1 Administrative Topics

- Any questions about the project?
- Go over procedure for turning in code:
  1. Mount the CS152 directory on the Courses server (This is the command-K thing in instruction 6 of the lab) (smb://filer.colby.edu/Courses)
  2. Navigate to your directory.
  3. Navigate to the Private directory in your directory.
  4. Make a Proj1 directory.
  5. Copy all of your .py files to this new Proj1 directory. That is where we will look for it when we are ready to grade it.

2 Python and its memory model

To learn how to program effectively in Python, we must understand how the language is storing and manipulating data – we must understand its “memory model.”

I think of Python programs as a sequence of commands that manipulate data in memory.

To learn Python, we need to learn two things:
1. What are the legal Python commands?

2. How does Python view memory? I.e. what is Python’s memory model?

On Friday, we wrote a function in Python. That enabled us to give a name to a collection of statements. Today, we are going to look at the ability of Python to name some data. We do so through variables and parameters. Let’s look at the code from Monday again:

```python
# Convert the temperature from Fahrenheit to Celsius
def convertFahrenheitToCelsius(temperatureF):
    temperatureC = (temperatureF - 32)*5.0/9
    return temperatureC
```

I want to focus on `temperatureC`. It is a variable. A variable is a name that is associated with a particular value, such as the result of a computation. It allows us to access that value later, using the name.

Let’s use the Python interpreter to explore simple calculations and variables. First, I want to create a named variable and associate some particular value with it. I do that with an assignment statement.

```python
a = 3 + 4
```

This line of code is calculating the sum of 3 and 4, then placing that result into memory and labeling it as `a`.

We can retrieve the value from memory using the label `print a`

Python programs generally consist of many statements that store and examine variables. And these variables are stored in tables. So, we can think of the basic memory model for Python as manipulation of tables.

In these tables, Python stores the names of values of all the variables. Values are things like numbers or strings. There are four “simple” types of data that Python manages:

1. Integers (`<int>`): whole numbers, e.g. -23, 0, 12324, 8

2. Floating Point Numbers (`<float>`): numbers with decimal points in them, e.g. 1.0, 5.3, 0.1
3. Booleans (<bool>): True or False

4. Strings (<str>): strings of characters, e.g. “hello”, or ’hi’

Python has different operations that apply to different types. For example, 32 + 4 performs math and results in 36. But for strings, the operation is concatenation ’32’ + ’4’ results in ’324’. And ’32’ + 4 results in an error because it doesn’t make sense to add the number 4 to a string.

2.1 Assignment statements store data in tables

When we run Python in the interpreter, there is a “main” table (I often call this a “symbol table” because it keeps track of all of the symbols/labels). Commands manipulate the entries in this table. This table has two columns: Name and Value.

One example of this manipulation is by the assignment statement. An assignment statement associates data with a label, or variable. It does this by adding a row to the symbol table.

Then we execute

miloWeight = 32

to store the fact that my son Milo weighs 32 pounds.

Note that this is not the same as “equals” in mathematics. Whenever you see this “equals” sign, you should say to yourself “this is an assignment statement”. We are assigning the value 32 to the variable miloWeight. I always say “gets”, e.g. “miloWeight gets 32”.

When this line is executed, Python checks the symbol table. If it contains an entry named miloWeight, it will update its value. If it does not contain an entry named miloWeight, it will add a row with the name miloWeight and value 32, e.g.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>miloWeight</td>
<td>32 &lt;int&gt;</td>
</tr>
</tbody>
</table>

I will illustrate the remaining points by extending this example. This example is constructed to demonstrate that data is just data. It is your responsibility as programmer to make sure the data are updated appropriately.
Now execute
bigEnough = miloWeight > 40

Python compares the value of miloWeight to 40 (the minimum weight requirement for using a booster seat instead of a full car set), finds that 32 is not greater than 40. I.e. the statement that 32 > 40 is False, so result is the boolean value False. This value is assigned to the variable bigEnough and the table is updated:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>miloWeight</td>
<td>32 &lt;int&gt;</td>
</tr>
<tr>
<td>bigEnough</td>
<td>False &lt;bool&gt;</td>
</tr>
</tbody>
</table>

Suppose Milo grows. We change his weight to reflect that growth:
miloWeight = 42

and the value of miloWeight changes in the table:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>miloWeight</td>
<td>42 &lt;int&gt;</td>
</tr>
<tr>
<td>bigBoy</td>
<td>False &lt;bool&gt;</td>
</tr>
</tbody>
</table>

Is Milo big? Well, yes, he is big enough now. But the variable bigEnough is still False. The entry in the table is just data – there is nothing in the table that tells us how we got that value – their is nothing in the table that indicates one variable depends on another. It is up to us, as the intelligent humans to update it.

So we again execute
bigEnough = miloWeight > 40

and the table becomes

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>miloWeight</td>
<td>42 &lt;int&gt;</td>
</tr>
<tr>
<td>bigEnough</td>
<td>True &lt;bool&gt;</td>
</tr>
</tbody>
</table>

### 2.2 Rules for Assignment

Here is a more formal description of what Python does when it executes an assignment statement.
1. Evaluate the right-hand-side – this means execute any code on the right-hand side of the equals sign. The result will be the value entered into the table.

2. Update the table

   (a) If there is already an entry for the variable on the left-hand side, then update the value for that row.

   (b) Otherwise, add a row to the table.

### 2.3 Rules for Variable Names

A variable name can contain upper case letters (A through Z), lower case letters (a through z), decimal digits (0 through 9), and the underscore character.

A variable name must begin with a letter (upper or lower case).

Example variable names include, A, pic1, pic2, my_string, myString, etc.

Note that this means that the statement “6=A” is illegal in Python.