3 — PHY 494: Homework assignment (52 points total)

Due Thursday, Jan 31, 2019, 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-2019*-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-2019-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.

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

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.

This assignment contains **bonus problems**. A bonus problem is optional. If you do it you get additional points that count towards this homework's total, although you can't get more than the maximum number of points. If you don't do it you can still get full points. Bonus problems and bonus points are indicated with an asterisk "*".

3.1 Python data types (7 points)

What is the Python data type of each of the following values?

(a) `3.14515` **[1 points]**

(b) `0` **[1 points]**

(c) `False` **[1 points]**

(d) `'To be or not to be'` **[1 points]**

(e) `[3, 2, 1, "lift off!"]` **[1 points]**

[2] `https://github.com/ASU-CompMethodsPhysics-PHY494/assignments-2019-`*`YourGitHubUsername`*

(f) `3, 2, 1, "lift off!"` **[1 points]**

(g) `None` **[1 points]**

(h) BONUS: `{'name': 'Hamlet', 'occupation': 'Prince'}` **[bonus +1*]**

3.2 Python Lists and Strings (20 points)

Lists and strings share some similarities but also have important differences. Let's look at them. (Type code in the Python interpreter e.g., `ipython`).

```
bag = ["guide", "towel", "tea", 42]
ga = "Four score and seven years ago"
```

(a) How do you have to slice `bag` in order to get `['towel', 'tea']`? **[1 points]**

(b) What does `bag[::-1]` do?

How do you slice bag in order to get `['tea', 'towel']`? **[2 points]**

(c) Strings can also be sliced. How do you have to slice `ga` to get

- "Four"
- "seven"

[2 points]

(d) You can access elements of a list in a variety of ways:

(i) Explain what

```
bag[0] = 'book'
```

does? (Hint: print `bag`!) **[1 points]**

(ii) Create two new variables:

```
mybag = bag
yourbag = bag[:]
```

and use them:

```
mybag[3] = "mice"
yourbag.append("money")
```

What is the content of `bag`, `mybag`, `yourbag`? **[2 points]**

(iii) From your observations, explain how the assignment `x = a` differs from `y = a[:]`? **[3 points]**

(e) Try

```
ga[:4] = "Three"
```

(i) Describe what happens?[3] **[1 points]**

(ii) How would you construct the string `"Three score and seven years ago"` from `ga` and the string `"Three"`? **[1 points]**

(f) What do the commands

```
ga.split()
a, b, c = ga.split()[:3]
list([1,2,3])
list(ga)
```

do? You can show the output but you need to explain in your own words what is happening. **[4 points]**

(g) Nested lists: Given the list

```
bags = [['salt', 'pepper'], ['pen', 'eraser', 'ruler']]
```

how do you have to index `bags` to get

(i) `['salt', 'pepper']` **[1 points]**

(ii) `'pepper'` **[1 points]**

(iii) `'ruler'` **[1 points]**

[3] Note that strings are "immutable" objects in Python whereas lists are "mutable".

3.3 Loops (14 points)

(a) Use a `for` loop to print each element of the list

```
sentence = ["We", "must", "walk", "before", "we", "can", "run"]
```

Show your code and your output. **[4 points]**

(b) BONUS: Find a way to only print every other line from `sentence`, i.e., the output should read "We walk we run". Show your code and your output. **[bonus +2*]**

(c) Use a loop to sum the integers from 1 to 1000 and print the final result (500500). **[4 points]**

Note: You can increment a variable `total` with a value `x` with the assignment

```
total = total + x
```

or equivalently but more compactly written

```
total += x
```

Both expressions do the same thing: add two values together and then assign the sum to the variable `total`, overwriting the old value of `total` in the process.

(d) Write a program that counts down from 10 to 0 and prints each number ("10 9 8 ... 3 2 1").

(i) Use a `while` loop. **[3 points]**

(ii) Use a `for` loop. **[3 points]**

(You can use any other Python functions that you might need, such as `range`.)

Show your code and output.

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.[4]

(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] 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.