Teaching London Computing

A Level Computer Science

Programming GUI in Python

William Marsh
School of Electronic Engineering and Computer Science
Queen Mary University of London

Outline

- A first program
- Concepts in **G**raphical **U**ser **I**nterface
  - Components / widgets and attributes
  - Events / actions
  - Layout
- *Practical examples*
- Challenges of GUI programming
  - Choosing a GUI library
  - Using Object-Oriented programming

First Program – Click the Button

- Code provided but not yet explained
- Use ‘pattern matching’ (i.e. intelligent guessing) to modify it

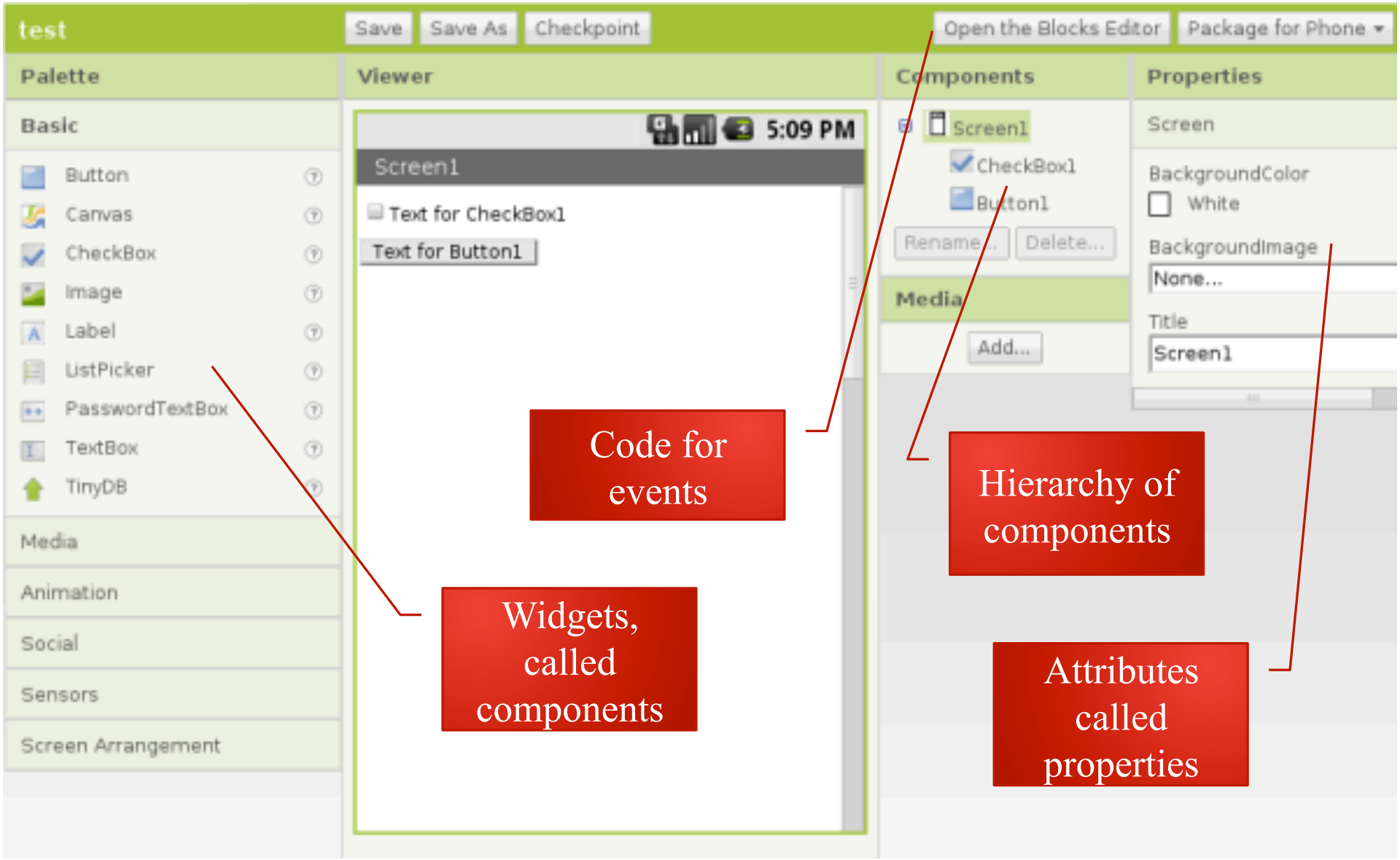
Key Concepts

Explained Using the Button Example

Key Concepts

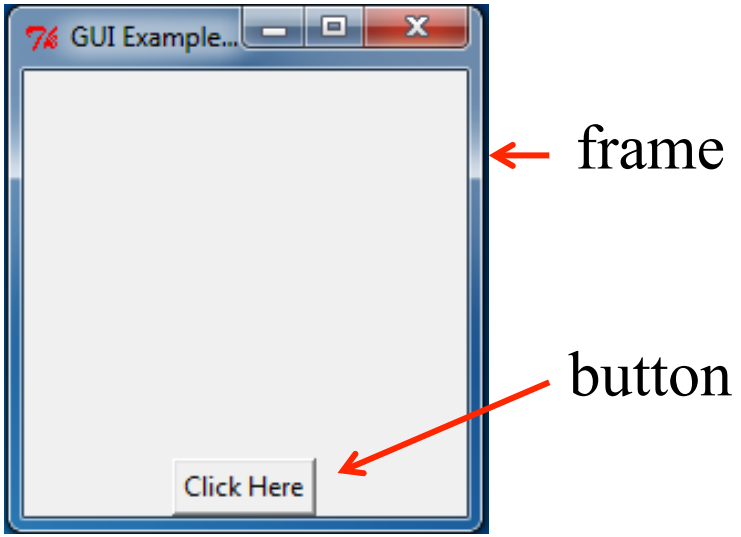
- A widget / component
  - E.g. a button, a frame
  - Attributes e.g. the button text
- Actions
  - E.g. what happens when you press the button
- Layout
  - Positioning widgets

AppInventor



Widgets

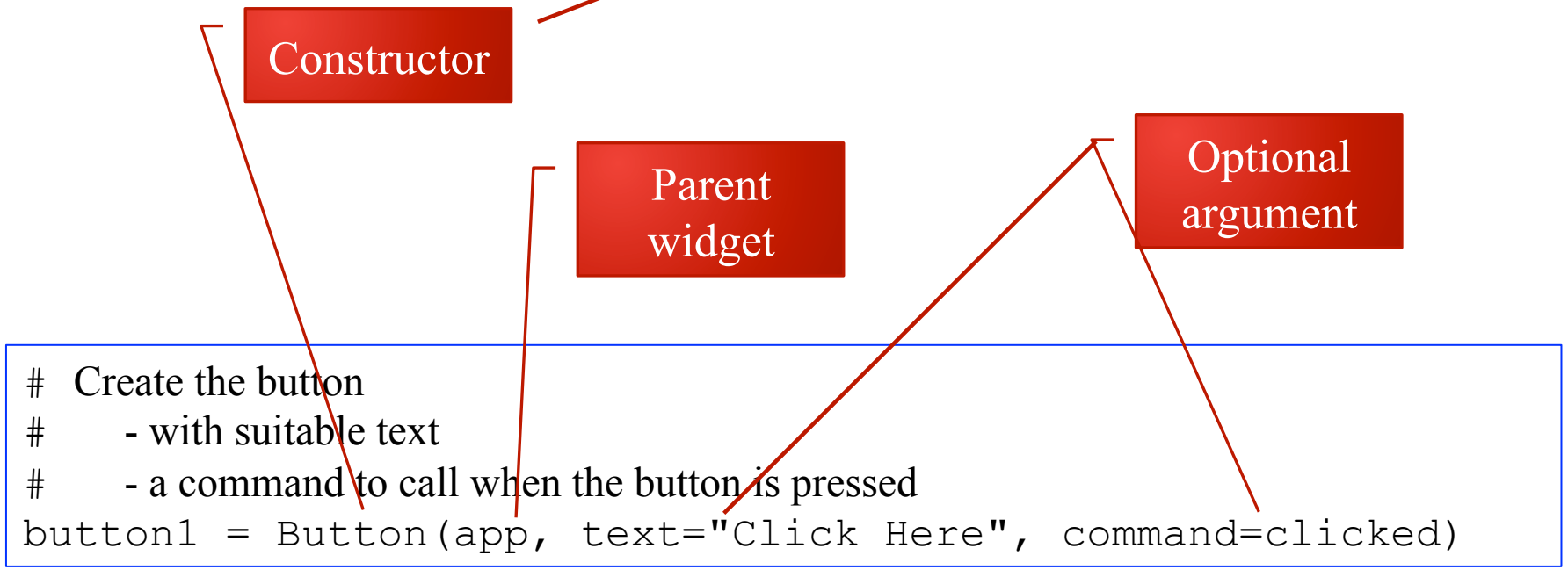
- A GUI is made up from widgets
- A widget is created
- Widget has attributes
- One widget may contain another:
  - Frame contains the button



Create a Widget

- Constructor
  - Name same as widget
  - Hierarchy of widget
  - Optional arguments

```
#  Create a main frame with
#        - a title
#        - size 200 by 200 pixels
app = Tk()
app.title("GUI Example 1")
app.geometry('200x200')
```



Widgets have Attributes

- E.g. a name, size
- Any property of the widget that makes it specific

```
#  Create a main frame with
#        - a title
#        - size 200 by 200 pixels
app = Tk()
app.title("GUI Example 1")
app.geometry('200x200')
```

Attributes set by constructor (note use of keyword arguments)

Methods to set attributes

```
#  Create the button
#       - with suitable text
#       - a command to call when the button is pressed
button1 = Button(app, text="Click Here", command=clicked)
```

How to Set / Get an Attribute

- Method 1 (setting):
  - Set value with the constructor
- Method 2 (setting and getting):
  - Widget is a dictionary

```
#  Change button text
mText = button1['text']
button1['text'] = mText.upper()
```

- Method 3 (sometimes)
  - Call a suitable method

Other methods exist

Aside: Dictionaries

- Dictionary: a map from a key to a value
  - Unique key
  - Built in (Python) versus library (many other languages)

Standard Array	Python Dictionary
Index by number	Key can be a string, pair, …
Indices continuous e.g. 0 → 10	Gaps ok
Holds only number, character	Any value – even a dictionary

```
#  Change button text
mText = button1['text']
button1['text'] = mText.upper()
```

Lookup

Update

Handle an Event

```
#  This method is called when the button is pressed
def clicked():
    print("Clicked")

#  Create the button with
#     - a command to call when the button is pressed
button1 = Button(app, text="Click Here", command=clicked)
```

- Events
  - Button, mouse click, key press
- Action
  - Event ‘bound’ to function

Name of a Method

Layout the Widget

```
#  Make the button visible at the bottom of the frame
button1.pack(side='bottom')
```

- Where does the widget go?
  - Hierarchy
  - Top-level window
- Layout manager
  - Several available
  - Problem of resizing
- The ‘pack’ layout manager is simplest
- ***Widget is not visible until packed***

A Minimal Application

```
#  Import the Tkinter package
#      Note in Python 3 it is all lowercase
from tkinter import *

#  Create a main frame
app = Tk()

#  Start the application running
app.mainloop()
```

Loop to handle events

```
#  Import the Tkinter package
#      Note in Python 3 it is all lowercase
import tkinter as tk

#  Create a main frame
app = tk.Tk()

#  Start the application running
app.mainloop()
```

import with prefix

(Some) tkinter Widgets

Widget	Use
Button	A button
Canvas	For drawing graphics
Entry	Entry a line of text
Frame	A rectangular area containing other widgets
Label	Display a single line of text
Menu	A set of options shown when on a menu bar
Radiobutton	Select one of a number of choices
Scrollbar	Horizontal or vertical scrolling of a window
Text	A multi-line text entry
Toplevel	*A top-level frame*

Further Practical Exercises

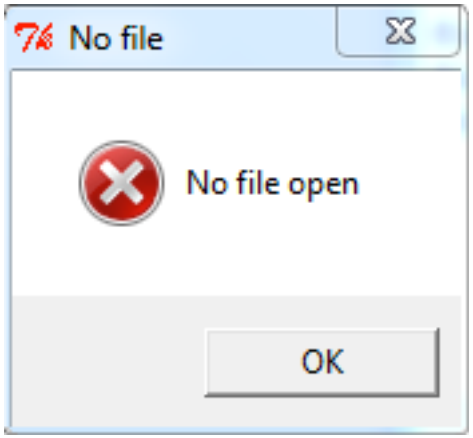
- **See practical sheet**
- A sequence of exercises introduce other widgets and apply the core concepts
- Answers included
- … probably too many to finish now

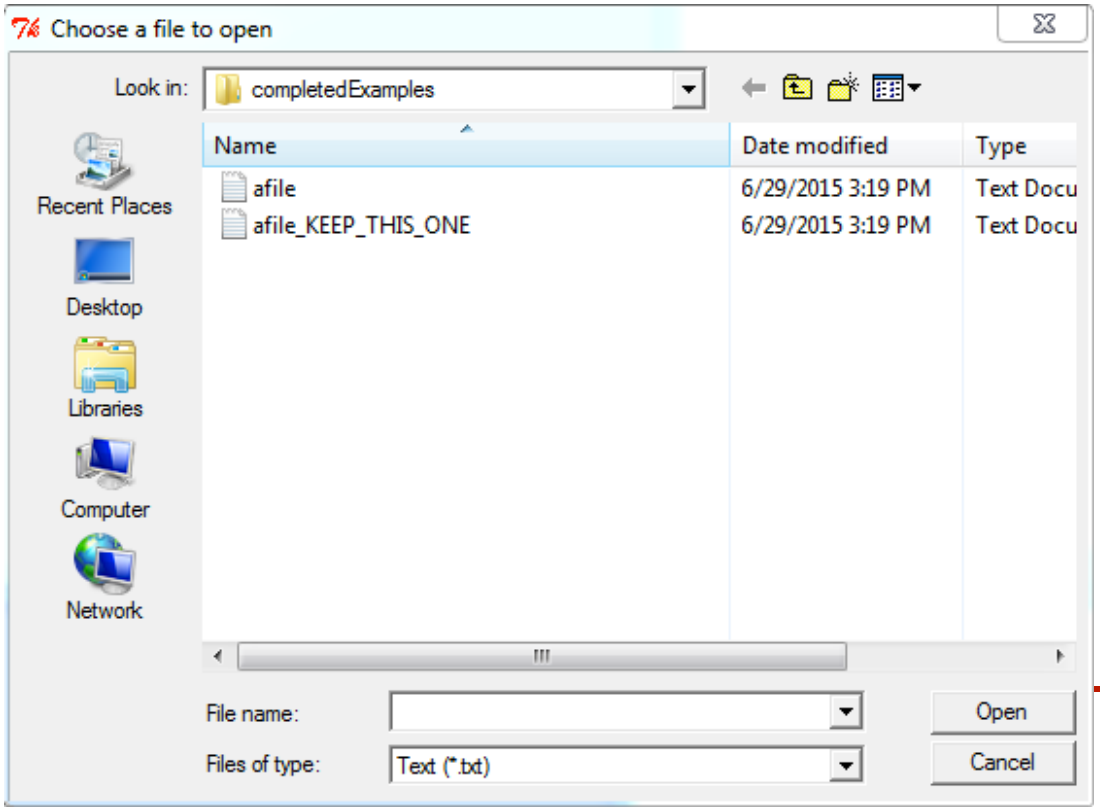
Further Concepts

- Dialog
- Top-level window
- Control variables

Dialogs

- You must respond to a dialog
  - Messages
  - File choosing





Top-Level Windows

- At least one top-level window
  - Conveniently created using `Tk()`
  - Like a frame but …
- Menu bar
- Standard buttons
- Borders

Control Variables

- Variables linking …
  - Entry widget to its text
  - Choices in a RadioButton
- These are objects in the framework

Challenges in GUI

- *Which framework?*
- *How to design a GUI*
- *How much OOP?*

GUI Framework

- A GUI framework defines a set of widgets
  - Windows has it's own GUI framework
- Python uses a portable GUI framework
  - tkinter, depends on Tk and TCL
  - PyQT, depends on QT
- Tkinter
  - Pro: simple, easy to install
  - Cons: a bit limited; documentation weak
- PyQT: more complex

Designing a GUI

- What am I trying to do?
- What widgets do I need?
  - Where will they go?
  - How do they behave?

The OOP Problem

- Why OO and GUI
  - Widgets are classes
  - Default behaviour
- GUI programs are often organised using classes

```
#!/usr/bin/env python                                    1
import Tkinter as tk                                     2

class Application(tk.Frame):                             3
    def __init__(self, master=None):
        tk.Frame.__init__(self, master)                  4
        self.grid()                                      5
        self.createWidgets()

    def createWidgets(self):
        self.quitButton = tk.Button(self, text='Quit',
            command=self.quit)                           6
        self.quitButton.grid()                           7

app = Application()                                      8
app.master.title('Sample application')                   9
app.mainloop()                                           10
```

- Practical Problem: most examples use OOP

Summary

- Core concepts common to all framework
- **Understand** principles
- **Learn about** available widgets
- **Look up** attributes and methods

- After programming … interface design