CS50 Section. Week 8. 10/25/16.

Tuesdays 4-5:30 PM, CGIS S-040

> Nicholas Boucher nboucher@college.harvard.edu

Important links

- This week's material on Study50: Study50
- C language reference: https://reference.cs50.net/
- CS50 Discuss: https://cs50.harvard.edu/discuss
- CS50 Style Guide: https://manual.cs50.net/style/

Section Agenda

1. Python
    - Variables
    - Conditionals
    - Loops
    - Arrays/Lists
    - Tuples
    - Dictionaries
    - Functions
2. Objects
3. MVC

Python

This week's pset will be in Python. Python is a *high-level* language that has many benefits over C. We will see that programs can often be expressed within Python using a fraction of the code as would be required in C.

Python is an interpreted language, which means that it does not *compile* to machine code like programs written in C. Rather, Python programs are interpreted line-by-line as the user progresses through the program. As such, you do not need to run `make` on Python programs.

Specifically, we will be studying Python 3. If you are looking up examples online, please do not be confused by examples of its slightly older cousin, Python 2.

We will explore Python by discussing the primary programming constructs individually.

Variables

- Variables are declared by assignment
- Variables in Python do not have explicit data types
- Python variables do have underlying data types, namely:
    - number
    - string
    - list
    - tuple
    - dictionary

```
coursenum = 50
```

```
coursename = "Introduction to Computer Science I"
```

Conditionals

- The keywords and, or, and not replace their symbolic representations in C.
- `else if` is replaced by `elif`.
- `switch` statements do not have a clean equivalent in Python and should probably be ignored.
- The code subject to a conditional is introduced by `:` instead of `{`.
- All code subject to the conditional must be indented in order things to work as intended.
- The conditional is terminated by returning to the previous indentation level.
- `print` statements have newlines added by default, unless you override that behavior.
- Comments can be introduced with the `#` symbol.
- Comparators (and assignments) are not necessarily binary operators.

```
y = cs50.get_int()
z = cs50.get_int()
if y == z == 0:
    print("y and z are both zero")
elif y > 0 and z > 0:
    print("y and z are both positive")
else:
    print("At least one of y or z is negative or zero")
```

Loops

- `for` and `while` are the two primary iterating constructs in Python.
- `for`, in particular, has extreme flexibility relative to its C cousin.
- `do-while` does not exist in Python and has to be hacked with a `while True:` and a `break`
- The code subject to a loop is introduced by `:` instead of `{`.
- All code subject to the loop must be indented in order things to work as intended.
- The loop is terminated by returning to the previous indentation level.

```
i = 0
while i < 100:
    print(i)
    i += 1

for j in range(0, 101, 2):
    print(j)
```

Arrays/Lists

- Arrays in Python are called lists.
- Lists are created by assigning a variable equal to an expression in square brackets.
- Methods called on lists can mutate their values, including:
  - `x.append(value)`
  - `x.extend([list])`
  - `x.insert(location, value)`
  - `x.remove(value)`
  - As well as `copy()`, `sort()`, `count()`, and others
- The length of a list can be obtained by calling the `len()` function, passing the list as a parameter.
- Lists can contain values of mixed data types.

- Lists are only ordered if you sort them.

```
grades = [87, 61, 90, 83, 78, 99, 93, 55, 81, 76]

# SORT
grades.sort()

# LEGAL
grades.append(100)

# PRINT GRADES
for grade in grades:
    print(grade)
```

Tuples

- Tuples are a data type in Python for a collection of ordered, immutable data.
- You can compare tuples to a C struct whose contents can never be changed.
- A good example of a tuple is an x, y coordinate
- Tuples can be iterated across like other data types representing collections.

```
three_d_coordinate = (5, 3, 0)

# notice that tuples are ordered as well
for dimension in three_d_coordinate:
    # override default print behavior
    print(dimension, end=" ")

print()
```

```
# a list of tuples
coordinates = [(2, 1, 0), (-1, -5, 6), three_d_coordinate, (

# plot only the surface points (z = 0) but notice that the l
for x, y, z in coordinates:
    if z == 0:
        print("({}, {})".format(x, y))
```

Dictionaries

- Dictionaries are effectively the equivalent of a hash table in C.
- Alternatively, you can think of a dictionary like an array (list) that you can index into using keywords, rather than numerical indices.
- Dictionaries consist of key-value pairs, where the keys are integers or strings, and the values are anything (including other dictionaries, lists, or tuples)
- Dictionaries are created by assigning a set of key-value pairs in curly braces, each set separated by commas, to a variable.
- Dictionaries are like structures where the contents ARE mutable.
- Methods called on dictionaries can mutate their values, including:
    - `x.clear()`
    - `x.update({dict})`
    - `x.keys()`
    - `x.values()`
    - `x.items()`

```
doug = {"name": "Doug", "age": 29, "class": 2009, "Python ex
for key, value in doug.items():
    print(key + "--" + str(value))
```

Functions

- Functions are introduced with the def keyword.
- Functions have names and parameter lists, just like in C.
- Python files are interpreted, not compiled, which means they are read top to bottom, left to right.
- Code does not necessarily, but can be, bound up in a main() function, though that requires special extra syntax.
- Functions do not require a prototype, but do need to be defined before they are called.
- Python functions can return multiple values if need be, and may also return tuples, lists, and dictionaries.

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

base = cs50.get_float()
print(square(x))
```

Miscellaneous Python Topics

- The equivalent of include in Python is import.
- There is no `++` operator in Python, you must use `+= 1`.
- No Python statement ends with a semicolon!
- `//` does integer division (familiar from C).
- `/` does floating point division always.

Objects

Objects are a signature feature of *object-oriented programming*. Objects are very intuitive, and represent a very powerful paradigm on which to build complex applications.

Objects are similar to structures in C in that they have fields. Additionally they have methods which are *functions that are inherently part of that object*, and may only be called directly by those objects.

You define the methods and properties of an object inside of a class. Classes are created using the `class` keyword. Class names conventionally start with a capital letter. At a minimum, a class must contain a method called `__init__`, which sets the initial values of properties in the object. All methods of classes must include the `self` parameter as their first parameter, which is a reference to the object that is invoking the method. When calling a method in a program, however, the self parameter is omitted (it is assumed to apply to the object that is invoking the method in the first place).

The following is an example of an object that we might choose to define:

student.py

```
class Student():

    def __init__(self, name, year="Freshman"):
        self.name = name
        self.year = year

    def endYear(self):
        if self.year == "Freshman":
            self.year = "Sophomore"
        elif self.year == "Sophomore":
            self.year = "Junior"
        elif self.year == "Junior":
            self.year = "Senior"
        else:
            self.year = "Alum"

    def info(self):
        print("{} is a {}".format(self.name, self.year))
```

We might, then, use this object as follows:

sample.py

```
from student import Student

# create two new students, one is a freshman
maria = Student("Maria", "Junior")
newkid = Student("John Harvard")

# everyone graduates at the end of the year
maria.endYear()
newkid.endYear()

# new years, now!
maria.info()
newkid.info()
```

MVC

MVC stands for Model-View-Controller. It is a way to break apart code, especially in large-scale applications, to separate components with specific functionalities. This makes it easy to update, for example, one specific component without changing the others.

These components are:

- Model: data access and storage
- View: how data is displayed to the user
- Controller: the *business-logic* of the application, i.e. the logic operations upon data