Software Lab 1 *6.01 – Spring 2011*

Object-Oriented Programming

If you have already worked through the Python programming tutor and/or have had other Python experience, then go ahead and do the problems below.

If not, then please work through the Python tutor. If you need extra help in Python, come to our help session on Sunday. At that session, you can sign up for a free 'new programmer' extension on the work of this week.

1 Setup

Please use a lab laptop unless you have already installed and tested the 6.01 software modules on yours.

Using a lab laptop

- Get a lab laptop, charger, and mouse.
- Connect the laptop to the wired ethernet network (cables from center of tables).
- Log in using your Athena user name and password.
- Open a terminal window (Applications menu -> Accessories -> Terminal)
- In the terminal window, type
  `> athrun 6.01 getFiles`
  to set up files for today's exercises in your Athena directory (under `~/Desktop/6.01/swLab01`).

Using your own laptop

- Be sure you have the 6.01 software libraries installed: Software tab on `http://mit.edu/6.01`
- Go to the calendar tab of course web page: `http://mit.edu/6.01`
- Download and unzip `swLab01.zip` into a convenient folder (e.g., `~/Desktop/6.01/swLab01`).

2 Using course notes in lab

- Click once on the Firefox icon at the top left of the screen.
- Go to `http://mit.edu/6.01`.
- Click on the `Reference Material` tab.
- Click on `Course Notes`.
- In the pop-up window, click on `Open with`, choose `Document Viewer` from the pull-down list, and click `Ok`.

3 Using the online Tutor

If you have not already registered for the 6.01 tutor, do so now. (Note that the online homework tutor is different from the online Python tutor, which you may have been using to prepare for 6.01. You need to register separately for the homework tutor.)

- Click once on the Firefox icon at the top left of the screen.
- Go to `http://mit.edu/6.01`.
- Click on the `Online Tutor` tab.
- Under the Homework Tutor section, click on the `register here` link and follow instructions

4 Exercises

4.1 Fibonacci

Now, you will write the definition of the `fib` procedure, so that `fib(n)` returns the n^{th} Fibonnaci number. Recall the definition of `fib`:

$$\text{fib}(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ \text{fib}(n-1) + \text{fib}(n-2) & \text{if } n > 1 \end{cases}$$

In the Terminal window, type
`> idle &`
You can type Python expressions in `idle`'s Python Shell window.
You can write your programs in a file and test them using `Run Module`.
For example:

- Click `idle`'s File menu, select New Window, and write the following line in the window: `print 'Hello World'`
- Click `idle`'s File menu, select Save as, and type `~/Desktop/6.01/swLab01/test.py` in the filename box.
- Click `idle`'s Run menu, then select Run Module.
- Look at the Python Shell window: you should see `Hello World`.

Now look at problem Wk.1.3.1 on the Tutor.

- Click Idle's `File` menu, select `Open`, navigate to `~/Desktop/6.01/swLab01/` and choose the file name `sl01Work.py`.
- Complete the definition in this window. After the definition include some test cases (remember to print the result). So, your file should look like:

```
def fib (n):
  ...
print fib(1)
print fib(6)
 ...
```

- Click Idle's `File` menu, select `Save`

- Click Idle's `Run`, select `Run Module` (or use `F5`).
- Look at the `Python Shell` window for your results.
- Debug `fib` in `idle` until it is correct. When something goes wrong, read the error message carefully to what the problem was. If it doesn't make sense to you, ask a staff member for help.

Wk.1.3.1 Check your results by copying the text of your procedure from `idle` and pasting it into the tutor problem Wk.1.3.1. Submit your answer when it passes all the tests.

4.2 Object-Oriented Practice

Wk.1.3.2 Get some practice with object-oriented concepts in this tutor problem.

Wk.1.3.3 Get some more practice with object-oriented concepts in this tutor problem.

4.3 Two-dimensional vectors

Open file `~/Desktop/6.01/swLab01/sl01Work.py` and complete the definition of the `V2` class; it represents two-dimensional vectors and supports the following operations:

- Create a new vector out of two real numbers: `v = V2(1.1, 2.2)`.
- Convert a vector to a string.
- Add two `V2`s to get a new `V2`.
- Multiply a `V2` by a scalar (real or int) and return a new `V2`.

Step 1. Define the basic parts of your class, with an `__init__` method and a `__str__` method, so that if you do

```
print V2(1.1, 2.2)
```

it prints

```
V2[1.1, 2.2]
```

Exactly what gets printed as a result of this statement depends on how you've defined your `__str__` procedure; this is just an example. Remember that `str(x)` turns `x`, whatever it is, into a string. Don't worry about making this beautiful!

Step 2. Write two accessor methods, `getX` and `getY` that return the `x` and `y` components of your vector, respectively. For example,

```
>>> v = V2(1.0, 2.0)
>>> v.getX()
1.0
>>> v.getY()
2.0
```

Step 3. Define the `add` and `mul` methods, so that you get the following behavior:

```
>>> a = V2(1.0, 2.0)
>>> b = V2(2.2, 3.3)
>>> print a.add(b)
V2[3.2, 5.3]
>>> print a.mul(2)
V2[2.0, 4.0]
>>> print a.add(b).mul(-1)
V2[-3.2, -5.3]
```

Step 4. A cool thing about Python is that you can overload the arithmetic operators. So, for example, if you add the following method to your `V2` class

```
    def __add__(self, v):
        return self.add(v)
```

then you can do

```
>>> print V2(1.1, 2.2) + V2(3.3, 4.4)
V2[4.4,6.6]
```

Add to the class the `__add__` method, which should call your `add` method to add vectors, and the `__mul__` method, which should call your `mul` method to multiply the vector by a scalar. The scalar will always be the second argument.

Test your implementation in `idle` until it seems correct to you.

Wk.1.3.4 Check your results by copying the text of your procedure from `idle` and pasting it into the tutor problem Wk.1.3.4.

6.01SC Introduction to Electrical Engineering and Computer Science

Fall 2011