Mod 1 Lab - Basic Python

This lab will walk you through the process of settig up your CSE 2050 VM and introduce the typical lab workflow:

1) Download lab instructions from HuskyCT
2) Write code
3) Submit code to Gradescope

This lab is cooperative - talk with your partner as you go through, and make sure you are progressing together.

Part 1 - VM Setup

Follow the instructions in `01_vm_intro.pdf` to initialize your VM.

STOP - make sure your partner is done here before continuing.

Part 2 - Code Server initialization

Follow the instructions in `02_codeserver_intro.pdf` to get used to working in code server.

Next, click on the *Extensions* icon on the left-hand sidebar in code-server and install:

- `python` by `ms-python`
- `vcode-pdf` by `tomoki1207`

STOP - make sure your partner is done here before continuing.

Part 3 – hello.py

Add a function `say_hi()` to the file `hello.py` you created in Part 2. This function should just return the string `Hello, world` for now.

```python
def say_hi():
   return "Hello, world"
```

Submit code to Gradescope

Next, we'll submit our code to gradescope.

If you are working in the VM, you can right-click on a file in the `EXPLORER` tab and download it. Regardless, you should keep local copies of all files used for this class in a common directory with a name like `cse2050`. Create new directories for each assignment, so you get a file structure like the following by the third week of class:

```
|cse2050
|  |lab1
|  |  |hello.py
|  |  |lab1.py
|  |
|  |lab2
|  |  |lab2.py
|  |
|   |lab3
|   |   |lab3.py
|   |
|   |hw1
|   |   |hw1.py
|   |
|   |hw2
|   |   |hw2.py
|   |
|   |hw3
|   |   |hw3.py
```

To submit to Gradescope, click the `Gradescope` link in HuskyCT (available on the left-hand sidebar for this course), then select the appropriate assignment. For now, just submit `hello.py`.

The autograder takes a minute or two to run. Once it completes, you should see 10/100 points for this assignment if `hello.py` is correct. Read over the error messages for the test cases you failed to get an idea of why they failed (largely, they depend on the file `lab1.py`, which you have not yet created and did not submit.)

STOP - make sure your partner is done here before continuing.

Rinse and repeat

Now, we'll continue to work on our code and re-submit until we pass all test cases. To complete this assignment, create a new file called `lab1.py`. This file should have a single function named `generic_hi()` which takes one argument: a name to add to the return string:

```
>>> generic_hi('jake')
'Hello, jake!'
>>> generic_hi('greninja')
'Hello, greninja!'
>>> generic_hi()
'Hello, world!'
```

Note that your function needs a default value (`'world'` in this case) to plug in to the return string if the user does not specify one.

External Modules

Do not use any imported modules (`math`, `collections`, `...`) when implementing functionality. It is okay to use imported modules for testing.

It is okay to import modules you write yourself; e.g. any data structures you write yourself.

Submitting

STOP!. Before you go, make sure to backup your files to local storage or (ideally) a cloud service like Onedrive.

At a minimum, submit the following files:

- `hello.py`

- `lab1.py`

Students must submit **individually** by the due date (typically, Sunday at 11:59 pm EST) to receive credit.