ECS 10

Announcements

- Last project due THURS Dec. 3.
- Make some progress on it this week.
- No lecture Wds! Little review project in lab hours, 11-1 in 2020 SLB.
- Today – adding and controlling complexity with functions.

Last time

- Functions in a module
- Function definition
- Arguments and parameters
- Return values
- Functions in a program

Functions in Programs

- Put function definitions at the top of the file, right after the imports.
- Functions must be defined before they can be used!
- The code in a function is not run until the function is called.
- This is true whether the function is in a separate module or in the file with the program.

Example

```python
def addOne(x):
    x = x + 1
    return x
age = 5
next = add_one(age)
print "I am", age, "and soon I will be", next
```

Local variables

- The variables inside a function are invisible to the main program.
- Variables created inside a function are called local variables.
- The parameters of the function are local variables.
- Any variable on the left-hand side in an assignment in a function is local.
- The variable in a for loop is local.
Global variables

- Variables defined outside a function (the usual kind) are global variables.
- The code in a function can get the value of a global variable.
- The code in a function cannot change the value of a global variable.
- Next time, how to get around this rule….

Metaphor

One-way glass. Function can see program, program can’t see function.

The philosophy

- Calling a function should not have any side effects in the main program.
- The only way a function changes any variables in the main program is perhaps by sending back a return value.
- This is a strategy for containing complexity (within a program) and for sharing code (in a module).

So why have global variables?

- Why should functions in a program be able to see the values of program variables?
- Sometimes it just makes your code a lot simpler.
- Example: snowflake drawing program.

Functions calling Functions

- This happens all the time.
- In a big, complicated program, you have to go from one function to another to follow what is going on.
- Comments describing what every function does are very important.
- Two lines, sometimes more, at the top of a function.
- Give the parameters and return value, and what it does.

Snowflake!

- Same arm six times.
- Each arm is symmetric.
- frac+.03, frac+.03… frac-.03, frac-.03, …
Three functions, all using center
- It’s the same in all of them.
- It doesn’t change for the whole snowflake.
- Let’s just use the global variable.
- Shorter, cleaner program.

New assignment
- Better snowflake picture
- Different centers
- Different snowflakes
- Add your own creations, in a second picture.

Two variables with same name
- Python will let you write a program with a local variable named x and also a global variable named x.

```python
def addTwo(x):
    x = x + 2
    return x
x = 10
z = addTwo(x)
```

A perfectly legal Python program, but a very bad idea!

Usually avoid this
- It can be really confusing.
- You need to know that it can be done because you will end up doing it by mistake.
- When you figure it out, fix it by changing the name of one of the variables.
- OK when you conciously re-use variable names, for example when you have lots of for loops using i as the variable:

```python
for i in range(len(s)):
```

Local or global?
```python
def addTwo(x):
    z = x
    y = y + 2
    print x, y, z
    return y
x = 5
w = addTwo(x)
```

- Is x global or local?
- Does the program crash?
- What does it print?

Local or global?
```python
def addTwo(y):
    x = 3
    y = y + 2
    print x, y
    return y
x = 5
w = addTwo(x)
```

- Is x global or local?
- Does the program crash?
- What does it print?