Programming on Purpose (CSC 120)
Winter 2024

Lab 1 – Re-presenting Python

1/4/2024

Objectives

- To re-acquaint yourself with programming in Python
- To get acquainted with the Visual Studio Code development environment
- To practice modularity using functions and modules

1 Preliminary setup

1. If you haven't already logged into a lab machine, do so now. If you have never changed your CS password before, it will be the word *union* followed by your ID number, e.g. *union12345*. In that case, you should now change it:
   (a) Open a terminal.
   (b) Type `yppasswd` and follow the instructions. (Note that you will see nothing as you are typing in your old and new passwords.)
2. Make a "csc120" folder. And in that folder, create a folder for this lab.

2 Integrated Development Environments

In previous classes where you've used Python, you used a program called IDLE to edit and run your programs. However, Python does not equal IDLE. Python is a *language* in which you can write programs, and IDLE the *tool* you use to edit those programs. IDLE is simply a text editor with some special Python programming support. Succinctly, Python is not IDLE in the same way that English is not Microsoft Word.

Once your programs are in the right language (in this case Python), they can be *compiled* or *interpreted* by appropriate tools (in this case, the Python interpreter). When you used IDLE before, it provided a window for text file editing (where you could type your programs) and a way to run those programs. To run the programs, IDLE would call the Python interpreter to do the job. Because editing and running are integrated together, we call IDLE an Integrated Development Environment (IDE).

Another IDE, one used by professional developers, is called Visual Studio Code. Visual Studio Code was developed by Microsoft. (Confusingly, Microsoft also has a line of IDEs called Visual Studio. We are using Visual Studio *Code*, aka VS Code, which is different.) In this course, you will use Visual Studio Code instead of IDLE. But remember, the language you are using is still the same old Python you are familiar with.

3 Getting started with Visual Studio Code

1. Start VS Code.
2. Check that you have the Python extension installed. Here is how to do that:
   (a) To install the VS Code extension, open VS Code and choose `File > Preferences > Extensions`. That gives you a vertical bar on the left, which lists all extensions that are already installed. If you haven't installed any, yet, that list will be empty.
   (b) Check that the Python extension shows up in your list of installed extensions.
   (c) If necessary, install the Python extension.
   There is also a search box at the top of that vertical bar. When you type "Python" into it, you get a long list of extensions that have to do with Python. The one that you want is most likely the first one in the list. The name is just "Python" and the description should also tell you that it was developed by Microsoft. Click on the little green "Install" button.
   On the lab machines, this takes a moment, but you can start to look at the code while you are waiting for the extension to be installed. You just can't run it, yet.

4 Add some starter code

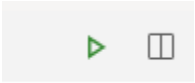
In many labs, including this one, I'll be providing you with starter code that you need to augment with your own code. Download `main.py` and `boneyard.py` from Nexus and add it to the folder you created earlier for this lab.

Then open that folder in VS Code: Choose `File > Open Folder` and select the lab 1 folder you created earlier. This should open up a new column on the left of your VS Code window which shows the contents of that folder.

If you ever loose that column, you can click on the Explorer icon in the top left to get it back.

5 Run the program

Open `main.py` (by clicking on the file name in the Explorer column on the left). To run your program, click the run button in the top right.



The output will appear in the terminal window at the bottom of your VS Code window. You should get an error complaining that the module named 'domino' is missing.

6 Add the domino module

Have a look at `boneyard.py`. Notice that it provides the functions used in the main module. Also notice that the module represents the boneyard as a list of domino representations, which are created using the `create` function in the `domino` module.

1. Add a new file `domino.py` to your lab folder. To do that, by clicking on the "add file" button in the Explorer column.

2. Add all of the functions that the `main` and `boneyard` modules expect the `domino` module to have (i.e. `create(left, right)`, `get_left(domino)`, `get_right(domino)`, and `as_str(domino)`). Start by creating preliminary dummy versions of those functions that just return 0 or `pass`.
3. Now, try running `main.py` again. Note: the program won't work correctly, yet, but it should run.

7 Finish the domino module

1. I'll help you with `as_str`:

```
def as_str(domino):
    return f"[{get_left(domino)} | {get_right(domino)}]"
```

2. Re-run the program and see what's changed.
3. Write the create function, which needs to create a representation for a domino with the given left and right values. For this lab, you should use a tuple to represent a domino (where the left value is the first item in the tuple and the right value is the second value). Recall that `(x, y)` creates a tuple with values x and y. (A tuple is a Python data structure that is like a list, except that it is immutable, meaning that it's elements cannot be modified.)
4. Re-run the program and see what's changed.
5. Now, write `get_left` and `get_right`. These functions get the left value and the right value of a given domino. The representation created by the create function determines how these two functions must do their job.
6. Re-run the program and see what's changed. Does the program do what we expect yet?

8 Revise your code

Before turning in any program in this class, remember this mantra:

> Just because it works doesn't mean it's good.

Part of your grade will also come from things like how understandable and readable your code is. You will also be graded on the neatness, presentation, and style of your program code.

Make sure that all modules (i.e. Python files) and functions are documented (even those that you didn't write).

Don't forget to cite who you received help from and include the honor code affirmation in your comments:

```
I affirm that I have carried out my academic endeavors
with full academic honesty. [Your Name]
```

9 How to submit

Submit the project by uploading all three Python files (`main.py`, `boneyard.py`, and `domino.py`) to Gradescope.

Make sure to add your partner to the submission so that you both have access to it through Gradescope.