ECS 10

3/13

Announcements

- Final is Wds Mar 20, 1-3pm, in this room.
- Prog 6 due Sunday Mar 17
- Practice final on SmartSite under resources, a few more programming examples to follow soon.

Functions review

- Style: only line of code outside a function is main()
- Variables defined in each function are local to that function.
- Information is passed explicitly using arguments/ parameters and return values.
- Flow of information is visible.
- Buttons use callback functions to do something in your program.

Problem

- Callback functions might want to have input and output.
- Not obvious how to do that with tkinter.

```
# a button that runs function whatButtonDoes
like = Button(frame,text="Like!",font=big,\
        command=whatButtonDoes)
```

Solution: Global variables

- Shared by all functions in the program
- Here x is declared to be global in setsX()

```
def usesX():
    print(x)

def createsX():
    global x
    x = 5

def main():
    createsX()
    usesX()

main()
```

Global variables

- All functions can see the value of a global variable.
- Only the ones that declare it global can change its value .

```
def usesX():
    print(x)

def createsX():
    global x
    x = 5

def main():
    createsX()
    usesX()

main()
```

Metaphor – one-way glass

Function can see globals, but not change them (unless it declares them).

Style tips

- When using global variables, the flow of information might not be clear.
- Keep global variables to a minimum.

Button callbacks

- Have no parameters or return values, so no way to get data in and out.
- But need to be able to change variables in the program in order to do anything!
- Use global variables.

Tricky bit #1

- A variable defined in a function, not declared to be global, is local.

```
def f():
    x = 3
    # x is local
```

```
def f():
    global x
    x = 3
    # x is global
```

Tricky bit #1

- This is true even if there is a global variable of the same name.
- Watch out for this!

```
def main():
    createX()
    localX()
    seesX()
```

```
def localX():
    x = 2
    print(x)

def seesX():
    print(x)

def createX():
    global x
    x = 5
```

Tricky bit #2

- Variables outside a function are always global. It's easy to forget this and get confused.
- Since we want be very aware of all global variables,
  - don't put code outside functions. (except imports and the call to main()),
  - watch out for assignments with global variables on the left; they have to be declared global in that function.

Style review

- Order of stuff in your program:
1. Imports. Import everything in the first lines.
2. All function definitions, with main() last. Functions get run only when they are called.
3. One statement outside of function definitions: main()
- Declare all global variables explicitly.

Another time globals are handy

- A function needs to remember info from last time it was called.
- Could remember the info in main(), but this defeats the purpose of separating conceptually different parts of the program from each other.