1 Administrative Topics

- Look at example write-ups from Project 1.
- I will assign homework tonight. If you email me your answers by Thursday night at 10:00, I will respond with comments. The homework is designed to help you study for the quiz.

2 Organizing Code

A Python program written for this class should be organized as follows:

1. Header comments (name of file, date, name of author)
2. Any import statements
3. Any function definitions (each of which should have a comment indicating what it does)
4. A comment indicating the “main” section of code is about to appear
5. The main code that you want this program to run

For example, a simple Python program that uses the turtle to draw two squares is this:
def drawSquare(length):
    """ Draw a square with side of the given length """
    turtle.color((0,0.5,0))
    turtle.forward(length)
    turtle.left(90)
    turtle.forward(length)
    turtle.left(90)
    turtle.forward(length)
    turtle.left(90)
    turtle.forward(length)
    turtle.left(90)

# main
# set-up
turtle.reset()
turtle.tracer(False)

# draw two squares
len = 30
drawSquare(len)
turtle.right(30)
drawSquare(100)

# done drawing, make sure it appears on the screen and stays there
turtle.update()
raw_input("press enter when ready ")

2.1 Comments

There are two ways to annotate code in Python.

1. docstrings: To describe the purpose of a function, you should put a
   triple-quoted string as the first line of the function’s body. In this doc-
   string, specify what the function does and what each of the parameters
   means.

2. comments: To otherwise provide information to the reader of your code,
use a hash mark. Everything that appears after the hash mark (up to the end of the line) will be ignored by Python. Use comments to identify the purpose of blocks of code or to provide any other information that would help someone who wanted to update the code.

3 Understanding what happens when Python executes code

The next example is meant to deepen our understanding of symbol tables and the process of execution. The code calculates and prints the amount you should tip on two different restaurant bills:

```python
# Stephanie Taylor
# February 8, 2017
# Lecture 4 example to illustrate symbol tables
# and functions.

def printTip(bill):
    """ Print a message telling us the amount to tip for the given bill ""
    rate = 0.18
    print('You should tip $' + str(bill*rate) + ' when the bill is $' + str(bill))

# main code
bill1 = 20;
printTip(bill1)
bill2 = 120.50
printTip(bill2)
```

3.1 Stepping through code

Let’s step through the code first, then think about memory. The basic approach is this: Python reads the file from the beginning to the end, executing the code as it goes along.

- Python reads the definition of printTip and stores it for use later. Now Python “knows” what the function printTip is.
• Python executes line 12, which is an assignment statement. Now Python “knows” what the amount of the first bill is.

• Python executes line 13, which means it needs to execute lines 8 and 9. A message is printed to the screen as a result of the code in line 9. It is

You should tip $3.6 when the bill is $20

• Python executes line 14, which is an assignment statement. Now Python “knows” what the amount of the second bill is.

• Python executes line 15, which means it needs to execute lines 8 and 9. A message is printed to the screen as a result of the code in line 9. It is

You should tip $21.69 when the bill is $120.5

• There is no more code. So Python stops executing.

3.2 How to view memory when functions are being executed

Now, let’s look at what happens to memory as this code is executed. To do that, I am going to draw symbol tables. We talked on Monday about symbol tables. We learned that there is a main symbol table when you are running the interpreter. We also learned that each module has a symbol table. Today, we are going to add to that: Each function has a symbol table while it is executing.

Here is how I will draw this. We will be executing “main” code and code within a function. I am going to write the line of code that is currently being executed underneath the appropriate symbol table.

Let’s step through the code again.

• Python reads the definition of printTip and stores it for use later. Now Python “knows” what the function printTip is. So let’s assume we are at line 11, which has no code, so I won’t write anything underneath the table.
Python moves to line 12, which I will write underneath the table (think of this as a "before" picture)

```
main
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>printTip</td>
<td>→ &lt;fcn&gt;</td>
</tr>
</tbody>
</table>
```

```
bill1 = 20
```

Python executes line 12, which is an assignment statement. Now Python “knows” what the amount of the first bill is. (think of this as an “after” picture – I am striking out the line of code to show that it has completed executing)

```
main
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>printTip</td>
<td>→ &lt;fcn&gt;</td>
</tr>
</tbody>
</table>
| bill1 | 20 <int> 
```

```
bill1 = 20
```

Python executes line 13. This will be a multi-step process:

- Python is about to call the printTip function. The first thing it does is look evaluate the code between the parentheses. It is going to pass (or send, or input) a VALUE to printTip. (think of this as a “during” picture - I am crossing out the variable name and replacing it with its value to indicate that Python looked up the variable’s value in the table).

```
main
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>printTip</td>
<td>→ &lt;fcn&gt;</td>
</tr>
</tbody>
</table>
| bill1 | 20 <int> 
```

```
printTip( bill1, 20 )
```
Now Python begins executing `printTip`. To do that, it creates a symbol table for it, and makes entries in the symbol table for each of the parameters.

```
printTip
Name  Value
printTip  → <fcn>
bill1    20 <int>
```

`printTip( bill1 20 )`

We move to line 8 and execute it, which results in a new row in the `printTip` symbol table.

```
main
Name  Value
printTip  → <fcn>
bill1    20 <int>
```

`printTip( bill1 20 )`

We move to line 9 and execute it. This line requires Python to look up value in the table, but does not make any changes to the tables, so I am not doing to redraw the tables. The output is

```
You should tip $3.6 when the bill is $20
```

Python is now done executing the function so its table goes away. And I will strike-out the code below the main function to show that the line has completed executing.

```
main
Name  Value
printTip  → <fcn>
bill1    20 <int>
```

- Python executes line 14, which is an assignment statement. Now Python “knows” what the amount of the second bill is.
Python executes line 15. This will be a multi-step process:

- Now Python begins executing printTip. To do that, it creates a symbol table for it, and makes entries in the symbol table for each of the parameters.

- We move to line 8 and execute it, which results in a new row in the printTip symbol table.

- We move to line 9 and execute it. This line requires Python to look up value in the table, but does not make any changes to the tables, so I am not doing to redraw the tables. The output is You should tip $21.69 when the bill is $120.5

- Python is now done executing the function so its table goes away. And I will strike-out the code below the main function to show that the line has completed executing.
```
\begin{tabular}{|l|l|}
\hline
\textbf{Name} & \textbf{Value} \\
\hline
printTip & → <fcn> \\
bill1 & 20 <int> \\
bill2 & 120.5 <flt> \\
\hline
\end{tabular}
```

- There is no more code. So Python stops executing.