Numerical Methods

Math 3338 – Spring 2022

Worksheet 1

Strings, Modules and Homework

1 Reading

CP	2.2.5, 2.2.6, 2.4
NMEP	1.3, 1.4, 1.5

Table 1: Sections Covered

2 Brief Functions

First, did you create a new project for this worksheet and homework? Do that. Now. Don't just put all the files for this course in a single folder[1].

Here is a really basic function,

```
def f(x):
    return x**2
```

In this function, x is the input and the output is x^2, which is computed as $x**2$ in Python. You can have multiple inputs, we'll discuss in more detail in worksheet 2.

3 Strings

Occasionally you have a lot of data that you want to display nicely. For example, let's say we have a function,

```
def f(x):
    return 3*x-x**2
```

This is a simple example, but it'll be nice to use. Let's look at this function evaluated at a few points.

```
L = [1,2,3,4]
for i in L:
    print(f(i))
```

The output to this is pretty basic, a line of evaluated points. But we don't know the inputs. There are a few ways to do this.

```
L = [1,2,35,400]
for i in L:
    print(i,f(i))

print('-'*50)

for i in L:
    print("i={0}, f(i)={1}".format(i,f(i)))

print('-'*50)
```

[1]If I find out everything is on your desktop, I may fail you.

```
for i in L:
    print("i={0:3d}, f(i)={1:4f}".format(i,f(i)))
```

There are a few things going on here, try to analyze which you like best. The important thing to notice is the *format* function. The short story is that any {0} in the string gets replaced by the zeroth argument in format. Format can also specially format things, as seen in the last example, for the i, it's reserving 3 digits and for the $f(i)$ 4 float decimal digits. This makes tables much easier to read. Try adding 1402 to L, or a float. See what happens.

Format is way more useful than just this.

```
print("Hello {name}".format(name = "Mitch"))

d = {"name":"Mitch"}
print("Hello {name}".format(**d))

name = "Mitch"
print(f"Hello {name}")
```

The second one, with **d *unpacks* the dictionary which makes it look like the first. The third uses a new *format string*, this knows what variables have been defined and inserts them.

Another useful function is "join". Try the following command,

```
", ".join(['a','b','c','d','eagle'])
```

The output will be

```
'a, b, c, d, eagle'
```

The elements of the list get "joined" with the ', '. The only really important thing to know is that elements of the list must be strings. If they are numbers, you'll need to call 'str(i)' with a list comprehension.

4 Modules

Go into iPython and type

```
help()
```

This opens, surprisingly, the help menu. You can read what it says, but we're going to type *modules*. This will generate a list of all the modules currently installed on your system.

4.1 Math

The standard arithmetic operations ($+$, $-$, $*$, $/$) work just fine. Try this,

```
print(2+3)
print(2-3)
print(2*3)
print(2/3)
print(2^3)
```

Notice, the last one did not give us 8. That's because ˆ is an xor statement. To get a power use

```
print(2**3)
```

Other useful operations include // and %. // is the integer portion of division, or floor(a/b). and % is the remainder.

```
print(5//3)
print(5%3)
```

As a math statement,

$$b = a \cdot (b//a) + b\%a.$$

Let's focus on the math module. To load a module you import it.

```
import math
```

Always put inputs at the top of the file. It's the standard location, cleaner and makes it easy to identify what your dependencies are. Let's see what the math module has to offer.

```
help(math)
```

You'll see a ton of math functions and some constants. One constant is *pi*. Let's see it's value

```
print(pi)
```

And error. Pi belongs to the math module so to use it, we need to tell Python where to look.

```
print(math.pi)
```

Now we get π. This is cool and all, but what if our module has a really long name, for example *matplotlib*. We would have to type this in every single time we used a function. Two options,

```
import math as m #now you can type m.pi

from math import * #this will load everything from math into the current namespace
```

The first option is safer, you won't accidentally overwrite something important. The second can be dangerous if the modules have two functions with the same names, you get conflicts. You can also selectively load functions by doing

```
from math import sqrt
```

now you can type `sqrt(4)` rather than `math.sqrt(4)`.

Let's try something. Before you run this, think about what it probably does and what will be printed. Type the following,

```
from math import sqrt

a = sqrt(2)
print(a**2)
```

4.2 time

At times it will be useful to benchmark our operations, or figure how long things take to run. That's where the time module comes into play.

Here is a little class that you can steal. It's basically a timer. Here is an example,

```
import time

class Timer(object):
    def __init__(self):
        self.start = time.time()

    def stop(self):
        out = time.time() -self.start
        self.restart()
        return out

    def restart(self):
        self.start = time.time()

if __name__ == "__main__":

    t = Timer()

    time.sleep(5) #sleep for 5 seconds. Typically this will be something interesting.

    print("This took {0} seconds".format(t.stop()))
```

This should have taken a little over 5 seconds. This will be in the class module for you to use whenever.

4.3 Pickle

Pickle is a module to save and load binary data. Pickle also preserves Python data structures, like lists, dictionaries, tuples, etc. Here is an example of loading data with pickle,

```
import pickle

with open("data.pickle","rb") as d:
    data = pickle.load(d)
```

The variable "data" now has whatever was in the "data.pickle" file. We'll discuss the "with" part of the code in Worksheet 02.

4.4 Custom Modules

Create a new file called "module_test.py"[2]. Create a couple of functions inside this file,

```
import math

def f(x):
    "Return sin(x) with x in radians"
    return math.sin(x)

def g(x):
    "Return sin(x) with x in degrees"
    return math.sin(x*math.pi/180)

if __name__ == "__main__":
    #This is where you can test your functions to ensure they work.
    pass
```

You can import this file in your original.

```
import module_test as mt

print(mt.f(1))
```

[2]Make sure it's in the same folder as your main file. You created a new project, right?

```
print(mt.g(1))
print(mt.math.pi)

help(mt)
```

Notice, we can use the math module that was imported in module_test, this is one reason to be careful with imports, it can slow down Python even farther. This isn't catastrophic impact for us, but you should be aware of it for a large project. Also, due to the way we commented, the help function just works.

5 iPython

This is a wonderful tool for testing things. In your Python file enter and run the following,

```
def test_function(input):
    return 3*input-8
```

In the iPython console, type

```
test_function(1)
```

it should print -5. iPython remembers things which makes it incredibly useful for testing. iPython has many, many features we won't get into here, but feel free to look them up. In particular there is a handy timer that does some magic things.

6 Submitting Homework

Homework is submitted as a Python file ".py" directly on Canvas. The file should contain only solutions to the homework and any dependecies. For example, if there were two problems one to write a function to compute the square root of a number an another to print a formatted string. You would submit

```
from math import sqrt

def problem_1(x):
    return sqrt(x)

def problem_2(input_string):
    out = "The input is: {0}".format(input_string)
    return out

if __name__ == "__main__":
    #you can test stuff here. It won't run if the file is imported.
    pass
```

The problems will tell you what to name the functions, what inputs should be expected and what output is required. There will be a provided data file as well, so you can check your answers before submission.

To grade these, I will download the files and test against a large dataset. I'll be looking for correctness and if there are errors. I will also examine your code to ensure things are working well.

It will be easy to copy code from another source or students. I'll know and the penalty will be a 0 for all parties.

Sometimes you'll also need to upload PDFs. It'll be clear when and you'll be using LaTeX to create these PDFs.

Numerical Methods

Math 3338 – Spring 2022

Homework 01 (Due: Thursday, January 13)

Problem 1 (1 pt) Write a function called `fizzbuzz`. There should be one input, a positive integer n and output a string. The function will iterate over all integers from 0 to $n-1$ if the integer is a multiple of 3, add fizz to the output, if it's a multiple of 5, add buzz and if it's a multiple of both 3 and 5, print fizzbuzz. Don't forget the newline character "`\n`" at the end of each line.

Here is some sample output if $n = 16$,

```
fizzbuzz
fizz
buzz
fizz
fizz
buzz
fizz
fizzbuzz
```

Problem 2 (1 pt) Write a function called `latex_table`. The input will be a list of lists, representing the rows of the table. The output is a properly formatted LaTeX table.

```
\begin{tabular}{ccc}
$a$ & $b$ & $c$ \\
1 & 2 & 4 \\
3 & 5 & 6
\end{tabular}
```

A couple of things that may be useful,

- To get a literal { in a Python3 format string, you need to use two, {{.
- To get \ in a Python3 string, you need to use two, \\.
- New lines are the special character `\n`.
- The string method *join* will probably be useful. Feel free to look it up.

You'll find this function to be incredibly useful in future homework.

Problem 3 (1 pt) This problem will teach you how to use pickle and how to check answers against provided datasets. For this problem you'll be using the file "hw_01-problem_03.pickle" on the Canvas page. This file contains a list of lists.

Create a function called `list_stats`. This funtion will take 1 input, a list of lists, and output a list of tuples of the form [(sum of list, min of list, max of list)]. For example,

```
list_stats([[1,2],[5,-4,3]])  ->  [(3,1,2),(4,-4,5)]
```

To check your output, use the following code (you'll need to load both data and sol).

```
#data -> Data from hw_01-problem_03.pickle
#sol -> Data from hw_01-problem_03-sol.pickle
test_sol = list_stats(data)
for a,b in zip(test_sol,sol):
    if a!=b:
        print("Error: {0} {1}".format(a,b))
```