KIT100 Programming Preparation

Tutorial Ten – Week 11

Today’s Flow

- Walk through Portfolio Tasks 11.1DN, 11.2 DN, and 11.3HD
- Complete Test 2

  and/or

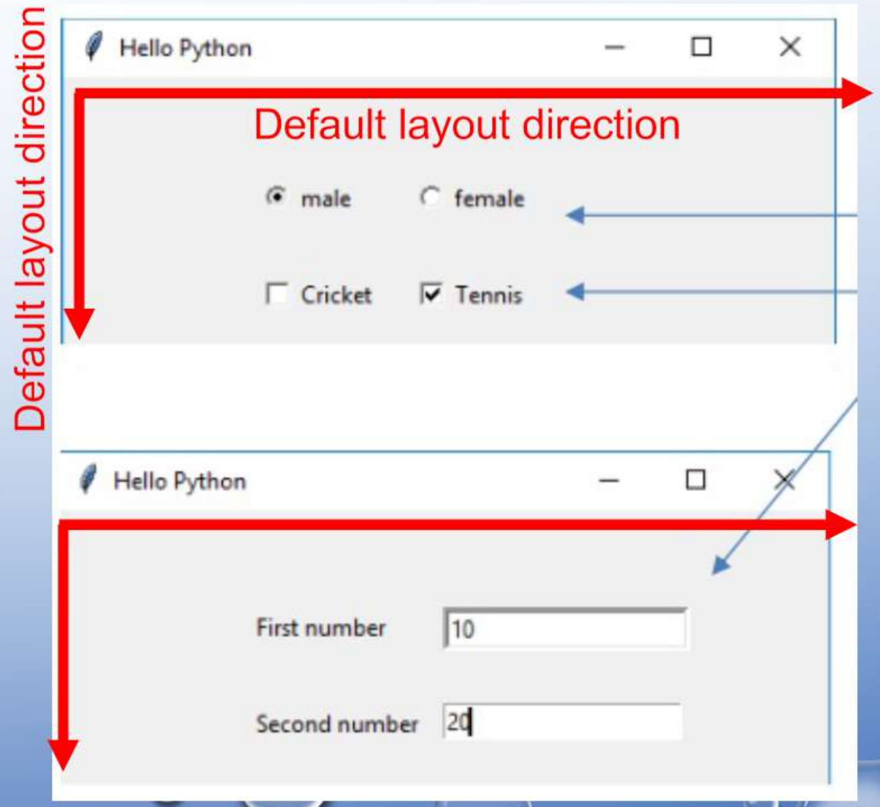
- One-on-one session with me
  - to help you to resolve the programming concerns from the previous weeks

Test 2

- Don’t forget we have Test 2 **this week** in MyLO.
- You have to complete Test 2 by
  13 May (Friday) 5pm.

Python GUI Programming

- Graphical User Interface uses graphic (windows, icons, menus, buttons etc) to carry out commands.
- Provides better user experience, users don't need to remember the commands
- **tkinter (tk)** is a library to provides GUI features in Python



Default layout direction

Default layout direction

Hello Python

● male ○ female

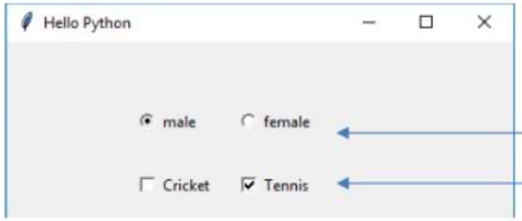
□ Cricket ☑ Tennis

Hello Python

First number 10

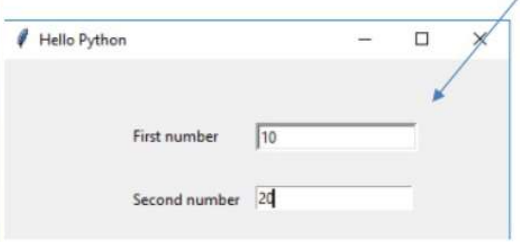
Second number 20

Tk Widget Class



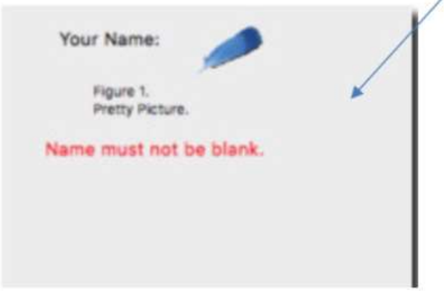
● male ○ female

□ Cricket ☑ Tennis



First number 10

Second number 20



Your Name:

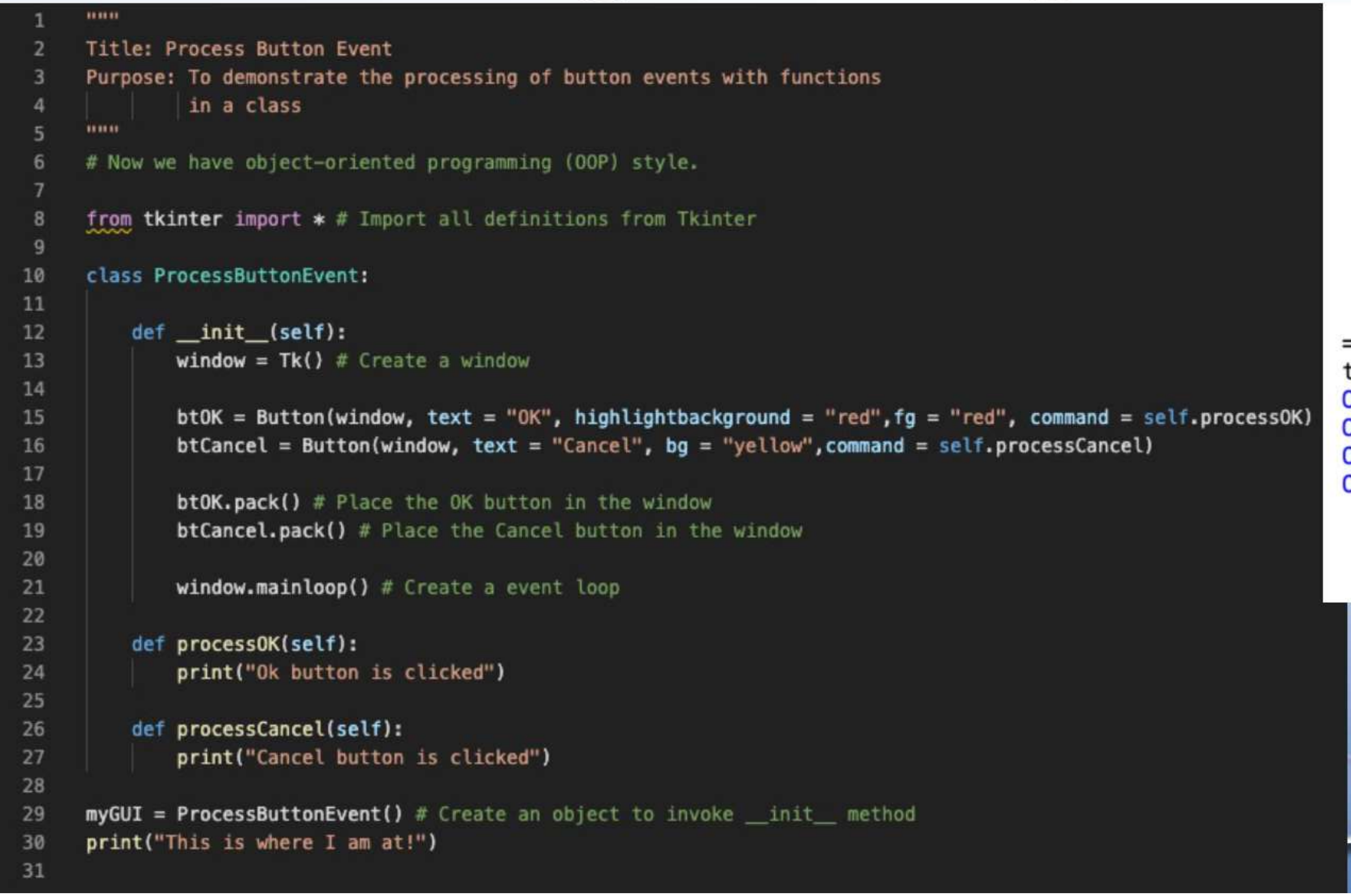
Figure 1.
Pretty Picture.

Name must not be blank.

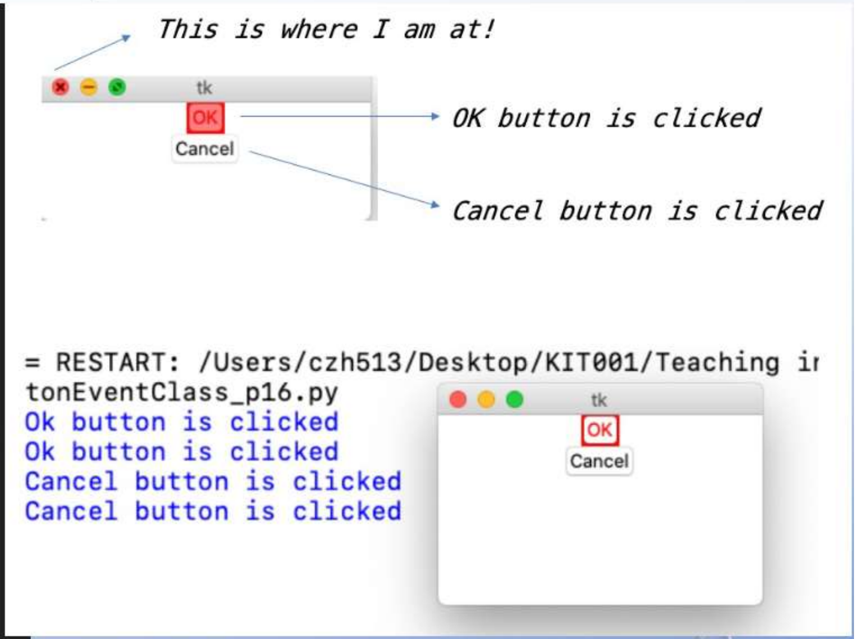
Widget Class	Description
Button	A simple button, used to execute a command.
Radiobutton	Clicking a radio button sets the variable to the value.
Checkbutton	Clicking a check button toggles between the values.
Entry	A text entry field ~ a text field or a text box.
Frame	A container widget placed inside a window.
Menu	A menu pane, used to implement pull down and popup menus.
Menubutton	A menu button, used to implement pull down menus.
Label	Displays a text or an image.
Message	Displays a text. Similar to the label widget, but can automatically wrap text to a given width or aspect ratio.
Text	Formatted text display. Allows you to display and edit text with various styles and attributes.

(Lecture 10 pp.18)

GUI Programming Example



```python
"""
Title: Process Button Event
Purpose: To demonstrate the processing of button events with functions
         in a class
"""
# Now we have object-oriented programming (OOP) style.

from tkinter import * # Import all definitions from Tkinter

class ProcessButtonEvent:

    def __init__(self):
        window = Tk() # Create a window

        btOK = Button(window, text = "OK", highlightbackground = "red",fg = "red", command = self.processOK)
        btCancel = Button(window, text = "Cancel", bg = "yellow",command = self.processCancel)

        btOK.pack() # Place the OK button in the window
        btCancel.pack() # Place the Cancel button in the window

        window.mainloop() # Create a event loop

    def processOK(self):
        print("Ok button is clicked")

    def processCancel(self):
        print("Cancel button is clicked")

myGUI = ProcessButtonEvent() # Create an object to invoke __init__ method
print("This is where I am at!")
```



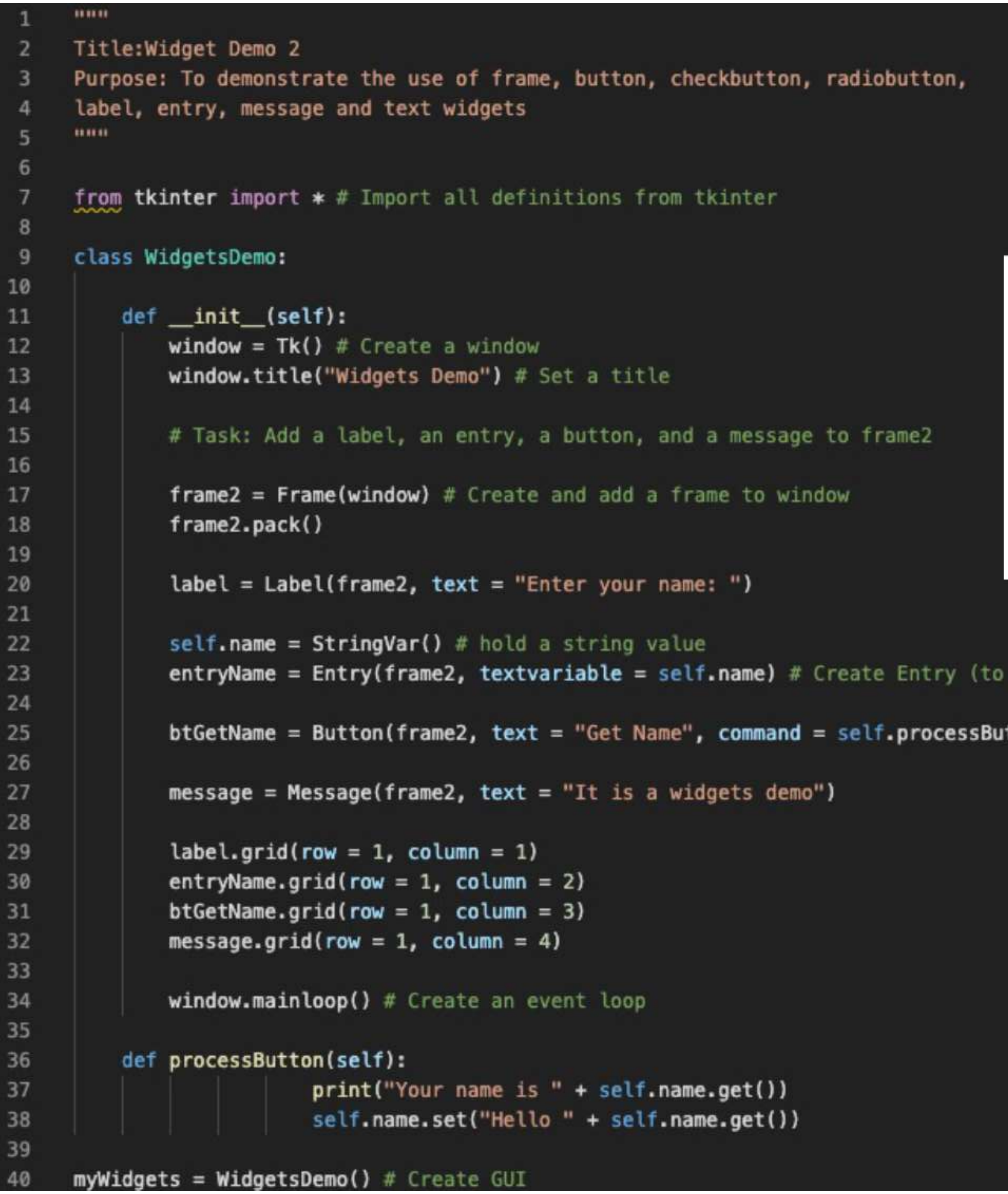
This is where I am at!

OK button is clicked

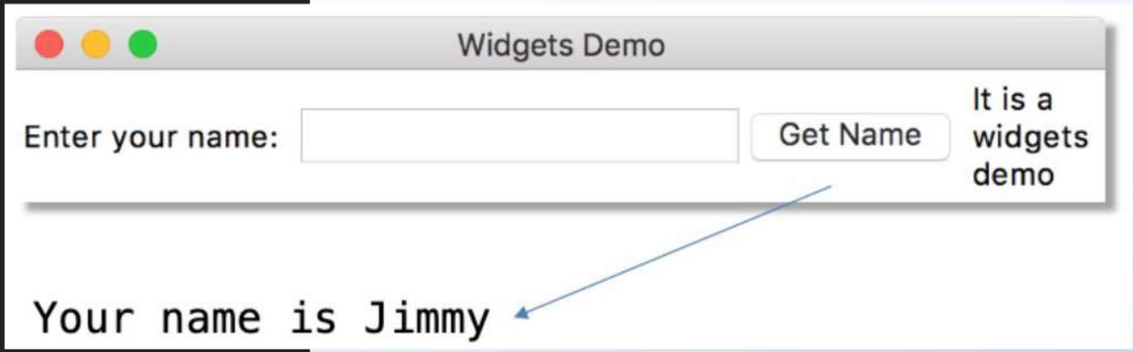
Cancel button is clicked

```
= RESTART: /Users/czh513/Desktop/KIT001/Teaching ir
tonEventClass_p16.py
Ok button is clicked
Ok button is clicked
Cancel button is clicked
Cancel button is clicked
```

(Lecture 10 pp.16)

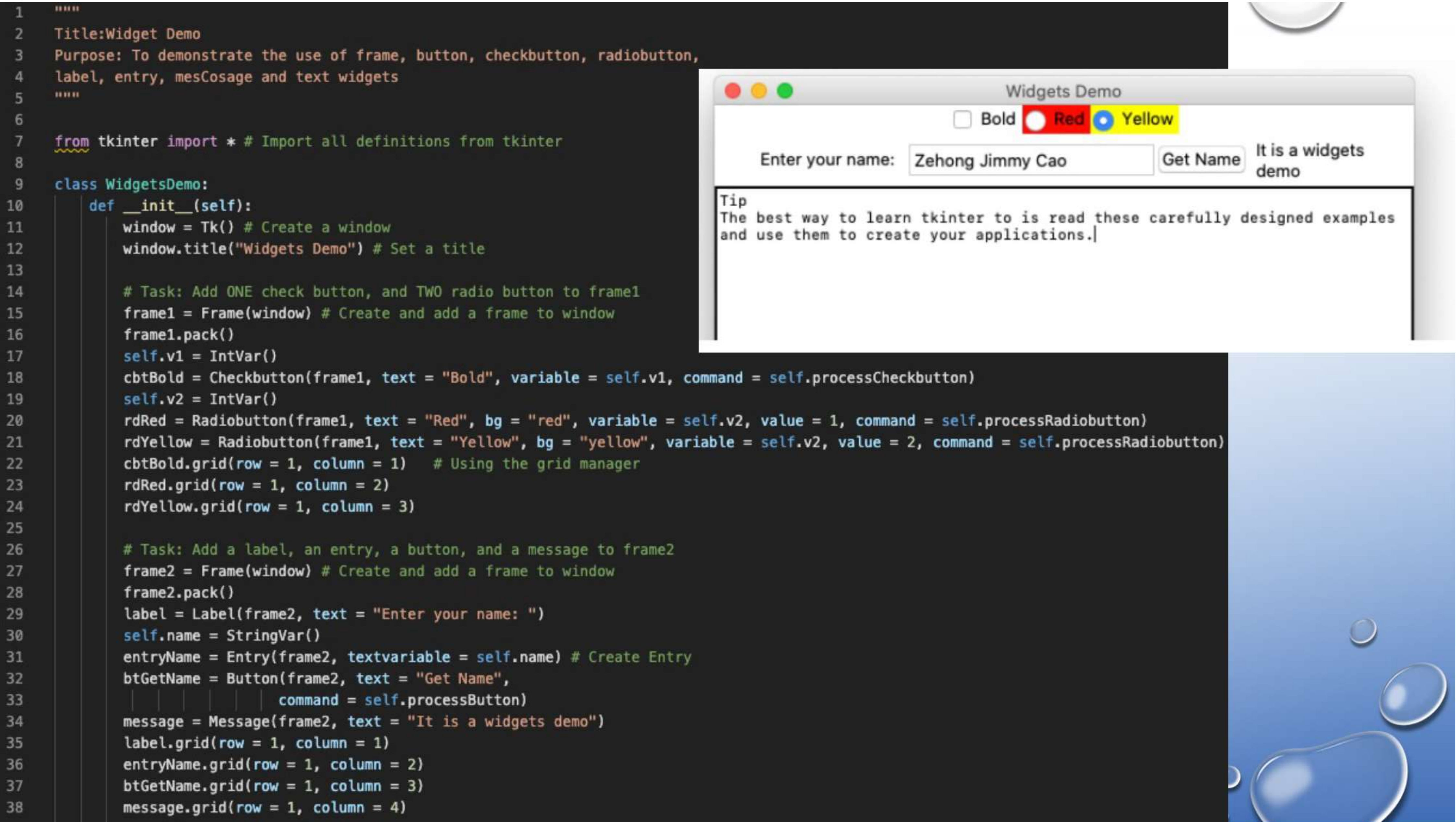


```python
"""
Title:Widget Demo 2
Purpose: To demonstrate the use of frame, button, checkbutton, radiobutton,
label, entry, message and text widgets
"""

from tkinter import * # Import all definitions from tkinter

class WidgetsDemo:

    def __init__(self):
        window = Tk() # Create a window
        window.title("Widgets Demo") # Set a title

        # Task: Add a label, an entry, a button, and a message to frame2

        frame2 = Frame(window) # Create and add a frame to window
        frame2.pack()

        label = Label(frame2, text = "Enter your name: ")

        self.name = StringVar() # hold a string value
        entryName = Entry(frame2, textvariable = self.name) # Create Entry (to entre the name)

        btGetName = Button(frame2, text = "Get Name", command = self.processButton) # get the button

        message = Message(frame2, text = "It is a widgets demo")

        label.grid(row = 1, column = 1)
        entryName.grid(row = 1, column = 2)
        btGetName.grid(row = 1, column = 3)
        message.grid(row = 1, column = 4)

        window.mainloop() # Create an event loop

    def processButton(self):
                print("Your name is " + self.name.get())
                self.name.set("Hello " + self.name.get())

myWidgets = WidgetsDemo() # Create GUI
```



```
Your name is Jimmy
```

(Lecture 10 pp.27-28)



```python
"""
Title:Widget Demo
Purpose: To demonstrate the use of frame, button, checkbutton, radiobutton,
label, entry, mesCosage and text widgets
"""

from tkinter import * # Import all definitions from tkinter

class WidgetsDemo:
    def __init__(self):
        window = Tk() # Create a window
        window.title("Widgets Demo") # Set a title

        # Task: Add ONE check button, and TWO radio button to frame1
        frame1 = Frame(window) # Create and add a frame to window
        frame1.pack()
        self.v1 = IntVar()
        cbtBold = Checkbutton(frame1, text = "Bold", variable = self.v1, command = self.processCheckbutton)
        self.v2 = IntVar()
        rdRed = Radiobutton(frame1, text = "Red", bg = "red", variable = self.v2, value = 1, command = self.processRadiobutton)
        rdYellow = Radiobutton(frame1, text = "Yellow", bg = "yellow", variable = self.v2, value = 2, command = self.processRadiobutton)
        cbtBold.grid(row = 1, column = 1)    # Using the grid manager
        rdRed.grid(row = 1, column = 2)
        rdYellow.grid(row = 1, column = 3)

        # Task: Add a label, an entry, a button, and a message to frame2
        frame2 = Frame(window) # Create and add a frame to window
        frame2.pack()
        label = Label(frame2, text = "Enter your name: ")
        self.name = StringVar()
        entryName = Entry(frame2, textvariable = self.name) # Create Entry
        btGetName = Button(frame2, text = "Get Name",
                            command = self.processButton)
        message = Message(frame2, text = "It is a widgets demo")
        label.grid(row = 1, column = 1)
        entryName.grid(row = 1, column = 2)
        btGetName.grid(row = 1, column = 3)
        message.grid(row = 1, column = 4)
```

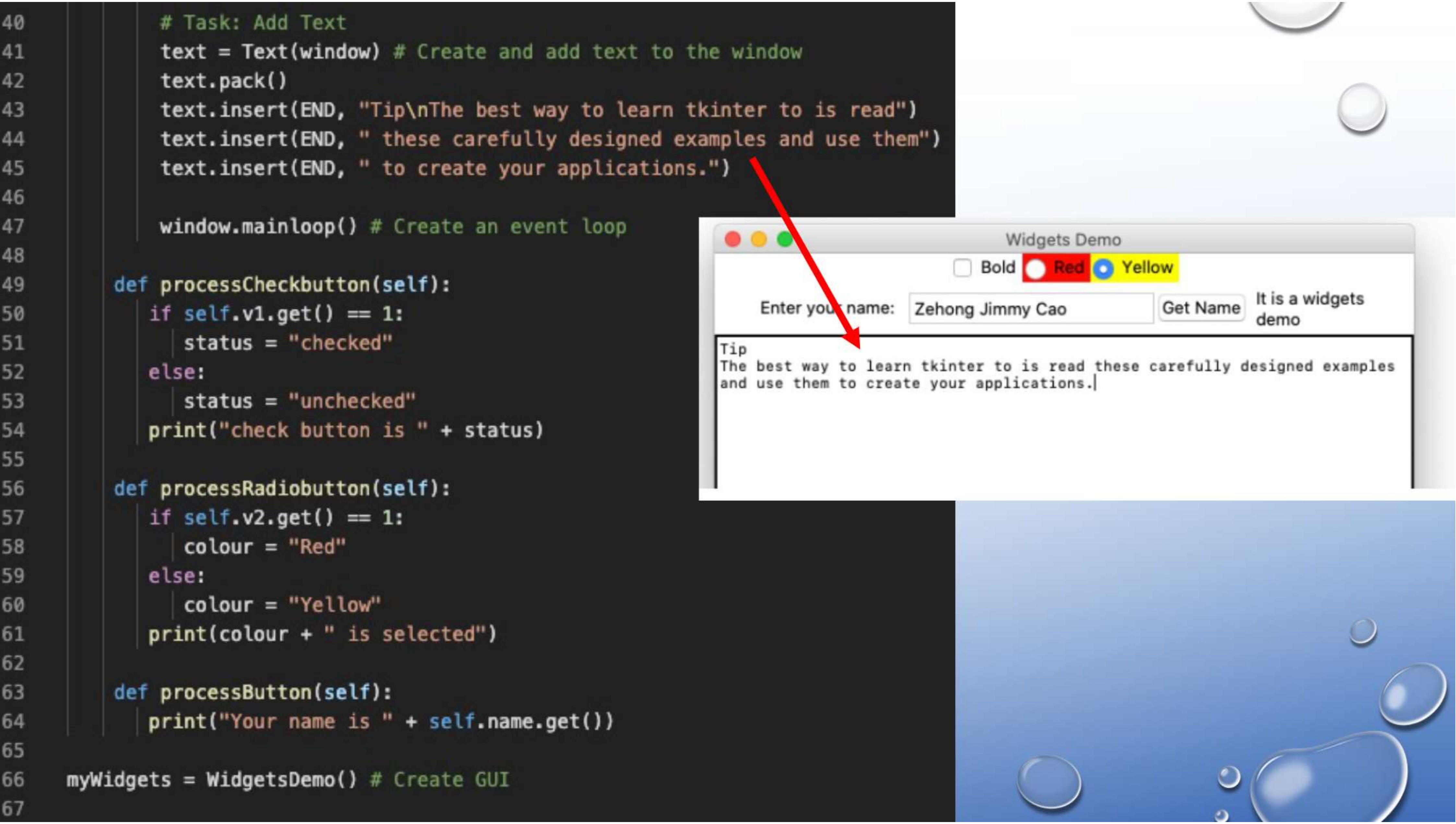
Widgets Demo

□ Bold ○ Red ● Yellow

Enter your name: Zehong Jimmy Cao

It is a widgets demo

```
Tip
The best way to learn tkinter to is read these carefully designed examples
and use them to create your applications.
```



```
        # Task: Add Text
        text = Text(window) # Create and add text to the window
        text.pack()
        text.insert(END, "Tip\nThe best way to learn tkinter to is read")
        text.insert(END, " these carefully designed examples and use them")
        text.insert(END, " to create your applications.")

        window.mainloop() # Create an event loop

    def processCheckbutton(self):
        if self.v1.get() == 1:
            status = "checked"
        else:
            status = "unchecked"
        print("check button is " + status)

    def processRadiobutton(self):
        if self.v2.get() == 1:
            colour = "Red"
        else:
            colour = "Yellow"
        print(colour + " is selected")

    def processButton(self):
        print("Your name is " + self.name.get())

myWidgets = WidgetsDemo() # Create GUI
```

Enter your name: Zehong Jimmy Cao

It is a widgets demo

```
Tip
The best way to learn tkinter to is read these carefully designed examples
and use them to create your applications.
```

11.1DN – GUI Statements

Description:

Your lecturer wants you to have some more experience with tkinter, you should be able to follow and understand GUI statements

Task:

Submit a ***plain text*** document (i.e. use Notepad or TextEdit) that contains the answers to the questions that follow the code below:

```
  1 from tkinter import *
  2
  3 class ProcessButtonEvent:
  4     def __init__(self):
  5         window = Tk()
  6         btOK = Button(window, text = "OK", fg = "red", command =
self.processOK)
  7         btCancel = Button(window, text = "Cancel", bg = "yellow",  command
= self.processCancel)
  8         btOK.pack()
  9         btCancel.pack()
 10         window.mainloop()
 11
 12     def processOK(self):
 13         print("OK button is clicked")
 14
 15     def processCancel(self):
 16         print("Cancel button is clicked")
 17
 18 myGUI = ProcessButtonEvent()
```

For each of the 14 non-blank lines in the code above i.e. line numbers coloured red, explain the purpose of each statement. Your text document should consist of a line number, followed by your description of the purpose of the statement.

Hint:

Examples: (these are correct)

line 1: import all code definitions from the tkinter class
line 3: start to define the ProcessButtonEvent class
line 18: create a ProcessButtonEvent object (and run the __init__ method) and assign it to myGUI

Submission Details

Upload the following to the MyLO submission folder for this task:

1. The text file (i.e. the a plain text file containing your answers)

Hints: Week10 Lecture Slide – examples on pp.16

11.2DN – GUI

Description:

The Galactic Bank has an unfortunate dilemma - citizens are becoming too engaged and agitated. Fidget Spinners and Anti-stress fidget cubes are now a passing fad and not occupying citizens' time for long enough. The bank asks you to design a time-waster GUI application they can then use to occupy the populace while they come up with another diabolical plan.

Task:

Your task is to create the GUI shown below, which has two labels, one text entry field and two buttons. This GUI allows the user to enter a 'click count' into the text entry field and then click a 'Fidget Click' button to see that number decreased by one every click. A status label displays a message to the user with the following values:

Click count	Status Message
More than 10	Keep going...
More than 5 but less than or equal to 10	You are nearly there!
More than 0 but less than or equal to 5	X more to go
0	You must enter another fidget click count

Where X above is the current number of clicks..

Hint

The *kilometre to miles converter* code shown in lectures is a very good template to use for this task. You can force a label to be a specific width and left-justified by using the following options in the Label method e.g.

```
width=30,anchor="w"
```

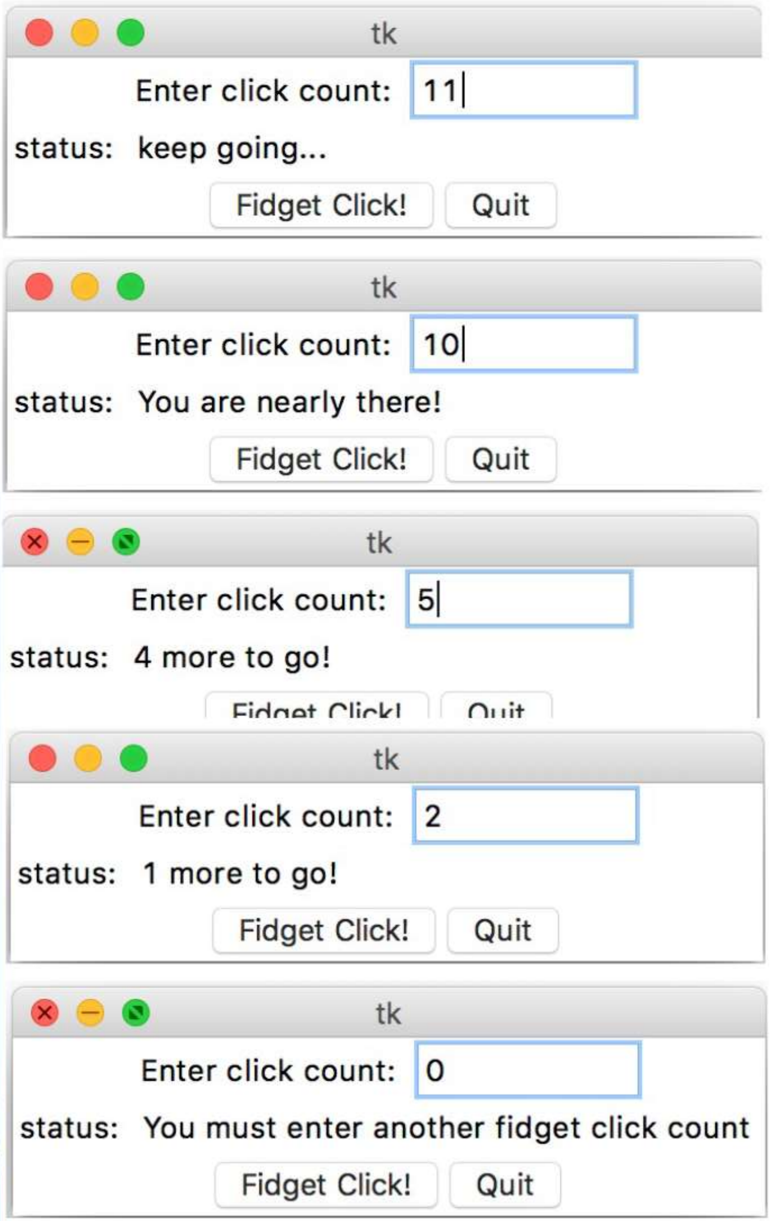
Submission Details

Upload the following to the MyLO submission folder for this task:

1. The source file (i.e. the text file containing your code)
2. Screenshots of the Python GUI window that shows the **execution** results of the source code

Hints: Week10 Lecture Slide – Demo 2, Demo 4

Example GUI:



Enter click count: 11
status: keep going...

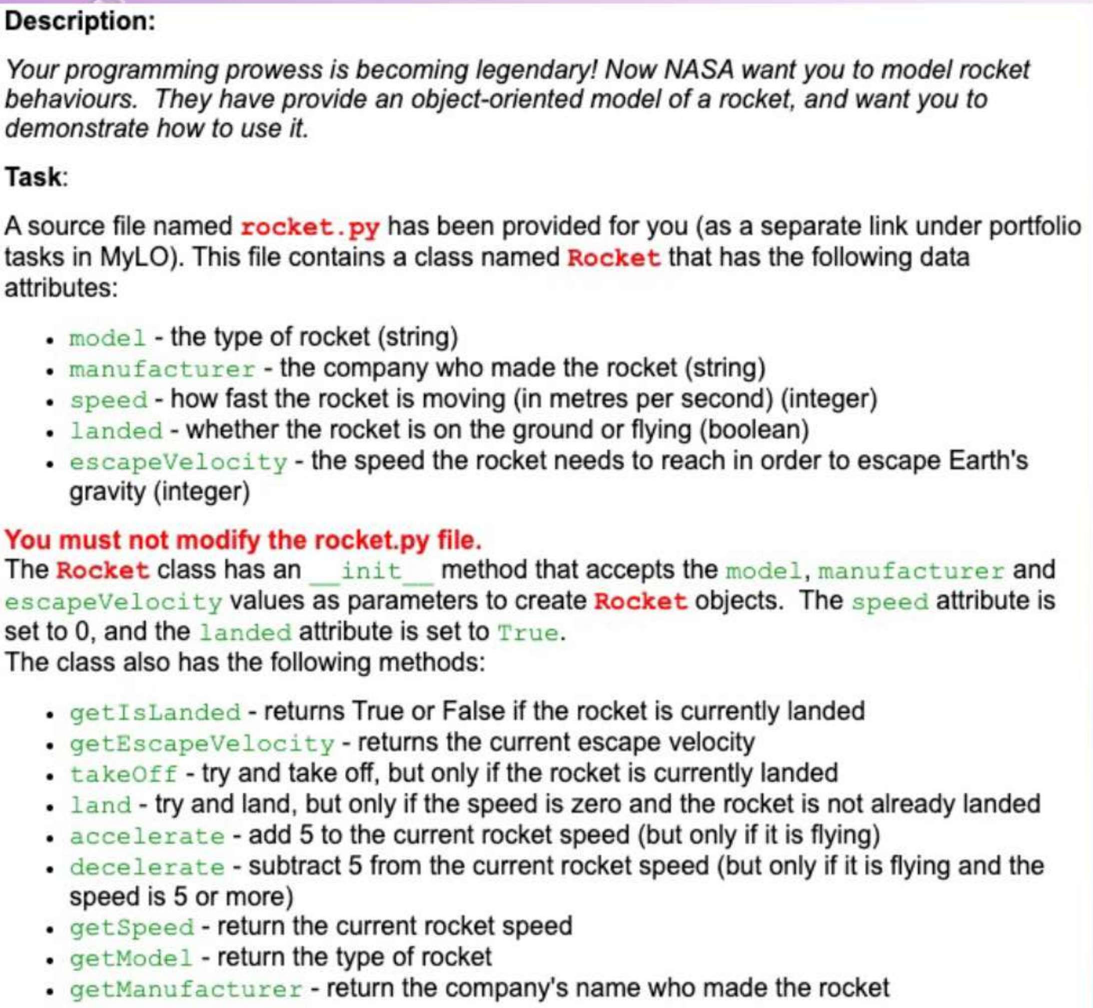
Enter click count: 10
status: You are nearly there!

Enter click count: 5
status: 4 more to go!

Enter click count: 2
status: 1 more to go!

Enter click count: 0
status: You must enter another fidget click count

11.3HD – Complex Class Definition and Usage



Description:

Your programming prowess is becoming legendary! Now NASA want you to model rocket behaviours. They have provide an object-oriented model of a rocket, and want you to demonstrate how to use it.

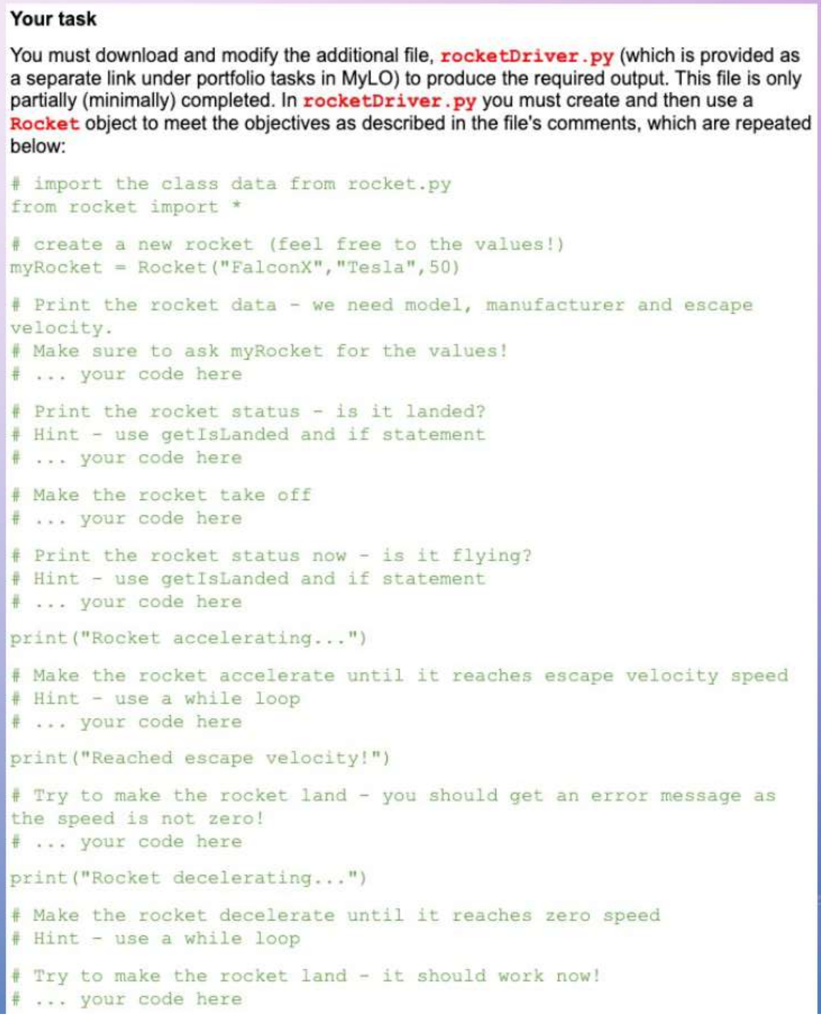
Task:

A source file named **rocket.py** has been provided for you (as a separate link under portfolio tasks in MyLO). This file contains a class named **Rocket** that has the following data attributes:

- `model` - the type of rocket (string)
- `manufacturer` - the company who made the rocket (string)
- `speed` - how fast the rocket is moving (in metres per second) (integer)
- `landed` - whether the rocket is on the ground or flying (boolean)
- `escapeVelocity` - the speed the rocket needs to reach in order to escape Earth's gravity (integer)

You must not modify the rocket.py file.
The **Rocket** class has an `__init__` method that accepts the `model`, `manufacturer` and `escapeVelocity` values as parameters to create **Rocket** objects. The `speed` attribute is set to 0, and the `landed` attribute is set to `True`.
The class also has the following methods:

- `getIsLanded` - returns True or False if the rocket is currently landed
- `getEscapeVelocity` - returns the current escape velocity
- `takeOff` - try and take off, but only if the rocket is currently landed
- `land` - try and land, but only if the speed is zero and the rocket is not already landed
- `accelerate` - add 5 to the current rocket speed (but only if it is flying)
- `decelerate` - subtract 5 from the current rocket speed (but only if it is flying and the speed is 5 or more)
- `getSpeed` - return the current rocket speed
- `getModel` - return the type of rocket
- `getManufacturer` - return the company's name who made the rocket



Your task

You must download and modify the additional file, **rocketDriver.py** (which is provided as a separate link under portfolio tasks in MyLO) to produce the required output. This file is only partially (minimally) completed. In **rocketDriver.py** you must create and then use a **Rocket** object to meet the objectives as described in the file's comments, which are repeated below:

```
# import the class data from rocket.py
from rocket import *

# create a new rocket (feel free to the values!)
myRocket = Rocket("FalconX","Tesla",50)

# Print the rocket data - we need model, manufacturer and escape
velocity.
# Make sure to ask myRocket for the values!
# ... your code here

# Print the rocket status - is it landed?
# Hint - use getIsLanded and if statement
# ... your code here

# Make the rocket take off
# ... your code here

# Print the rocket status now - is it flying?
# Hint - use getIsLanded and if statement
# ... your code here

print("Rocket accelerating...")

# Make the rocket accelerate until it reaches escape velocity speed
# Hint - use a while loop
# ... your code here

print("Reached escape velocity!")

# Try to make the rocket land - you should get an error message as
the speed is not zero!
# ... your code here

print("Rocket decelerating...")

# Make the rocket decelerate until it reaches zero speed
# Hint - use a while loop

# Try to make the rocket land - it should work now!
# ... your code here
```