the initial commands, your table should look like this:

command	staging area	repository
git init		
git add shopping.txt	shopping.txt	
git add planets/alderaan.jpg	shopping.txt, planets/alderaan.jpg	

List *all* files that are in either staging area or repository, not just new ones. If there are no files (as in the repository during the first three `git` commands), leave the space empty.

Add the table above to your submitted solution and continue it by listing all the files in the staging area and in the repository, after each of the following commands have been carried out:

[3] You can pretend to be git or you can also create the directory structure yourself and run the commands.

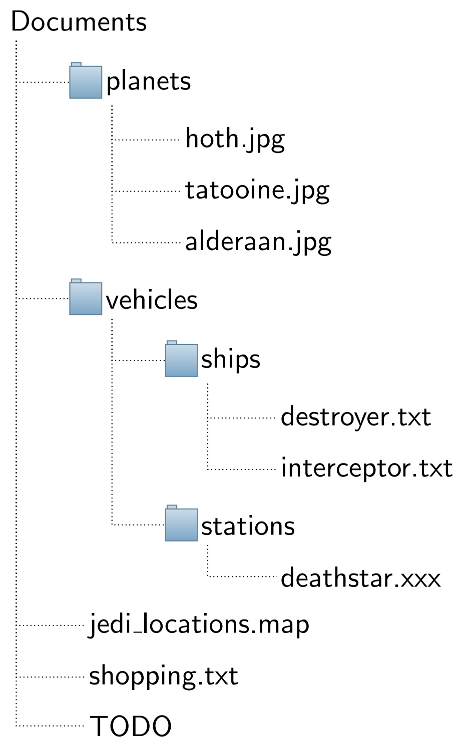

Figure 1: Part of the directory tree of user dvader. *Documents* is the top-level directory, sub-directories are indicated by blue folder icons, files are just shown as text with their file name, e.g., *TODO* or *hoth.jpg*.

```
git commit
git add vehicles/stations
git commit
git add vehicles TODO
git commit
git rm planets/alderaan.jpg
git add planets
git commit
```

3.3 Your GitHub account (10 points)

As part of the last lesson you should have set up your own GitHub account on https://github.com (if you have not done it yet, do it now!). What is your **GitHub username**?

- Write down your GitHub username. [**10 points**]

- Take the survey PHY 494: Your GitHub account[4] if you have not done so already.

3.4 Simple coordinate manipulation in Python (11 points)

We can represent the cartesian coordinates $\mathbf{r}_i = (x_i, y_i, z_i)$ for four particles as a list of lists `positions`:

```
positions = \
    [[0.0, 0.0, 0.0], [1.34234, 1.34234, 0.0], \
     [1.34234, 0.0,  1.34234], [0.0, 1.34234, 1.34234]]
```

For the following, do not import any additional modules: *only use pure Python.* The repository contains skeleton code `coordinates_a.py`, `coordinates_b.py`, `coordinates_c.py`, `coordinates_d.py` for the sub-problems. Add your code to these files and *submit them as part of your solution*—code must be included to get full points.[5]

(a) Access the coordinates of the second particle, assign it to a variable `particle2`, and print the coordinates. [**1 points**]

(b) Access the y-coordinate of the second particle, assign it to the variable `y2`, and print the value. [**1 points**]

(c) Write Python code to translate all particles by a vector $\mathbf{t} = (1.34234, -1.34234, -1.34234)$,

```
t = [1.34234, -1.34234, -1.34234]
```

Assign the translated coordinates to the variable `new_positions` and print them. [**3 points**]

(d) Make your solution of (c) a function `translate(coordinates, t)`, which translates all coordinates in the argument `coordinates` (a list of N lists of length 3) by the translation vector in `t`. The function should return the translated coordinates.

Apply your function to (1) the input `positions` and `t` from above and (2) for `positions2 = [[1.5, -1.5, 3], [-1.5, -1.5, -3]]` and `t = [-1.5, 1.5, 3]` and show the output. [**6 points**]

[4] In case the link to the survey is not clickable: got to https://goo.gl/forms/AZMtF6c60FBdCCkt1. You must be logged in with your ASU account. Log in to https://my.asu.edu first and then go to the survey.

[5] You will also see a file `test_coordinates.py`. It contains *tests* that check your code. Your instructors will *run these tests on your code.* You can run them yourself with the `pytest` command,

```
pytest -v test_coordinates.py
```

If everything is correct, you should see something like `===== 9 passed in 0.10 seconds =====`. If tests fail then you can correct your code until you get the tests to pass.

3.5 Squash the bug (10 points)

Three files `bug_a.py`, `bug_b.py`, and `bug_c.py` are include. Each contains at least one Python bug. Fix them and commit your fixed files.[6]

(a) Fix `bug_a.py` and commit the fixed file. The code should run, assign the correct value to the variable `value`, and print the correct value. [**3 points**]

(b) Fix `bug_b.py` and commit the fixed file. The code should add two vectors $\mathbf{a} = (12.3, 3.90, 4.5)$ and $\mathbf{b} = (1.3, 0.91, -3.3)$ and print the value of the new vector $\mathbf{c} = 5\mathbf{a} - 3\mathbf{b}$ and assign the value to the variable `c`. [**3 points**]

(c) Fix `bug_c.py` and commit the fixed file. You should correctly implement the 2D sinc function

$$\operatorname{sinc}(x, y) := \frac{\sin x}{x}\frac{\sin y}{y}$$

(where x and y are given in radians). The function should be correct for arbitrary input.[7] [**4 points**]

[6]You will also see a file `test_bugs.py`. It contains *tests* that check your code. Your instructors will *run these tests on your code*. You can run them yourself with the `pytest` command,

```
pytest -v test_bugs.py
```

If everything is correct, you should see something like `===== 36 passed in 0.36 seconds =====`. If tests fail then you can correct your code until you get the tests to pass.

[7]Hint: Especially consider the edge and corner cases.