October 15 – Python

15 October 2012

Use Python 2.7 from http://python.org/

1. Start menu » Python 2.7 » IDLE (Python GUI)
2. You get a window called "Python Shell" with ">>>" prompt.
3. File » New Window
4. In new window, File » Save and give it a filename that ends with `.py` (this will enable syntax coloring).
5. Type your code into the `.py` file, save
6. Run » Run Module (F5) to run it, interact in Shell window.

Try it with this first program:

```
print "Hello world."
# Lines that start with a "#" are called comments.
# You can type whatever you want there, python ignores them.
```

Variables

Named location to store data. Names cannot have spaces in them. They are case-sensitive (upper/lower). Names can have numbers, but cannot start with numbers —

- **Good:**
  - `quiz3`
  - `average`
  - `amount2pay`
- **Bad:**
  - `quiz 3` (due to space)
  - `3rdquiz` (can't start with number)

It's okay to have underscore character instead of a space: `quiz_3`, `amount_to_pay`

```
x = 14          # evaluate right side of equal sign,
y = x * 2       # put that value in the variable on the left.
x = x - 4
# At this point, x holds 10 and y holds 28.
```

Output

```
print "Testing!"  # Quotes indicate a string of characters (text)
print "x+1"       # Variables, arithmetic not evaluated inside quotes
print x+1         # Performs variable lookup and addition.

print "The answer is", y  # Combine multiple parts on one line.
```

Input

```
name = raw_input("Your name:")  # Prompt in quotes and parentheses.
  # Result of what user typed assigned to variable on the left.
print "Your name is", name

# raw_input returns a string of characters (text).
# If you want a number, convert it using int() or float().
#  int for integer (whole numbers)
#  float for floating-point (decimal numbers)

score = int(raw_input("Enter score:"))
print "Twice that score is", score*2
```

If shell window appears messed up, not responding, try control-C a few times.

Class exercise 1

Try to write a program that will behave something like the following. (The user types in the text after the colon on the first two lines.)

```
Enter your name: Alice
Enter the year you were born: 1984

Alice, you are about 28 years old.
```

Conditions

```
# Python has True and False (Boolean values).
#  3 < 4   produces True
#  3 == 4  produces False (are these equal?)
# Note the use of '==' for equality, not '=' which is assignment

if y < 30:      # colon required
    print "That's small."   # only happens if y < 30.
    y = y + 10

if y < 30:
    print "This does not happen."
else:
    print "This does happen, because y is now big."
```

In addition to less than (<), greater than (>), and equal to (==), you can use less-than-or-equal-to (written in mathematics as ≤ but in Python as <=) or greater-than-or-equal-to (>=).

Extended example

```
temp = float(raw_input("Enter temperature:"))
if temp < 40:
    print "That's cold!"
else:
    if temp > 90:
        print "That's hot."
    else:
        print "That's comfortable."
```

Compound conditions

We previously learned the Boolean operators **and**, **or**, **not**. You can directly apply these in your Python programs. For example, to check whether a temperature is *between* 60 and 80:

```
if temp >= 60 and temp <= 80:
    print "What a mild day!"
```

Class exercise 2

This is an extension of the previous exercise, that computed a person's age based on the birth year (by subtracting from the current year, 2012).

In this one, you should print a message at the end that depends on the *decade* of the user's birth. So, that's done with a condition, or a series of conditions.

Here is one example:

```
Enter your name: Chris
Enter the year you were born: 1973

Chris, you are about 39 years old.
Dig it, man.
```

You can come up with your own messages that characterize each decade, but here are some suggestions:

- 1960s: Groovy!
- 1970s: Dig it, man.
- 1980s: Hey, that's fly.
- 1990s: Phat decade!
- 2000s: Sweet!