Computing Science 1P

COMPSCI 1001

Lecture 6: Graphical User Interfaces

February 21st, 2020

Lecturer: Dr Mohamed Khamis
Mohamed.Khamis@Glasgow.ac.uk
https://www.gla.ac.uk/schools/computing/staff/mohamedkhamis/
http://mkhamis.com/

1

Questions from slido

- Coursework recap:
  - Class exam: 15% (you've already done that)
  - Quizzes: 5% (you've already done that)
  - Lab exam 1: 10% (you've already done that)
  - Lab exam 2: 10% in the labs between 23-25 March (week 11)
    - Details about the lab exam will be announced next week.
    - A sample lab exam will be posted to help you with preparation.
  - The final exam is worth 60%
    - https://frontdoor.spa.gla.ac.uk → there are 4 previous exams that you can check for practice.
    - The last lecture will in week 9 (13 March) and will be a revision. If you have questions about previous exams you can post them on slido, and I will dedicate time to answering them in the revision lecture.

2

Questions from slido

- Errors in the except block

You can have nested try/except blocks

```
try:
  1/0
except:
  try:
    1/0
  except:
    pass
```

- Difference between if/else and try/except
- CS1P is not affected by the strikes

Graphical User Interfaces (GUI)

- Key ideas of GUIs developed at Xerox PARC in the 1970s and SRI International in the 1960s
  - On-screen windows, icons, menus, buttons, etc, and a pointing device (mouse)

Xerox Star 8010 workstation

SRI's first computer mouse prototype

Graphical User Interfaces (GUI)

- Key ideas of GUIs developed at Xerox PARC in the 1970s and SRI International in the 1960s
  - On-screen windows, icons, menus, buttons, etc, and a pointing device (mouse)
- Ideas were adopted by Apple
  - First in the Lisa (Local Integrated System Architecture) - 1983
  - Then in the popular Macintosh - 1984
- In 1985, Microsoft introduced Windows
  - First as an application and later as an operating system
- Python makes it very easy to implement simple GUIs
  - We will look at the main points

5

XL

128K

The original Macintosh

6

Terminology

- **Widget**: generic term for a GUI element
  - Button, menu, label, …
- **Widget** (Collins English Dictionary):
  - Informal: any small mechanism or device, the name of which is unknown or temporarily forgotten
  - A small device in a beer can which, when the can is opened, releases nitrogen gas into the beer
  - A small computer program that can be installed on and executed from the desktop of a personal computer

7

The main ideas

- Programs written so far…
  - Almost no user interface, or no input from the user
- Also programs with a simple text-based menu interface
  - Birthday book program: structure of the program is a loop, prompting the user for a command and then calling a function to carry it out
- **GUI programs** similar structure, except…
  - The main loop is provided by a module
  - The functions are called in response to mouse clicks on buttons, etc.
- WIMP (windows, icons, menus, pointer) and post-WIMP HCI

8

Where to find more information

- We will use the tkinter module…
  - Python's de-facto standard GUI package (Linux, Mac, Windows)
  - Not covered in the course textbook. **It will be in the exam !!!**
- Other Python books cover GUI programming with tkinter…
  - But tend to make simple examples more complicated than required
- Useful reference for tkinter8.5 (examples overcomplicated):
  - http://infohost.nmt.edu/tcc/help/pubs/tkinter/tkinter.pdf
- Other links:
  - http://wiki.python.org/moin/TkInter
  - https://docs.python.org/3/library/tkinter.html

The simplest GUI program in Python

example1

```
# Use the tkinter module
import tkinter

# Create the top-level (or root) window
top = tkinter.Tk()

# Create a button ...
quitButton = tkinter.Button(top,text="Quit",
                            command=top.destroy)
# ... and display it in the window
quitButton.grid()

# Start the main loop: responds to the mouse etc
tkinter.mainloop()
```

Line by line

```
# Use the tkinter module
import tkinter
```

Required to use tkinter, widgets, etc…

Line by line

```
# Create the top-level (or root) window
top = tkinter.Tk()
```

Must be present in order to create a window…

You always need at least one window to put buttons etc in

Line by line

```
# Create a button ...
quitButton = tkinter.Button(top,text="Quit",
                            command=top.destroy)
```

must be present, to associate the button with the root window

optional

specifies a callback to be used when the button is pressed

Line by line

```
# ... and display it in the window
quitButton.grid()
```

Uses the layout manager called grid to place the button in the root window

Without this line, the button will not be displayed !!!

Line by line

```
# Start the main loop: responds to the mouse etc
tkinter.mainloop()
```

Starts the main loop, which tracks the mouse and works out when and where it has been pressed

Clicking the mouse on the Quit button causes the callback to be called: i.e., the method **`top.destroy`** is called, which terminates the root window

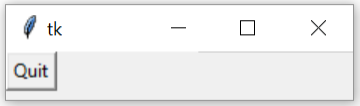
In some books you will see **`top.mainloop()`** instead…

The simplest GUI program in Python

example1

```
# Use the tkinter module
import tkinter

# Create the top-level (or root) window
top = tkinter.Tk()

# Create a button ...
quitButton = tkinter.Button(top,text="Quit",
                            command=top.destroy)
# ... and display it in the window
quitButton.grid()

# Start the main loop: responds to the mouse etc
tkinter.mainloop()
```

The simplest GUI program in Python




```
# Use the tkinter module
import tkinter

# Create the top-level (or root) window
top = tkinter.Tk()

# Create a button ...
quitButton = tkinter.Button(top,text="Quit",
                            command=top.destroy)
# ... and display it in the window
quitButton.grid()

# Start the main loop: responds to the mouse etc
tkinter.mainloop()
```


Event-driven programming

- GUI applications use event-driven programming…
- Events are mouse movements, mouse clicks, key presses, and many higher-level events constructed from these
  - Example: clicking the mouse while the pointer is over a button generates a button press event
  - Modern GUI: Touchscreen
- Some events are handled completely within the main loop provided by tkinter
  - Mouse movements used to update the position of the pointer
  - Clicking the minimise button of the window has the usual effect

Event-driven programming

- Other events that are initiated by the user's input require defining callback functions
  - Usually higher-level (explicit) events such as button presses, menu selections, typing in a text field
  - This is controlled by defining callbacks

- Example: button
  - The event we are interested in is pressing it
  - When the button is created, the command parameter is used to specify which function to call when the button is pressed

```
quitButton = tkinter.Button(top,text="Quit",
                            command=top.destroy)
```


Extending the example

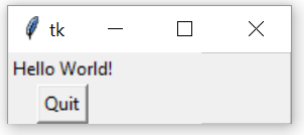
Let's add something to enable us to display a message to the user… tkinter provides **`Label`** for this purpose

example2

```
import tkinter

top = tkinter.Tk()

messageLabel = tkinter.Label(top,text="Hello World!")
messageLabel.grid()

quitButton = tkinter.Button(top,text="Quit",
                            command=top.destroy)
quitButton.grid()

tkinter.mainloop()
```

Extending the example

Let's add something to enable us to display a message to the user… tkinter provides **`Label`** for this purpose




```
import tkinter

top = tkinter.Tk()

messageLabel = tkinter.Label(top,text="Hello World!")
messageLabel.grid()

quitButton = tkinter.Button(top,text="Quit",
                            command=top.destroy)
quitButton.grid()

tkinter.mainloop()
```


Extending the example

Instead of displaying the message immediately, let's add another button with a callback that will display the message

```
import tkinter

def display():
    messageLabel.configure(text="Hello World!")

top = tkinter.Tk()

messageLabel = tkinter.Label(top,text="")
messageLabel.grid()

showButton = tkinter.Button(top,text="Show",command=display)
showButton.grid()

quitButton = tkinter.Button(top,text="Quit",command=top.destroy)
quitButton.grid()

tkinter.mainloop()
```

What do we do here?
create 'placeholder' and then configure

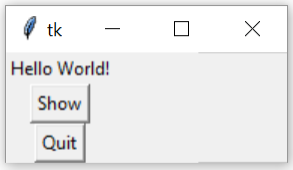
example3

Extending the example

Instead of displaying the message immediately, let's add another button with a callback that will display the message

```
import tkinter

def display():
    messageLabel.configure(text="Hello Wor

top = tkinter.Tk()

messageLabel = tkinter.Label(top,text="")
messageLabel.grid()

showButton = tkinter.Button(top,text="Show",command=display)
showButton.grid()

quitButton = tkinter.Button(top,text="Quit",command=top.destroy)
quitButton.grid()

tkinter.mainloop()
```

What do we create 'plac



example3

Extending the example

Can we do this instead?

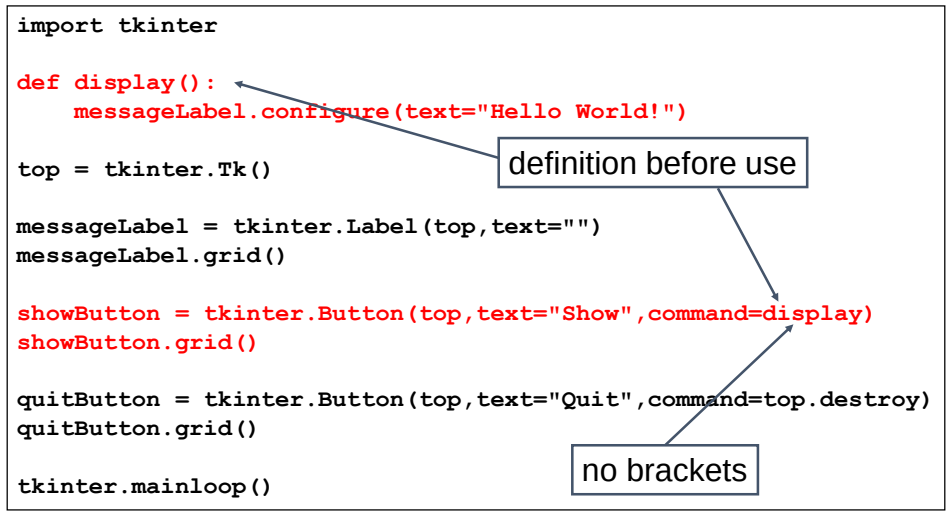
```
import tkinter

def display():
    messageLabel = tkinter.Label(top,text="Hello World!")
    messageLabel.grid()

top = tkinter.Tk()

showButton = tkinter.Button(top,text="Show",command=display)
showButton.grid()

quitButton = tkinter.Button(top,text="Quit",command=top.destroy)
quitButton.grid()

tkinter.mainloop()
```

How is this different than the last slide?

example3_2

Points to note

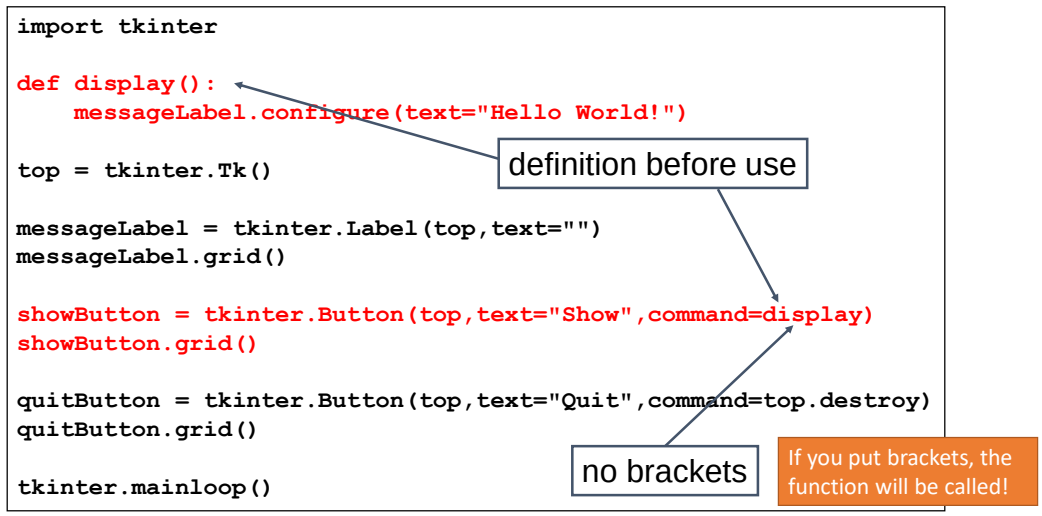
```
import tkinter

def display():
    messageLabel.configure(text="Hello World!")

top = tkinter.Tk()

messageLabel = tkinter.Label(top,text="")
messageLabel.grid()

showButton = tkinter.Button(top,text="Show",command=display)
showButton.grid()

quitButton = tkinter.Button(top,text="Quit",command=top.destroy)
quitButton.grid()

tkinter.mainloop()
```



25

Points to note

```
import tkinter

def display():
    messageLabel.configure(text="Hello World!")

top = tkinter.Tk()

messageLabel = tkinter.Label(top,text="")
messageLabel.grid()

showButton = tkinter.Button(top,text="Show",command=display)
showButton.grid()

quitButton = tkinter.Button(top,text="Quit",command=top.destroy)
quitButton.grid()

tkinter.mainloop()
```



26

Changing the layout

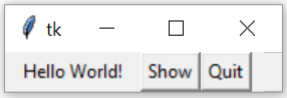
We can use optional arguments with the **grid** method to control how widgets are placed…

example4

```
import tkinter

def display():
    messageLabel.configure(text="Hello World!")

top = tkinter.Tk()

messageLabel = tkinter.Label(top,text="",width=12)
messageLabel.grid(row=0,column=0)

showButton = tkinter.Button(top,text="Show",command=display)
showButton.grid(row=0,column=1)

quitButton = tkinter.Button(top,text="Quit",command=top.destroy)
quitButton.grid(row=0,column=2)

tkinter.mainloop()
```


Changing the layout

We can use optional arguments with the **grid** method to control how widgets are placed…



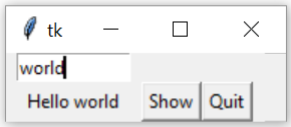

```
import tkinter

def display():
    messageLabel.configure(text="Hello World!")

top = tkinter.Tk()

messageLabel = tkinter.Label(top,text="",width=12)
messageLabel.grid(row=0,column=0)

showButton = tkinter.Button(top,text="Show",command=display)
showButton.grid(row=0,column=1)

quitButton = tkinter.Button(top,text="Quit",command=top.destroy)
quitButton.grid(row=0,column=2)

tkinter.mainloop()
```

Getting input from the user

example5

```
import tkinter

def display():
    name = textVar.get()
    messageLabel.configure(text="Hello "+name)

top = tkinter.Tk()

textVar = tkinter.StringVar("")
textEntry = tkinter.Entry(top,textvariable=textVar,width=12)
textEntry.grid(row=0,column=0)

messageLabel = tkinter.Label(top,text="",width=12)
messageLabel.grid(row=1,column=0)

showButton = tkinter.Button(top,text="Show",command=display)
showButton.grid(row=1,column=1)

quitButton = tkinter.Button(top,text="Quit",command=top.destroy)
quitButton.grid(row=1,column=2)

tkinter.mainloop()
```


Getting input from the user

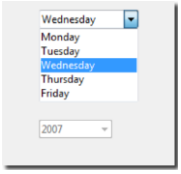



```
import tkinter

def display():
    name = textVar.get()
    messageLabel.configure(text="Hello "+name)

top = tkinter.Tk()

textVar = tkinter.StringVar("")
textEntry = tkinter.Entry(top,textvariable=textVar,width=12)
textEntry.grid(row=0,column=0)

messageLabel = tkinter.Label(top,text="",width=12)
messageLabel.grid(row=1,column=0)

showButton = tkinter.Button(top,text="Show",command=display)
showButton.grid(row=1,column=1)

quitButton = tkinter.Button(top,text="Quit",command=top.destroy)
quitButton.grid(row=1,column=2)

tkinter.mainloop()
```

Important idea

- The **`Entry`** widget allows the user to enter text, but how do we store or manipulate it?
  - We have created a **`tkinter.StringVar`** object and gave it to the **`Entry`** object
  - Then use the **`get`** method of the **`StringVar`** to obtain the text

- This style of programming is also needed with several other tkinter widgets
  - It must be a **`StringVar`**, not an ordinary string variable
  - Example: Combobox

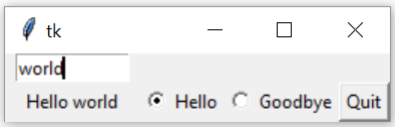


Example GUIs so far

Widgets: button, label, entry…

Another example: Radiobutton

example6




```
def display():
    name = textVar.get()
    ch = choice.get()
    if ch == 1:
        message = "Hello "+name
    elif ch == 2:
        message = "Goodbye "+name
    else:
        message = ""
    messageLabel.configure(text=message)

top = tkinter.Tk()
textVar = tkinter.StringVar("")
textEntry = tkinter.Entry(top,textvariable=textVar,width=12)
textEntry.grid(row=0,column=0)
messageLabel = tkinter.Label(top,text="",width=12)
messageLabel.grid(row=1,column=0)
choice = tkinter.IntVar(0)
helloButton = tkinter.Radiobutton(top,text="Hello",
                                  variable=choice,value=1,command=display)
helloButton.grid(row=1,column=1)
goodbyeButton = tkinter.Radiobutton(top,text="Goodbye",
                                    variable=choice,value=2,command=display)
goodbyeButton.grid(row=1,column=2)
quitButton = tkinter.Button(top,text="Quit",command=top.destroy)
quitButton.grid(row=1,column=3)
tkinter.mainloop()
```


Another example: Radiobutton

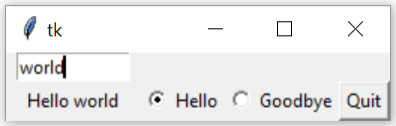
example6

```
def display():
    name = textVar.get()
    ch = choice.get()
    if ch == 1:
        message = "Hello "+name
    elif ch == 2:
        message = "Goodbye "+name
    else:
        message = ""
    messageLabel.configure(text=message)

top = tkinter.Tk()
textVar = tkinter.StringVar("")
textEntry = tkinter.Entry(top,textvariable=textVar,width=12)
textEntry.grid(row=0,column=0)
messageLabel = tkinter.Label(top,text="",width=12)
messageLabel.grid(row=1,column=0)
choice = tkinter.IntVar(0)
helloButton = tkinter.Radiobutton(top,text="Hello",
                                  variable=choice,value=1,command=display)
helloButton.grid(row=1,column=1)
goodbyeButton = tkinter.Radiobutton(top,text="Goodbye",
                                    variable=choice,value=2,command=display)
goodbyeButton.grid(row=1,column=2)
quitButton = tkinter.Button(top,text="Quit",command=top.destroy)
quitButton.grid(row=1,column=3)
tkinter.mainloop()
```

Another example: Radiobutton

example6

```
def display():
    name = textVar.get()
    ch = choice.get()
    if ch == 1:
        message = "Hello "+name
    elif ch == 2:
        message = "Goodbye "+name
    else:
        message = ""
    messageLabel.configure(text=message)

top = tkinter.Tk()
textVar = tkinter.StringVar("")
textEntry = tkinter.Entry(top,textvariable=textVar,width=12)
textEntry.grid(row=0,column=0)
messageLabel = tkinter.Label(top,text="",width=12)
messageLabel.grid(row=1,column=0)
choice = tkinter.IntVar(0)
helloButton = tkinter.Radiobutton(top,text="Hello",
                                  variable=choice,value=1,command=display)
helloButton.grid(row=1,column=1)
goodbyeButton = tkinter.Radiobutton(top,text="Goodbye",
                                    variable=choice,value=2,command=display)
goodbyeButton.grid(row=1,column=2)
quitButton = tkinter.Button(top,text="Quit",command=top.destroy)
quitButton.grid(row=1,column=3)
tkinter.mainloop()
```



Summary

- GUIs
- Tkinter in Python
- Widgets:
  - Buttons
  - Labels
  - Entry
  - RadioButton
- Important functions:
  - tkinter.Tk()
  - tkinter.StringVar(“”)
  - tkinter.IntVar(0)
  - widget.grid()
  - tkinter.mainloop()

Ask questions on Slido! #CS1P