Comments are marked with #

Don’t need to define variables

Just do

I = 2

B = false

Etc.

Def() defines a block of code to a name

Def doom()

Print “and the trees spoke to me”

Print “they told me the secrets buried in the deepest parts of the earth”

Print “I listened, and I remember”

10\*2 = 20

10\*\*2 = 100

% is used for modulus

spam = 3 % 2 #returns 1

if statements

elif == ‘else if’

def greater\_less\_equal\_5(answer):

if \_\_\_\_\_\_\_\_: #< that “:” is important! Ends the if?

return 1

elif \_\_\_\_\_\_\_\_:

return -1

else:

return 0

the code

print greater\_less\_equal\_5(4)

runs greater\_less\_equal\_5 using 4 as the answer variable

strings

first\_letter = the\_string[0] #left(a1,1)

functions

the header contains “def” and any parameters “(whatever)” and a colon “:” – very important

calling modules

import module

module.code

from module import (Specific function)

assignment operator

c -= a is equivalent to c = c – a

def trip\_cost(city, days):

total = hotel\_cost(days - 1) + plane\_ride\_cost(city) + rental\_car\_cost(city)

return total

def trip\_cost(city, days):

return hotel\_cost(days - 1) + plane\_ride\_cost(city) + rental\_car\_cost(days)

def trip\_cost(city, days):

return rental\_car\_cost(days) + hotel\_cost(days - 1) + plane\_ride\_cost(city)

lists

Lists are a datatype you can use to store a collection of different pieces of information as a sequence under a single variable name. (Datatypes you've already learned about include strings, numbers, and booleans.)

my\_list[:2]

# Grabs the first two items

Dictionary

menu = {} # Empty dictionary

menu['Chicken Alfredo'] = 14.50 # Adding new key-value pair

print menu['Chicken Alfredo']

# Your code here: Add some dish-price pairs to menu!

menu['Spaghetti'] = 15

del zoo\_animals['Sloth']

del zoo\_animals['Bengal Tiger']

zoo\_animals['Rockhopper Penguin'] = 'Subarctic'

bitwise

print 5 >> 4 # Right Shift returns 0

print 5 << 1 # Left Shift returns 10

print 8 & 5 # Bitwise AND returns 0

print 9 | 4 # Bitwise OR returns 13

print 12 ^ 42 # Bitwise XOR returns 38

print ~88 # Bitwise NOT returns -89