CSC 108H: Introduction to Computer Programming

Summer 2011

Marek Janicki

Administration

- Questions on the assignment at the end.
- There were some questions about the memory model from last week.
  - Accordingly, there will be a file posted to the lectures page that will go through the example line by line.
  - When it is posted, will be posted on the Announcements.

Assignment 1

- Find_factor()
  - Between 0 and 100 is exclusive, means 1<=x<=99
  - We are counting non-prime factors, so 4 is an acceptable answer.
  - 1 is not considered a factor, unless the input is 1.
- Find intercept()
  - Don't need to worry about lines that don't have intercepts.
- In general, don't need to worry about inputs for which there is no answer.

What if we don't want numbers?

- So far we've seen ints, reals and booleans
- Allow for number manipulation and logic manipulation
- But what if we want to use text?
- Then we need to use a new type – strings.

Strings

- Sequences of characters.
- Two types `str` and `unicode`.
  - We'll use `str` in this course.
  - It contains the roman alphabet, numbers a few symbols.
  - Unicode is larger, contains more accented letters, Chinese characters, and more.
- Strings are denoted by single or double quotes.
  - Quote type must match!

String operations.

- Strings can be 'added'.
  - We call this concatenation.
  - `“str” + “ing”` results in `“string”`.
- Can also be multiplied, sort of.
  - You can't multiply a string with itself, but the multiplication operator functions as a copy.
  - So `“copy”*3` results in `“copycopycopy”`.
- Can also compare strings using relational operators.
- Can check if substrings are in a string using `in`.
- Long strings that span multiple lines can be made using '''.

Escape Characters

- Denoted by a backslash, they indicate to python that the next character is a special character.
  - \n - a new line
  - \' - a single quote
  - \” - a double quote
  - \\ - a backslash
  - \t - a tab.
- Aside `len(string)` will return an int that is the number of characters in the string.

Converting types to strings.

- If we have a variable that is not a string and we want to add it to a string, we need to convert it.
- We use str(x) to convert x to a string.
- Print will display the variable, and can display mixed types.
  - They must be separated with a comma.
  - `print “string”, x, “ “, real_num`
- Can be awkward.
  - `print “Person“, name, “has height”, height, “age“, age, “weight“, weight`

Can use string formatting instead.

- Can use special characters to tell python to insert a type into a string.
- `print “My age is %d.” % age`
- The %d tells python to take age, and format it as an integer.
- %s says to take a value and format it as a string.
- %f says to take a value and format it as a float.
- %.2f says to pad the float to 2 decimal places.

Multiple variables.

- What if we want multiple variables in our string?
  - `print “Person“, name, “has height”, \`
    `height, “age“, age, “weight“, weight`
- We put them in parentheses separated by commas.
  - `print “Person %s has weight %.2f \`
    `and age %d and height %d.“ \`
    `% (name, weight, age, height)`

User input.

- Here we mean the user as the person who is using a program while it is running.
- Thus far, the only way we've had of giving input to a program is to hardcode it in the code.
- Inefficient and not user-friendly.
- Python allows us to ask for user input using raw_input().
- Returns a string!
  - So it may need to be converted.

Break, the first

Modules.

- Sometimes we want to use other people's code.
- Or make our own code available for use.
- It's convenient if we can bundle up related functions in one file.
- Modules allow us to do this.
- A Module is a group of related functions and variables.

Using modules.

- To use a module, one needs to `import` it.
- Importing a module causes python to run each line of code in the module.
  - It it is just function definitions this doesn't cause much trouble.
  - But it can be annoying if there is code that you don't care about in the module.
- To use a function in a module one uses.

```
module_name.function_name()
```

  - We can also run a module. Then we just use `function_name()`

__name__

- In addition to variables that are defined in the module, each module has a variable that is called `__name__`.
- If we import a module called `module_m`, then

  `module_m.__name__ == “module_m”`
- But if we run a module, then
  - `__name__ == “__main__”`
- Recall that if we are running a module, we don't need the module name as a prefix.

Another way to import things.

- `from module_name import fn_name1(), fn_name2()`
  - Will import fn_name1 and fn_name 2
  - Can be referred to by just `fn_name1()`
- Can also use * as a wildcard to import all the functions.
  - `from module_name import *`
- What if two modules have a function with the same name?
- The most recent one stays.

Methods.

- We've seen that modules can have their own functions.
- A similar thing is true of values.
- Values contain ways that you can modify them. We call these methods.
- These are called by value.fn_name()
- Or, if we've assigned a value to a variable we can use variable_name.fn_name()
- We can call help(type) to figure out what methods a type has available to it.

String methods.

- Can find them by using help(str).
- Useful ones include:
- `s.replace(old, new)` - return a new string that is a copy of s with all instances of old replaced by new.
- `string.count(substr)` – return the number of instances of substr in the string.
- `string.lower()` - shift to lower case letters.
- `string.upper()` - shift to capitalised letters.

Getting method information

- Most direct way is to use help().
- But help isn't searchable. Can use dir() to browse.
  - Sometimes you know what you want, and you think it might already exist.
- An alternative is to check the standard library:
  - http://docs.python.org/library/
  - Being able to browse this is useful skill.

Remember!

- Functions belong to modules.
- Methods belong to objects.
  - All of the basic types in python are objects.
  - We will learn how to make our own later.
  - This is covered in greater detail in 148.
- len(str) is a function
- str.lower() is a method.
- Subtle but important distinction.

Break, the second.

Repetition

- Often times in programs we want to do the same thing over and over again.
- For example, we may want to add a number to a variable until it reaches some number.
- Or we may want to execute a block of code until some condition is true.
- Ages ago, this was done with a goto statement.
  - This lead to unreadable 'spaghetti' code.
  - Python has no goto statement.

The while loop

- Instead Python uses loops.
  - We will cover the `for` loop next week.
- The while loop has the form:

```
while condition:
    block
```

- The condition is checked first. If it evaluates to True, the block executes, otherwise the block is skipped, and the next line of code is executed.

Why loops?

- While loops can be used if:
  - You want to repeat an action indefinitely
  - You want to repeat an action until a condition is met.
  - You want to repeat an action a fixed number of times.

Assignment Questions