Hello Again

*Ch 9 in Perkovic...*
Today: more about organizing GUI apps

Sec 9.5 in Perkovic
  • has complex example (calculator)

Bonus: more Python programming constructs
A “big idea” in CS

(1) **global vars**
works, but unstructured, not portable

```python
[hello_global.py]
```

Note: Unicode (UTF-8) chars inserted directly, not with `\uNNNN`

why does this work?

(2) **inherit from Frame**
similar to what we did last week with Day, Temp, other examples

```python
[hello_frame.py]
```

note callbacks are instance methods, and command argument to Button constructor is `command = self.lang` (where lang is the name of a method)
(3) Common callback function

Problem with frame version: too many callbacks. They all do the same thing. Why have 5 functions when 1 will do?

Yet Another Aphorism: don’t duplicate code, write a function/method instead

“Abstraction” — what do the two pieces have in common, how do they differ, figure out how to parameterize the differences…

Issue: callback has to have 0 params; tkinter doesn’t let us write something like

```
command = print_hello(lang)
```

Trick (shown in Perkovic’s calculator app in Sec 9.5):
- write a loop that creates buttons, getting labels from an array
- define a function **inside the loop**
- hacky part: give the function a default parameter value that “captures” the argument you want to pass it
- save the function as the callback for the button
- note the function has 1 arg, but the command that calls it passes 0 args

New version of hello app: dictionary mapping lang name to phrase; use keys as button labels; callback just looks up phrase and displays it

```
[hello_shared_cb.py]
```

** major ** advantage: table driven. what do we need to do to add a button?

**Background on why this works:**

def foo ... is basically an assignment statement, adds foo, with value <function..> to current namespace

★ when the function object is created, it “captures” the values of local variables

Can you think of a set of interactive experiments that will test this claim?

- def as assignment statement
- values captured as part of definition

CS terminology for this behavior: “closure”

Back to the hello_shared_cb: note that 5 function objects are created
(4) Replace locally defined function with lambda expression

lambda = “anonymous function”, example:

```python
foo = lambda x: x * x
```

defines a var named `foo`, makes it reference to function object

[hello_lambda.py]

Do some more interactive experiments

• define a function using lambda
• create a list of functions
• watch out for “late binding closures”:
  

• use the “`i = i`” trick…

• not an issue for callbacks because we’re forced to define default args anyway…
A Big Idea in CS

The idea that a function is a piece of data (can be represented by an object in memory, assigned as value of a variable, passed as an argument to another function, ...) is the foundation of computer science.