GUIs: Menu boxes, Dialogs, and Menus

Introduction to Software Engineering

Lecture 11A

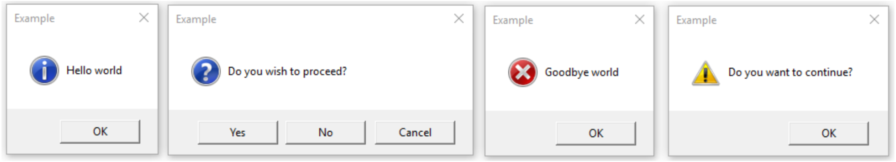
[illegible]

May 9, 2023

Menu Boxes

Message Box

The *sub-library* `tkinter.messagebox` provides a variety of convenience methods for raising dialogs like below.

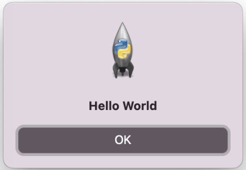


From left to right, we have:

1. Information box,
2. Question box.
3. Warning box,
4. Error box.

tkinter.messagebox.showinfo

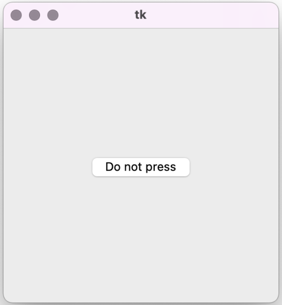
```
import tkinter as tk
from tkinter import messagebox
                              Importing a submodule

messagebox.showinfo(
    title="Title",
    message="Hello World"
)
  Title is OS specific;  OSX doesn't display it.
```

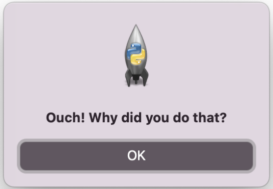

Hello World

```
>>> mbox = messagebox.showinfo(
...     title="Title",
...     message="Hello World"
... )                                         Dialog window will display

>>> mbox
'ok'                                  The message box returns this string
```

Exercise

Create a GUI with a *single button* that, when pressed, raises a message box.



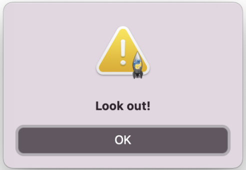


Ouch! Why did you do that?

Answer: `Ex01.py`

tkinter.messagebox.showwarning

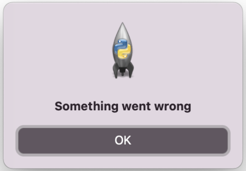
```python
import tkinter as tk
from tkinter import messagebox

messagebox.showwarning(
    title=None,
    message="Look out!"
)
```


Look out!

Returns ok.

tkinter.messagebox.showerror

```
import tkinter as tk
from tkinter import messagebox

messagebox.showerror(
    title=None,
    message="Something went wrong"
)
```

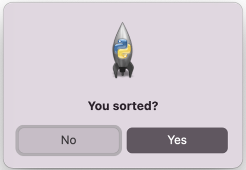


Something went wrong

Returns the string `ok`.

tkinter.messagebox.askquestion

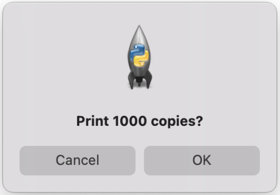
```python
import tkinter as tk
from tkinter import messagebox

messagebox.askquestion(
     title=None,
     message="You sorted?"
)
```



Returns the string `yes` or `no`.

tkinter.messagebox.askokcancel

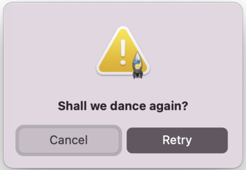
```
import tkinter as tk
from tkinter import messagebox

messagebox.askokcancel(
    title=None,
    message="Print 1000 copies?"
)
```


Print 1000 copies?

Returns boolean True or False.

tkinter.messagebox.askretrycancel

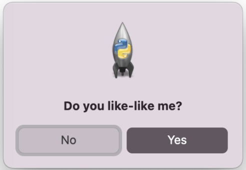
```
import tkinter as tk
from tkinter import messagebox

messagebox.askretrycancel(
    title=None,
    message="Shall we dance again?"
)
```


Shall we dance again?

Returns boolean `True` or `False`.

tkinter.messagebox.askyesno

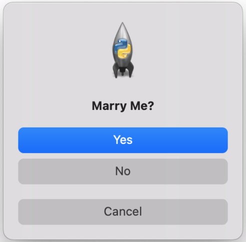
```python
import tkinter as tk
from tkinter import messagebox

messagebox.askyesno(
    title=None,
    message="Do you like-like me?"
)
```


Do you like-like me?

Returns boolean `True` or `False`.

tkinter.messagebox.askyesnocancel

```
import tkinter as tk
from tkinter import messagebox

messagebox.askyesnocancel(
    title=None,
    message="Marry me?"
)
```



Returns `None` or boolean `True` or `False` .

`destroy`

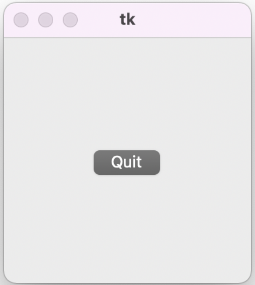
One common use of these message dialogs is to *ask the user if they do indeed want to quit the program* (i.e. terminate the GUI).

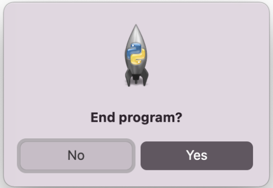
To terminate a GUI we do

```
window.destroy()
```

Exercise

Write a GUI that has a *single button* that triggers a dialog window asking if the user wants to quit. If `yes`, destroy the GUI.




End program?

Answer: `Ex02.py`

Dialogs

Dialog

A *dialog* is a *conversation between (at least) two entities*. In our context the *entities* are the *GUI* and the *user*.

Typically we need to enter a *dialog* with the user when *opening* or *saving* files.

Consider the following code block that opens a file for reading.

```
with open("path/to/some_file.txt", ‘r’) as the_file:
    for line in the_file:
        print(the_file)
```

Rather than hard-code the file, we can use a *dialog* instead.

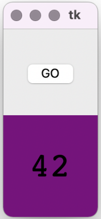
```
from tkinter import filedialog

with filedialog.askopenfile(mode=‘r’) as the_file:
    for line in the_file:
        print(line)
```

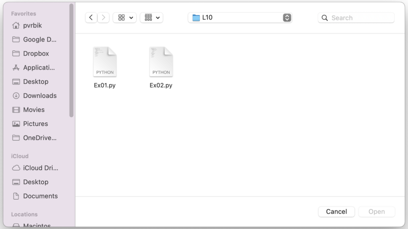
The above will get the operating system to open a *file-picker*.

Exercise

Write a GUI with a single button that, when pressed, prompts the user to open a file.

Then display the *line count* of the file in a label.





Answer: `Ex03.py`

Exercise

Write a version of the GUI from the previous exercise that *does not crash* when the user presses cancel.

File Dialogs

`askopenfile(mode="r", **options)`	Create an *open* dialog and return the *file pointer*.
`askopenfilename(**options)`	Create an *open* dialog and return the *file name*.
`asksaveasfile(mode="w", **options)`	Create a *save-as* dialog and return the *file pointer*.
`asksaveasfilename(**options)`	Create an *save-as* dialog and return the *file name*.
`askdirectory(**options)`	Create an *open* dialog and return the *directory path*.

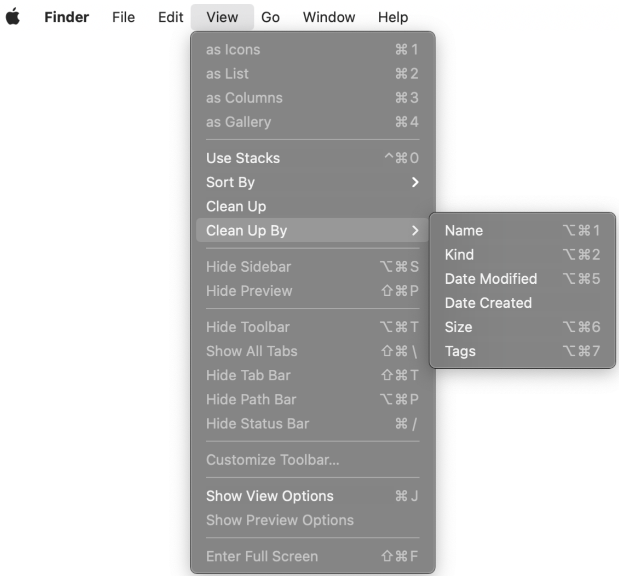
Menus

Menus

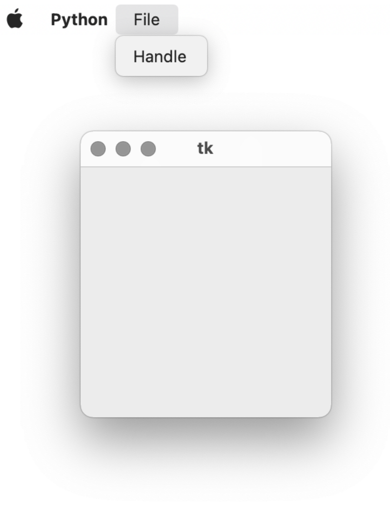
Actions like *opening and closing files* or *changing settings* are usually done through *menus*.

Menus are subcategorized into columns or *submenus* comprised of options (which are essentially just buttons).

These submenus can *cascade* in the sense that a menu item can be made to show more menu items.

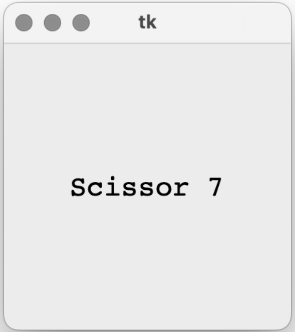

Finder
File
Edit
View
Go
Window
Help
as Icons ⌘1
as List ⌘2
as Columns ⌘3
as Gallery ⌘4
Use Stacks ^⌘0
Sort By
Clean Up
Clean Up By
Hide Sidebar ⌥⌘S
Hide Preview ⇧⌘P
Hide Toolbar ⌥⌘T
Show All Tabs ⇧⌘\
Hide Tab Bar ⇧⌘T
Hide Path Bar ⌥⌘P
Hide Status Bar ⌘/
Customize Toolbar…
Show View Options ⌘J
Show Preview Options
Enter Full Screen ⇧⌘F
Name ⌥⌘1
Kind ⌥⌘2
Date Modified ⌥⌘5
Date Created
Size ⌥⌘6
Tags ⌥⌘7

```
root = tk.Tk()

menu = tk.Menu(root)
root.config(menu=menu)

file_menu = tk.Menu(menu)

menu.add_cascade(   a single menu column
    label="File",
    menu=file_menu    cascade from here
)

file_menu.add_command(
    label="Handle",
    command=handler
)
```



Exercise

Write a GUI that *changes* the background and foreground colours.



Scissor 7

Answer: `Ex04.py`

Exercise

Write a program that reads a file like the one below and creates the menus. The menu options should just *echo their label* to bash.

```
File
    New
    Open
    Save
        Save
        Save as
View
    Zoom Out
    Zoom In
```

Summary

There are widgets for *menus*, *dialogs*, and *text boxes*.

Next

The Model-View-Controller design pattern.