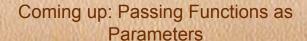
Passing a function as a parameter and How NOT to do menus

Dan Fleck



Passing Functions as Parameters

 You have seen in GUI programming that Buttons (for example) allow you to pass a function to the button to use as a callback. The Button stores this function and calls it later.

 Is there magic here? No... can you do it on your own? YES!

Functions as parameters

 A function parameter can be stored just like a variable:

currentFunction = None

def multiplyFunc(x, y):
 print "MULTIPLY ", x*y

def storeFunction(theFunc):
 global currentFunction
 currentFunction = theFunc

def callCurrent(x, y):
 currentFunction(x, y)

main(): storeFunction(multiplyFunc)

Functions as parameters

 A function parameter can be stored just like a variable:

Functions as parameters

- So, when you construct a Button using this:
- myButton = Button(root, text="Hello", command=sayHello)
- All the Button class does is store your function "sayHello" in a variable to be called later when the button is pressed
- If any of your friends ask, "magic" is an acceptable answer though ©

Text Menus... how not to do it

 When you write a menu, you should use a loop construct, NOT just calling the menu function again:

```
def aFunction():
```

```
-- do something -
myMenu() # Call menu again
```

def myMenu():

```
ans = getSelection()
```

if ans == 'a':

aFunction()

bFunction()

elif ans == 'q':

print "quit" # Exits

BAD!

This is going to put lots of calls on the stack... and will get very odd. For example, bFunction is never executed until the user quits!

Text Menus... how to do it

```
-- do something -
 myMenu() # Call menu again
def myMenu():
 ans = 'x' # Anything but q
 while ans != 'q':
   ans = getSelection()
   if ans == 'a'
        aFunction()
        bFunction()
   elif ans == 'q':
        print "quit" # Exits
```

def aFunction():

GOOD!

Now, afunction is called, then returns and bfunction is called. Then the loop asks the question again, what should I do? Much cleaner and works correctly!