PIC 16, Winter 2018 – Preparation 3M

Assigned 1/19/2018. To be completed by lecture 1/22/2018.

Intended Learning Outcomes

By the end of this preparatory assignment, students should be able to:

- import definitions from standard modules,
- distinguish between syntax errors and exceptions, and
- `try` commands that might raise exceptions and handle `except`ional cases when they arise.

Tasks

- ☐ Read the introductory text of Python Tutorial Topic 6 (Modules). I find it a little confusing despite the concepts being quite simple. Here are a few thoughts that might help:
  - You've been writing scripts (`.py` files) in Spyder since the beginning of the class. I guess the tutorial assumes that you've been writing everything so far in the interpreter (the Python or IPython console typically shown on the right side of Spyder).
  - The "current symbol table" is the list of variable/function names that the Python interpreter knows about. Before you define a variable, say `x = 1`, you can't write `print x` because the interpreter doesn't know what `x` is. When you define `x = 1`, `x` is added to the "current symbol table". The point the tutorial is making is that even after you `import fibo`, you still can't call `fib(100)` because `fib` is not added into the "current symbol table". Only `fibo` is added. In order to use `fib`, you have to call it like `fibo.fib(100)`.
  - I recommend that you do what the tutorial suggests – actually create the `fibo.py` file, try `import`ing it in the interpreter, and try using its functions. See what happens if you try to call the functions like `fibo.fib(100)` *before* importing the module[1], and see what happens if you try to call the functions like `fib(100)` even after importing the module. (Those won't work.)
- ☐ Predict and check the output of the following in a fresh instance[1] of the python interpreter (so that the `math` module has not been imported previously). You can think of `pi` as a "global" variable declared in a file `math.py` like `pi = 3.14159265`.

```
print pi
print math
print math.pi
import math
print pi
print math
print math.pi
import math as m # This is not covered in the text yet.
print m.pi       # It's convenient when packages have very long names.
```

- ☐ Read 6.1. You can skip 6.1.1 - 6.1.3 for now. You can come back if you want to write your own modules, but you won't need to do that for this class.
- ☐ Predict and check the output of the following in a fresh instance[1] of the Python interpreter (so that the `math` module has not been imported previously):

```
print sin(pi)
```

[1] You can get a fresh instance of the interpreter by selecting "Open an IPython console" from the Consoles menu.

```
from math import pi
print pi
print sin(pi)
print math.sin(pi)
from math import *
print sin(pi)
print cos(pi)
```

- Watch the video at https://youtu.be/IJ7iurivK9g if you could use another example.
- Read 8.1 - 8.4. Know how syntax errors and exceptions are reported in the console, the difference between them, and how `try-except` statements work. You'll need to `raise` an exception when we start working with iterators, but you won't need this immediately for your next assignment.
- Note that exception handling (the `try-except` construct) in Python is not as costly as in other languages, and so some programmers would argue that `try-except` statements should be used relatively freely - even in many places `if-else` statements would be used in other languages: https://www.jeffknupp.com/blog/2013/02/06/write-cleaner-python-use-exceptions/.
- If you want to write bulletproof, maximally efficient, and fully "Pythonic" (http://blog.startifact.com/posts/older/what-is-pythonic.html) code, you should probably read the rest of 8. However, it is unlikely that we will need to write custom exceptions, or use the `finally` keyword in this class, so it is OK with me if you skip the rest of 8 (from 8.5 on).
- Watch the video at https://youtu.be/ws_dFAUd9es if you'd like to see an example.
- No exercises corresponding with the exception handling material (8.1 – 8.4) are assigned here. The syntax of it is quite simple once you understand the concepts, any you'll get practice with both during your next assignment.