Computer Science 1 — CSci 1100
Lecture 5 — Modules, Objects, Methods, Images

Reading

- Material for this lecture is drawn from Chapter 4 of *Practical Programming*.
- We will concentrate on making use of existing modules.
- We will NOT use the `media` module discussed in the text, opting instead for the more widely-used `PIL` (Python Imaging Library), comprised of multiple modules.
- We will also talk briefly about writing our own modules, returning to this more later in the semester, mostly during labs.

What is a Module?

- A collection of Python variables, functions and objects, all stored in a file
  - We'll define the notion of an object soon, but strings, ints and floats are all (built-in) objects.
- Modules allow code to be shared across many different programs.
- Before we can use a module, we need to import it:

```
>>> import module_name
```

- Importing makes the functions in the module available for use.
- Python has many, many modules to accomplish many different tasks, ranging from manipulating images to accessing Twitter feeds.

The Math Module

- An important example is the `math` module:

```
>>> import math
>>> math.sqrt(5)
2.2360679774997898
>>> math.trunc(4.5)
4
>>> math.ceil(4.5)
5.0
>>> math.log(1024,2)
10.0
>>> math.pi
3.1415926535897931
```

- We can get an explanation of what functions and variables are provided in a module using the `help` function

```
>>> import math
>>> help(math)
```

- We will work through an number of examples of using the `math` module in class.

Different Ways of Importing

- A selection of functions and variables:

  ```
  >>> from math import sqrt,pi
  >>> pi
  3.141592653589793
  >>> sqrt(4)
  2.0
  ```

  or everything (which is NOT recommended!):

  ```
  >>> from math import *
  ```

- Using a short-hand:

  ```
  >>> import math as m
  >>> m.pi
  3.141592653589793
  >>> m.sqrt(4)
  2.0
  ```

Exercise Set 1

1. Write Python code to calculate the distance between two points whose coordinates are stored in the four variables `x0`, `y0`, `x1`, `y1`.

2. What happens when we type

   ```
   import math
   math.pi = 3
   ```

   and then use `math.pi`?

A Bit on Writing our Own Modules

- We can add understand a bit better what is happening by writing our own modules, gathering functions (and variables, if any) that share a common purpose into a file.

- For example, we might create a file called `cylinder.py` that contains

  ```
  import math

  def area_circle(radius):
      return math.pi * radius ** 2
  ```

```
def volume(radius,height):
    return area_circle(radius) * height

def surface_area(radius,height):
    circle_area = area_circle(radius)
    height_area = 2*radius * math.pi * height
    return 2*circle_area + height_area
```

- When we import this we leave off the `.py`

  ```
  import cylinder
  ```

- In class, we will write code to use `help`, and run the functions in our new module.

Exercise Set 2

1. Write a temperature conversion module that includes a function to convert from Fahrenheit to Celsius and a second function to convert from Celsius to Fahrenheit. Save this to a file. (If you are writing solutions by hand, write down the name that you would have used.)
2. Write code to import and use one of the functions
3. Add variables to your temperature conversion module that store the freezing point and boiling point of water. Show how these variables can be printed.

Built-In Modules

- Type

  ```
  >>> help(__builtins__)
  ```

  to see all of the modules Python has built-in. These do not need to be imported.

- We can get help on any number of these. We'll start with `int`, `float` and, especially, `str`.
  - These are our first (of many) Python "objects"

Objects and Methods

- All variables in Python are objects.
- Objects are abstractions:
  - They have a specific organization and structure to the data they store.
  - They have operations/functions — we call them **methods** — applied to access and manipulate this data. specific to them.
- We will focus on strings.

Some str Methods

- Operators, including +, <, and >

```
>>> s0 = "Hello"
>>> s1 = "hello"
>>> s2 = "good-bye"
>>> s0 < s1
True
>>> s1 < s2
False
>>> s1 + ', ' + s2
'hello, good-bye'
```

- Methods:

```
>>> s = "GeorGe WasHINgton"
>>> s.lower()
'george washington'
>>> s.upper()
'GEORGE WASHINGTON'
>>> s.capitalize()
'George washington'
```

Using help to Find a Complete List of str Methods

- Type

```
>>> help(str)
```

  to see all of the different methods associated with strings.

- Most of these use the dot notation we just examined

```
>>> s = "GeorGe WasHINgton"
>>> t = s.lower()
>>> print t
'george washington'
>>> print s
GeorGe WasHINgton
```

- The general format for employing methods for a class of objects is
  - Variable name or literal (except for integers)
  - Dot .
  - Function name
  - Parentheses, perhaps with arguments
- Notice that, like most `str` methods, this did not change the string but instead created a new string.

- Methods we will examine in class include
  - `capitalize`, `count`, `find`, `lower`, `replace`, `strip`, `upper`
- You may use any methods that you find and understand.

Methods Using Special Python Syntax

- You will notice several that have two underscores at the start and end, such as

```
__add__
__contains__
__eq__
__len__
```

- These names have special Python syntax associated with them:
  - `__add__` is equivalent to `+`,
  - `__contains__` is equivalent to `in`, and
  - `__eq__` equivalent to `==`.
- For example,

```
>>> s0 = "Hello"
>>> s1 = "Good-bye"
>>> s0 + s1
'HelloGood-bye'
>>> s0.__add__(s1)
'HelloGood-bye'
>>> s2 = "Go"
>>> s1.__contains__(s2)
True
>>> s2 in s1
True
>>> s1.__len__()
8
>>> len(s1)
8
```

- We will generally use the "special" syntax, but be aware of the equivalence to the `__` syntax when you are learning to use object methods.

Exercise Set 3

1. Using string operations, write expressions that
   (a) Capitalize `'integer'`
   (b) Find the index of the first occurrence of `'2'` in `'CO2 H2O'`
   (c) Removes all surrounding whitespace from `'  monday  '` and capitalizes the result.

2. Given

```
s = 'Animal'
t = 'Crackers'
u = 'Are'
w = 'Good'
```

write code that uses both `+` and `__add__` to generate a new value stored in variable `s`

```
'Animal crackers are good'
```

PIL — Python Image Library

- PIL is a series of modules built around the `Image` type, our first object type that is not part of the main Python language
  - We have to tell Python about this type through `import`
- We will use images as a continuing example of what can be done in programming beyond numbers and beyond text.

Images

- An image is a two-dimensional matrix of pixel values
- The origin is in the upper left corner
- Pixel values stored in an image can be:
  - RGB — a "three-tuple" consisting of the red, green and blue values, all non-negative integers
  - L — a single "gray-scale" integer value representing the brightness of each pixel

Our First Image Program

- Code is in `image_ex1.py`:

```
###############

import Image

filename = "amy_sheldon.jpg"
im = Image.open(filename)
print '\n' '********************'
print "Here's the information about", filename
print im.format, im.size, im.mode

gray_im = im.convert('L')
scaled = gray_im.resize( (128,128) )
print "After converting to gray scale and resizing,"
```

```
print "the image information has changed to"
print scaled.format, scaled.size, scaled.mode

scaled.show()
scaled.save(filename + "_scaled.jpg")

#################
```

- The `Image.open` function:
  - Opens the file and creates an image (new type, remember), and associates it with the variable name `im`
- `format`, `size`, `mode` are all *variables* associated with the image, describing
  - the image file format — JPG in this case
  - the width and height of an image, as a "tuple" (a new Python type we have not yet used)
  - the `mode` of the pixel data in the image — RGB values or gray scale values (indicated by `'L'`)
    - Note that `'RGB'` and `'L'` are both strings.
- image methods used in this example
  - `convert` to change the mode of the image from RGB to gray
  - `resize` to create a new image of a different size
    - The new size is given by the tuple $(200, 200)$
  - `show` to display the image
  - `save` to output it.

Example 2: Cut and pasting parts of an image

- This example is stored in `image_cutandpaste.py`. It crops two boxes from an image, and pastes them at different locations.

```
import Image

im = Image.open("amy_sheldon.jpg")

# Set the upper left corners of the Amy and Sheldon regions
# Set a fixed height and width
amy_x = 110
amy_y = 105
sheldon_x = 445
sheldon_y = 40
w = 100
h = 100
```

```
# Crop out and reflect Amy's and Sheldon's heads
sheldon_head = im.crop((sheldon_x, sheldon_y, sheldon_x+w, sheldon_y+h))
sheldon_reflected = sheldon_head.transpose(Image.FLIP_LEFT_RIGHT)
amy_head = im.crop((amy_x,amy_y, amy_x+w, amy_y+h))
amy_reflected = amy_head.transpose(Image.FLIP_LEFT_RIGHT)

# Paste the regions back into the image and show it
im.paste(sheldon_reflected, (amy_x,amy_y))
im.paste(amy_reflected, (sheldon_x,sheldon_y))
im.show()
```

- Important image methods used:
  - `crop` — extracts a region of the image given by the upper left and lower right corners represented as a four-tuple
  - `transpose` — flips the image left and right
    - `FLIP_LEFT_RIGHT` is a variable in the `Image` module (see other variables using `help`)
  - `paste` — overwrites one image with another, starting at the upper left corner.
- Note that unlike string methods, many of these actually change the image object we are working with
  - This is an implementation decision made by the designers of PIL, mostly because images are so large and copying is therefore time consuming.
- Images will be studied more in Lab 4 and in future lectures.

Summary

- Modules contain a combination of functions, variables, object definitions, and other code, all designed for use in other Python programs and modules
- After they are imported, the functions in a module can be executed by a call of the form:

  ```
  module_name.function_name(arguments)
  ```

- Objects have associated functions called methods that apply an operation to an object: `object.method(arguments)`.
- `str` objects, a built-in Python object type we already began to study in Lecture 4, have a substantial number of useful methods associated with them
- `PIL` provides a set of modules that define the `Image` object type and associated methods.