This activity aims to give you a (*very*) brief introduction to Python so that you will be able to focus on the physics while you are working on the solar system simulation. If you are familiar with programming, this will likely be something you can move through quickly; when you finish it, work on the challenge problem at the bottom. If you have not spent much time programming, take this opportunity to ask lots of questions of Dr. Slinker and your peers.

As you work through this lab, make sure you are answering the questions throughout the document. They are also included at the end to give you a concrete list of deliverables; but make sure you look at the question in the full instructions to get context.

Getting Python

If you have not used Python before (and I'm not expecting that you have) you can run Python on Google Collabs.

If you believe that you'll be using Python a lot (which is pretty likely if you're going to do science), I recommend that you obtain Python on your own computer by downloading Anaconda `https://www.anaconda.com/products/individual`. Follow the instructions on the website to download the most recent version of Python. This should work on any operating system. You can edit your code in a browser-based front end through Jupyter.

If you already have experience with Python, feel free to use whatever incarnation of it you like best.

Getting started with Python

Before you can jump straight into simulating the solar system, open your own notebook and experiment around with Python a bit. For now, you should just get in the habit of starting all of your Python code with the following two lines:

```
import numpy as np
import matplotlib.pyplot as plt
```

The first line allows you to use a range of mathematical tools. The second allows you to produce plots. The first line especially is very standard practice and something you will find in pretty much any Python code used for scientific computing.

Try running a few simple things. Can you get Python to tell you what $2 + 3$ is? How about 4×9? Or $\sqrt{4}$ (hint: try `np.sqrt`)? If you hit `Enter`, you will insert a line in your code. To run a chunk of code, hold `Shift` and press `Enter` (in a Jupyter notebook).

Vectors in Python

In physics computing you will need to use a lot of vectors. A good way to do that in Python is using numpy arrays. In the simulation code I will give you, I would implement the vector $\vec{v} = (1, 2, 3)$ using `v=np.array([1,2,3])`. Define this vector in your code. Also define the vector $\vec{u} = (4, 5, 6)$.

Question 1: Try the following operations on $\vec{v}$ and $\vec{u}$. For each, describe what the function is doing (in words, not just the output).

```
u+1
2*v
v+u
v-u
v**2
u**0.5
np.sum(u)
#comment
v[1]
v[-1]
v[2]
v[-2]
```

Try combining all of the ideas from above to find $|\vec{v}-\vec{u}|$ (i.e., the length of the vector $\vec{v}-\vec{u}$). As a reminder, $|\vec{w}| = \sqrt{w_x^2 + w_y^2 + w_z^2}$.

Functions in Python

You are probably familiar with mathematical functions, even if you are not familiar with functions in programming. **The following example can be helpful to see the outline of how a function works, though it is not something you will use directly in your own code.** We can think of addition as a function which takes two inputs (the numbers you're adding) and returns a single output (the numbers' sum). A short piece of Python code to add numbers could look like the following:

```
#bring in numerical functions and plotting utilities
import numpy as np
import matplotlib.pyplot as plt

#define a function to add three numbers
def add_three_numbers(n1,n2,n3):
    sum = n1 + n2; #find the sum of the first two
    sum = sum + n3;
    return sum #output the sum

add_three_numbers(5,6,7) #add 5, 6, and 7
```

Make sure you understand what each of the three chunks of the code is doing:

- the first three lines have to do with importing mathematical functions and plotting utilities (see above),
- the last line adds 5, 6, and 7 (it should return 18) by calling the function `add_three_numbers`, and
- the rest of the code gives a definition for the function `add_three_numbers`, which accepts three arguments called `n1`, `n2`, and `n3` and returns the `sum` of the three arguments.

Now take the code you wrote to compute $|\vec{v}-\vec{u}|$ and make a Python function called `magnitude_of_difference` which takes two arbitrary vectors and computes the magnitude of their difference. Once you write this function, you should be able to type in the line

```
magnitude_of_difference(np.array([1,0,0]),np.array([0,1,2]))
```

and get `2.449489742783178`. Test out various different vectors to make sure the function works as expected.

Question 2: Submit the code you wrote for `magnitude_of_difference`.

Keep this function; you will need to use it for the solar system simulation lab!

Logic in Python

In the solar system simulation, the code will need to divide by $|\vec{v} - \vec{u}|$. But if $\vec{v}$ and $\vec{u}$ are the same vector you will end up dividing by zero! You will need to use `if` statements to avoid doing this.[1]Note: `==` is used to test whether two things are equal,[2] `&` is the logical AND function, and `|` is the logical OR function.

Use this to write a function called `reciprocal_of_magnitude_of_difference` which computes $1/|\vec{v} - \vec{u}|$ when $\vec{u} \neq \vec{v}$ and gives 0 when $\vec{u} = \vec{v}$. Two hints: $|\vec{v} - \vec{u}| = 0$ if and only if $\vec{u} = \vec{v}$ and `reciprocal_of_magnitude_of_difference` can call `magnitude_of_difference`. Once again, test out various different vectors to make sure the function works as expected.

Question 3: Submit the code you wrote for `reciprocal_of_magnitude_of_difference`.

[1]The solar system simulation code just returns zero instead of trying to compute $\frac{1}{|\vec{v} - \vec{u}|}$ if the vectors are the same. This strategy works in this case, but it is arguably bad coding practice (though why this is so is not necessarily important to understand this class).

[2]Whereas a single equals sign `=` sets the value of the variable on the left hand side to be that of the right hand side.

Questions

Question 1: Try the following operations on $\vec{v}$ and $\vec{u}$. For each, describe what the function is doing (in words, not just the output).

```
u+1
2*v
v+u
v-u
v**2
u**0.5
np.sum(u)
#comment
v[1]
v[-1]
v[2]
v[-2]
```

Question 2: Submit the code you wrote for `magnitude_of_difference`.

Question 3: Submit the code you wrote for `reciprocal_of_magnitude_of_difference`.