

Graphical user interfaces (GUI)

- Tkinter

primitive_calculator.py

```
accumulator = 0

while True:
    print("Accumulator:", accumulator)
    print("Select:")
    print("  1: clear")
    print("  2: add")
    print("  3: subtract")
    print("  4: multiply")
    print("  5: quit")

    choice = int(input("Choice: "))

    if choice == 1: accumulator = 0
    if choice == 2: accumulator += int(input("add: "))
    if choice == 3: accumulator -= int(input("subtract: "))
    if choice == 4: accumulator *= int(input("multiply by: "))
    if choice == 5: break
```

Python shell

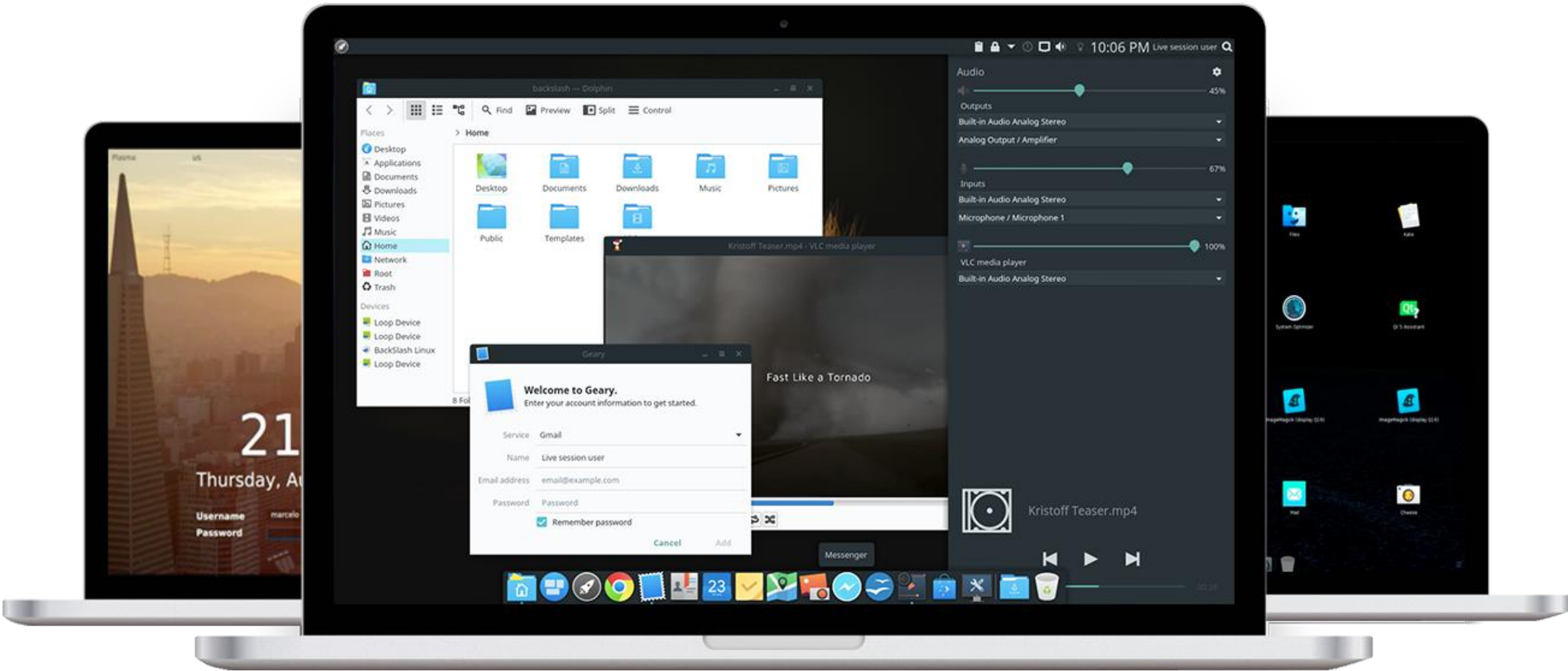
```
Accumulator: 0
Select:
  1: clear
  2: add
  3: subtract
  4: multiply
  5: quit
Choice: 2
add: 10
Accumulator: 10
Select:
  1: clear
  2: add
  3: subtract
  4: multiply
  5: quit
Choice: 2
add: 15
Accumulator: 25
Select:
...
```

Python GUI's (Graphical Users Interfaces)

- There is a long list of GUI frameworks and toolkits, designer tools
 - we will only briefly look at Tkinter
- GUI are, opposed to a text terminal, **easier to use, more intuitive** and **flexible**
- Windows, icons, menus, buttons, scrollbards mouse / touch / keyboard interaction etc.
- Operating system (e.g. Windows, macOS, iOS, Linux, Android) provides basic functionality in particular a **windows manager**
- Writing GUI applications from scratch can be painful – frameworks try to provide all standard functionality



en.wikipedia.org/wiki/Colossal_Cave_Adventure



BackSlash Linux GUI
www.backslashlinux.com

Tkinter

- “Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of Tcl/Tk.”
- “Tcl is a high-level, general-purpose, interpreted, dynamic programming language.”
- “Tk is a free and open-source, cross-platform widget toolkit that provides a library of basic elements of GUI widgets for building a graphical user interface (GUI) in many programming languages.”
- “The popular combination of Tcl with the Tk extension is referred to as Tcl/Tk, and enables building a graphical user interface (GUI) natively in Tcl. Tcl/Tk is included in the standard Python installation in the form of Tkinter.”

Terminology

- **widgets** (e.g. buttons, editable text fields, labels, scrollbars, menus, radio buttons, check buttons, canvas for drawing, frames...)
- **events** (e.g. mouse click, mouse entering/leaving, resizing windows, redraw requests, ...)
- **listening** (application waits for events to fired)
- **event handler** (a function whose purpose is to handle an event, many triggered by OS/Window manager)
- **geometry managers** (how to organize widgets in a window: Tkinter *pack, grid, place*)



ipsa18

Aarhus Universitet

www.au.dk

IPSA18 PeerWise - Login GTD

ipsa18.pptx - PowerPoint

File Home Insert Design Transitions Animations Slide Show Review View Developer ACROBAT Storyboarding Tell me what you want to do... Sign in Share

Paste New Slide

Layout Reset Section

Clipboard Slides

Font

Paragraph

Drawing

Editing

1 Graphical user interfaces (GUI)

2

3 Python GUI's (Graphical Users Interfaces)

4

Graphical user interfaces (GUI)

Tkinter

Click to add notes

Slide 1 of 15 English (United States) 42 %

docs.python.org/3/library/tk.html



“tkinter is also famous for having an outdated look and feel”

Welcome example



welcome.py

```
import tkinter

root = tkinter.Tk() # root window

def do_quit(): # event handler for "Close" button
    root.destroy()

root.title("Tkinter Welcome GUI")

label = tkinter.Label(root, text="Welcome to Tkinter", background="yellow")
label.pack(side=tkinter.LEFT)

close_button = tkinter.Button(root, text="Close", command=do_quit)
close_button.pack(side=tkinter.RIGHT)

tkinter.mainloop() # loop until all windows are closed/destroyed
```

Welcome example (class)

welcome_class.py

```
import tkinter

class Welcome:
    def do_quit(self): # event handler for "Close"
        self.root.destroy()

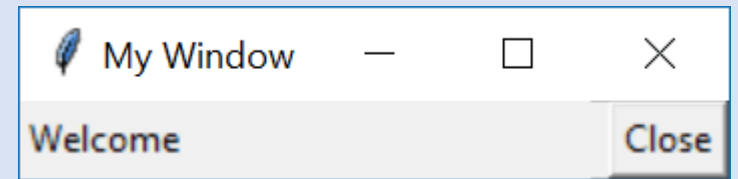
    def __init__(self, window_title):
        self.root = tkinter.Tk()
        self.root.title(window_title)

        self.label = tkinter.Label(self.root, text="Welcome")
        self.label.pack(side=tkinter.LEFT)

        self.close_button = tkinter.Button(self.root, text="Close", command=self.do_quit)
        self.close_button.pack(side=tkinter.RIGHT)

Welcome("My Window")

tkinter.mainloop()
```



increment.py (part I)

```
import tkinter

class Counter:
    def do_quit(self):
        self.root.destroy()

    def add(self, x):
        self.counter += x
        self.count.set(self.counter)

    def __init__(self, message):
        self.counter = 0

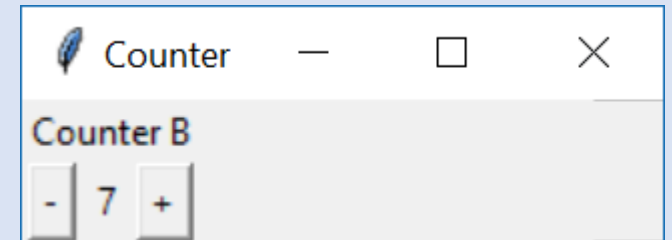
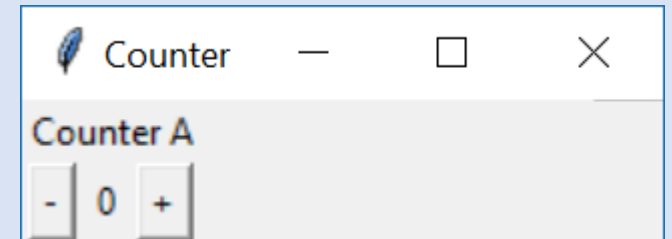
        self.root = tkinter.Toplevel() # new window
        self.root.title("Counter")

        self.label = tkinter.Label(self.root, text=message)
        self.label.grid(row=0, columnspan=3)

        self.minus_button = tkinter.Button(self.root, text="-", command=lambda: self.add(-1))
        self.minus_button.grid(row=1, column=0)

        self.count = tkinter.IntVar()
        self.count_label = tkinter.Label(self.root, textvariable=self.count)
        self.count_label.grid(row=1, column=1)

        self.plus_button = tkinter.Button(self.root, text="+", command=lambda: self.add(+1))
        self.plus_button.grid(row=1, column=2)
```



increment.py (part II)

```
class Counter_app:
    def new_counter(self):
        Counter("Counter " + chr(ord('A') + self.counters))
        self.counters += 1

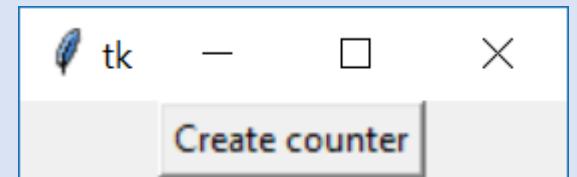
    def __init__(self):
        self.counters = 0

        self.root = tkinter.Tk()

        self.create = tkinter.Button(self.root, text="Create counter", command=self.new_counter)
        self.create.pack()

Counter_app()

tkinter.mainloop()
```



Canvas

`canvas.py`

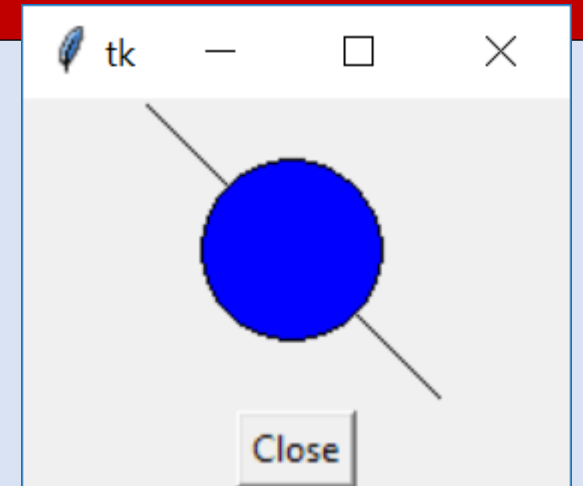
```
import tkinter

root = tkinter.Tk()

canvas = tkinter.Canvas(root, width=100, height=100)
canvas.pack()
canvas.create_line(0, 0, 100, 100)
canvas.create_oval(20, 20, 80, 80, fill="blue")

close = tkinter.Button(root, text="Close", command=root.destroy)
close.pack()

tkinter.mainloop()
```





Calculator



42

7

8

9

*

C

4

5

6

/

%

1

2

3

-

=

0

.

+

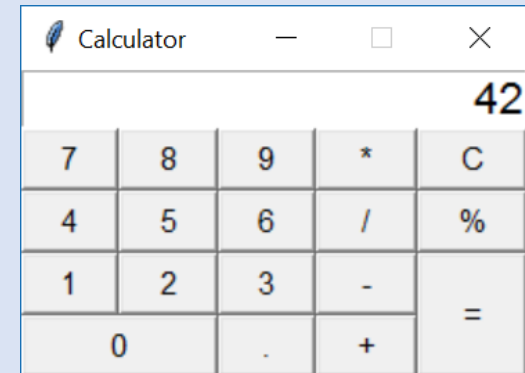
calculator.py (Part I)

```
import tkinter
from tkinter import messagebox

class Calculator:
    def __init__(self, root):
        self.root = root

        self.display = tkinter.Entry(self.root, font=("Helvetica", 16), justify=tkinter.RIGHT)
        self.display.insert(0, "0")
        self.display.grid(row=0, column=0, columnspan=5)

        self.button(1, 0, '7')
        self.button(1, 1, '8')
        self.button(1, 2, '9')
        self.button(1, 3, '*')
        self.button(1, 4, 'C', command=self.clearText) # 'C' button
        self.button(2, 0, '4')
        self.button(2, 1, '5')
        self.button(2, 2, '6')
        self.button(2, 3, '/')
        self.button(2, 4, '%')
        self.button(3, 0, '1')
        self.button(3, 1, '2')
        self.button(3, 2, '3')
        self.button(3, 3, '-')
        self.button(3, 4, '=', rowspan=2, command=calculateExpression) # '=' button
        self.button(4, 0, '0', columnspan=2)
        self.button(4, 2, '.')
        self.button(4, 3, '+')
```



calculator.py (Part II)

```
def button(self, row, column, text, command=None, columnspan=1, rowspan=1):
    if command == None:
        command = lambda: self.appendToDisplay(text)
    B = tkinter.Button(self.root, font=("Helvetica", 11), text=text, command=command)
    B.grid(row=row, column=column, rowspan=rowspan, columnspan=columnspan, sticky="NWNESWSE")

def clearText(self):
    self.replaceText("0")

def replaceText(self, text):
    self.display.delete(0, tkinter.END)
    self.display.insert(0, text)

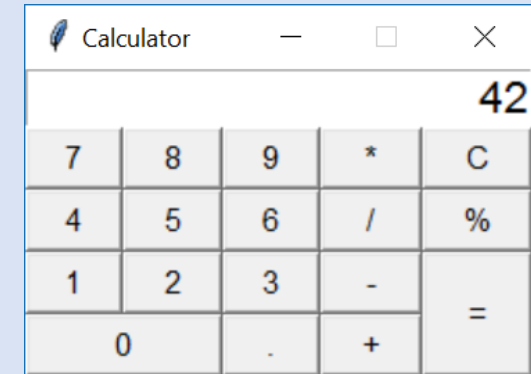
def appendToDisplay(self, text):
    if self.display.get() == "0":
        self.replaceText(text)
    else:
        self.display.insert(tkinter.END, text)

def calculateExpression(self):
    expression = self.display.get().replace("%", "/ 100")
    try:
        result = eval(expression) # DON'T DO THIS !!! ⚠
        self.replaceText(result)
    except:
        messagebox.showinfo("Message", "Invalid expression", icon="warning")

root = tkinter.Tk()
root.title("Calculator")
root.resizable(0, 0)

Calculator(root)

tkinter.mainloop()
```



Creating a menu

rectangles.py

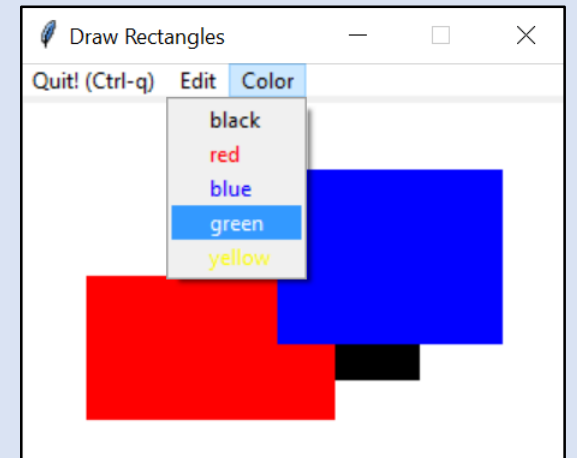
```
...
menubar = tkinter.Menu(self.root)
menubar.add_command(label="Quit! (Ctrl-q)", command=self.do_quit)

editmenu = tkinter.Menu(menubar, tearoff=0)
editmenu.add_command(label="Clear", command=self.clear_all)
editmenu.add_command(label="Delete last (Ctrl-z)", command=self.delete_last_rectangle)

def get_color_handler(self, color):
    return lambda : self.current_color = color

colormenu = tkinter.Menu(menubar, tearoff=0)
for color in self.Colors: # list of color names
    colormenu.add_command(label=color,
                          foreground=color,
                          command=self.get_color_handler(color))

menubar.add_cascade(label="Edit", menu=editmenu)
menubar.add_cascade(label="Color", menu=colormenu)
self.root.config(menu=menubar) # Show menubar
...
```

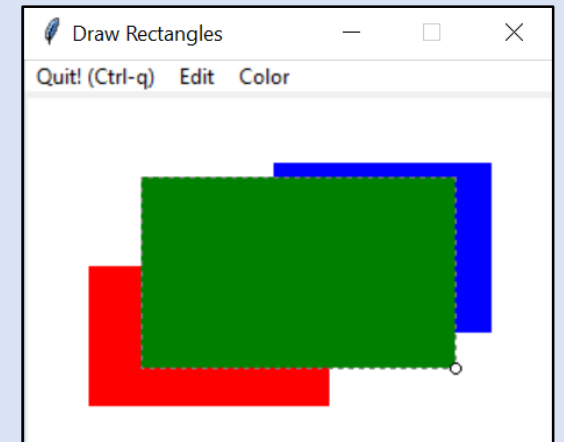


Binding key and mouse events

- Whenever a key is pressed, mouse button is pressed/released, mouse is moved, mouse enters/leaves objects etc. **events** are triggered that can be bound to call a user defined **event handler**

rectangles.py

```
...
self.root = tkinter.Tk()
self.root.bind('<Control-q>', self.do_quit)
self.root.bind('<Control-z>', self.delete_last_rectangle)
...
self.canvas = tkinter.Canvas(self.root, width=300, height=200,
                             background='white')
self.canvas.bind('<Button-1>', self.create_rectangle_start)
self.canvas.bind('<B1-Motion>', self.create_rectangle_mouse_move)
self.canvas.bind('<ButtonRelease-1>', self.create_rectangle_end)
...
```



Exercise 25.1 (convex hull GUI)

