Chapter 18

How to build a GUI program

Objectives

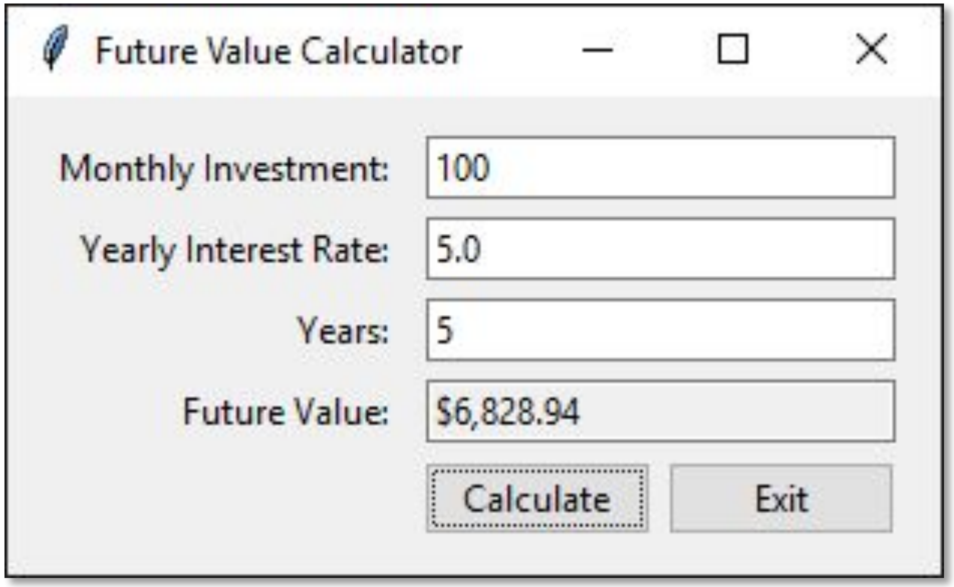
Applied

1. Develop a GUI program that has a user interface that consists of frames, buttons, labels, and text entry fields in a grid format.

Knowledge

1. Describe the need for the mainloop() method of a tkinter root window in terms of the event processing loop.
2. Describe the way an event handler works with a GUI component like a button.
3. Describe how the grid() method is used to lay out the components in a frame.
4. Describe the reason for creating a subclass of the ttk.Frame class when you're building a GUI.

A window with ten components



Monthly Investment: 100
Yearly Interest Rate: 5.0
Years: 5
Future Value: $6,828.94

Creating a GUI application using Tkinter

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

The constructor of the root window

```
Tk()
```

The methods of the root window

```
title(title)
geometry(str)
mainloop()
```

How to import the tkinter module

```
import tkinter as tk
```

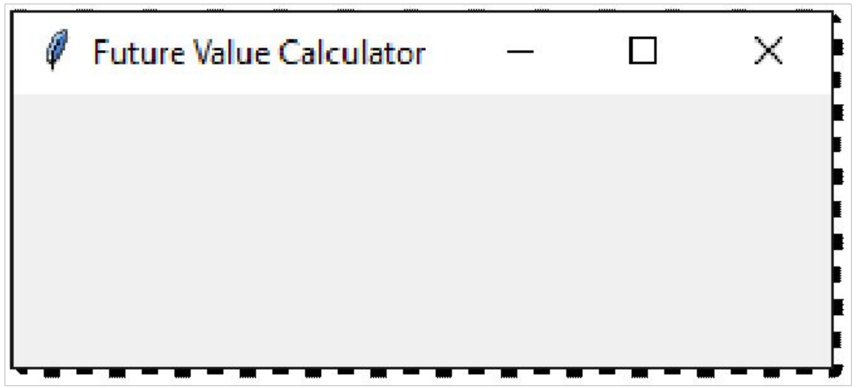
How to create an empty root window

```
root = tk.Tk()
root.title("Future Value Calculator")
root.geometry("300x200")
```

How to make the root window visible

```
root.mainloop()
```

An empty root window



Tkinter Widgets

Operator & Description
Frame: The Frame widget is used as a container widget to organize other widgets.
Button: The Button widget is used to display buttons in your application.
Label: The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
Entry: The Entry widget is used to display a single-line text field for accepting values from a user.

Two constructors of the ttk module

```
Frame(parent[, padding])
Button(parent, text)
```

A method for working with all components

```
pack([fill][, expand])
```

How to import the ttk module

```
from tkinter import ttk
```

How to add a frame to the root window

```
frame = ttk.Frame(root, padding="10 10 10 10")
frame.pack(fill=tk.BOTH, expand=True)
```

The padding allows you to add extra space around the inside of the frame.
Paddings are in pixels. And you can specify padding for each side of the frame

pack - This geometry manager organizes widgets in blocks before placing them in the parent widget.

Syntax

```
widget.pack( pack_options )
```

Here is the list of possible options −

- **expand** − When set to true, widget expands to fill any space not otherwise used in widget's parent.
- **fill** − Determines whether widget fills any extra space allocated to it by the packer, or keeps its own minimal dimensions: NONE (default), X (fill only horizontally), Y (fill only vertically), or BOTH (fill both horizontally and vertically).
- **side** − Determines which side of the parent widget packs against: TOP (default), BOTTOM, LEFT, or RIGHT.

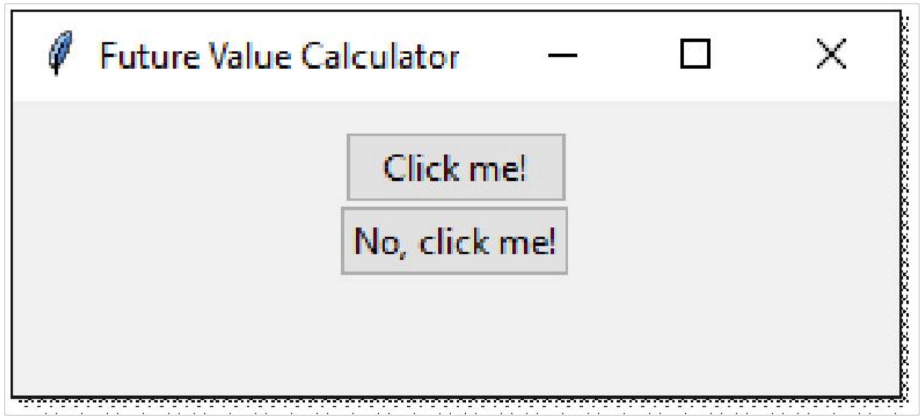
How to add two buttons to the frame

```
button1 = ttk.Button(frame, text="Click me!")
button2 = ttk.Button(frame, text="No, click me!")
```

How to display the buttons

```
button1.pack()
button2.pack()
```

Two buttons in a frame



An argument of the Button constructor

```
command
```

The destroy() method of the root window

```
destroy()
```

How to connect two buttons to callback functions

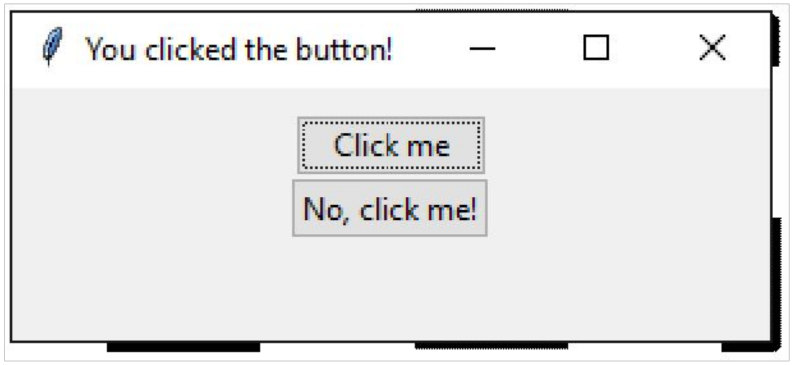
```
button1 = ttk.Button(frame, text="Click me",
                     command=click_button1)
button2 = ttk.Button(frame, text="No, click me!",
                     command=click_button2)
```

The callback functions

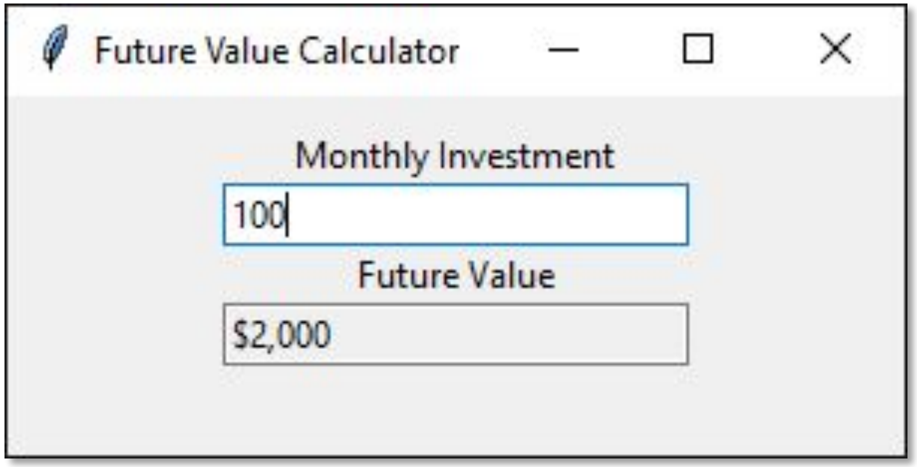
```
def click_button1():
    root.title("You clicked the button!")

def click_button2():
    root.destroy()
```

The GUI after the user clicks the first button



A window with labels and text entry fields


Future Value Calculator

Monthly Investment

100

Future Value

$2,000

Constructors for labels and text entry fields

```
Label(parent, text)
Entry(parent, width, textvariable[, state])
```

How to create a label and display it

```
investmentLabel = ttk.Label(frame,
                           text="Monthly Investment")
investmentLabel.pack()
```

Another way to create a label and display it

```
ttk.Label(frame, text="Monthly Investment").pack()
```

Constructors and methods of the StringVar class

```
StringVar()
get()
set(str)
```

How to bind a text entry field to a StringVar object

```
investmentText = tk.StringVar()
investmentEntry = ttk.Entry(frame, width=25,
                            textvariable=investmentText)
```

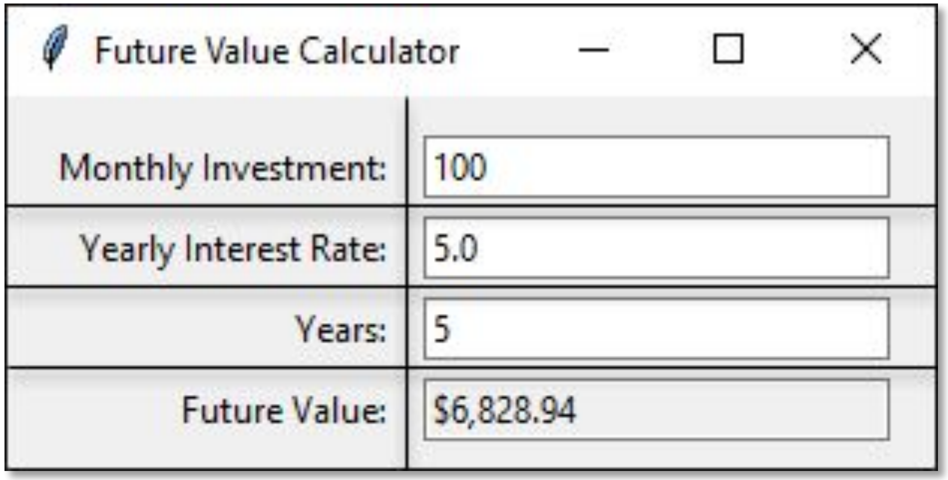
How to create a read-only text entry field

```
fvText = tk.StringVar()
fvEntry = ttk.Entry(frame, width=25, textvariable=fvText,
                    state="readonly")
```

How to get or set a string in a text entry field

```
investment = investmentText.get()
fvText.set("$2,000")
```

Eight components in a grid


Future Value Calculator

Monthly Investment:	100
Yearly Interest Rate:	5.0
Years:	5
Future Value:	$6,828.94

Some arguments of the grid() method

```
column
row
sticky
padx
pady
columnspan
rowspan
```

How to lay out components in a grid

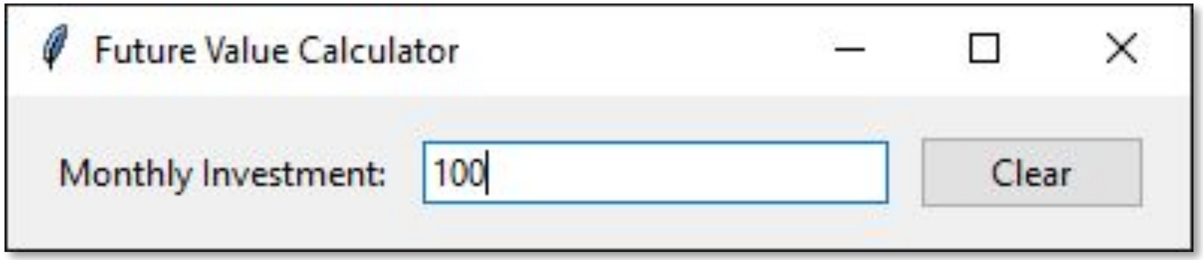
```
ttk.Label(frame, text="Monthly Investment:").grid(
    column=0, row=0, sticky=tk.E)
ttk.Entry(frame, width=25,
          textvariable=investmentText).grid(
    column=1, row=0)

ttk.Label(frame, text="Yearly Interest Rate:").grid(
    column=0, row=1, sticky=tk.E)
ttk.Entry(frame, width=25, textvariable=rateText).grid(
    column=1, row=1)
```

How to add padding to all components in a frame

```
for child in frame.winfo_children():      # get children
    child.grid_configure(padx=5, pady=3) # pad each child
```

A window that contains a frame



A class that defines a frame

```
import tkinter as tk
from tkinter import ttk

class InvestmentFrame(ttk.Frame):
    def __init__(self, parent):
        ttk.Frame.__init__(self, parent, padding="10 10 10 10")
        self.pack(fill=tk.BOTH, expand=True)

        # Define string variable for the entry field
        self.monthlyInvestment = tk.StringVar()

        # Create a label, an entry field, and a button
        ttk.Label(self, text="Monthly Investment:").grid(
            column=0, row=0, sticky=tk.E)
        ttk.Entry(self, width=25,
                  textvariable=self.monthlyInvestment).grid(
            column=1, row=0)
        ttk.Button(self, text="Clear", command=self.clear).grid(
            column=2, row=0)
```

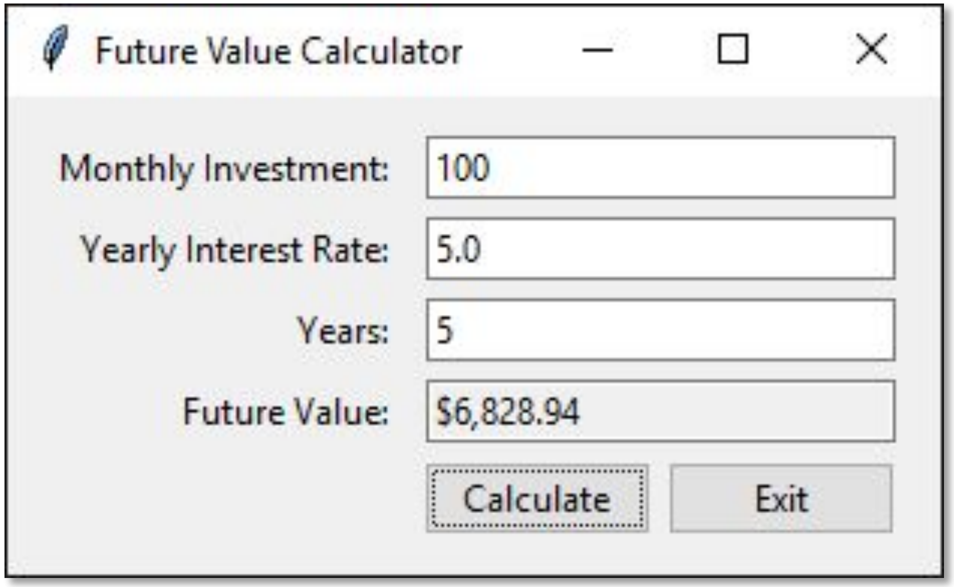
A class that defines a frame (cont.)

```
        # Add padding to all child components
        for child in self.winfo_children():
            child.grid_configure(padx=5, pady=3)

    # Define the callback method for the Clear button
    def clear(self):
        print("Monthly Investment:",
              self.monthlyInvestment.get())
        self.monthlyInvestment.set("")

if __name__ == "__main__":
    root = tk.Tk()                  # Create the root window
    root.title("Future Value Calculator")
    InvestmentFrame(root)           # Create the frame
    root.mainloop()                 # Display the frame
```

The GUI for the Future Value Calculator



Monthly Investment: 100

Yearly Interest Rate: 5.0

Years: 5

Future Value: $6,828.94

The business module

```
class Investment():
    def __init__(self):
        self.monthlyInvestment = 0
        self.yearlyInterestRate = 0
        self.years = 0

    def calculateFutureValue(self):
        monthlyInterestRate = self.yearlyInterestRate / 12 / 100
        months = self.years * 12

        futureValue = 0
        for i in range(months):
            futureValue += self.monthlyInvestment
            monthlyInterestAmount = futureValue *
                                    monthlyInterestRate
            futureValue += monthlyInterestAmount

        return futureValue
```

The ui module

```
import tkinter as tk
from tkinter import ttk
import locale

from business import Investment

class FutureValueFrame(ttk.Frame):
    def __init__(self, parent):
        ttk.Frame.__init__(self, parent, padding="10 10 10 10")
        self.parent = parent
        self.investment = Investment()
        result = locale.setlocale(locale.LC_ALL, '')
        if result == 'C':
            locale.setlocale(locale.LC_ALL, 'en_US')

        # Define string variables for text entry fields
        self.monthlyInvestment = tk.StringVar()
        self.yearlyInterestRate = tk.StringVar()
        self.years = tk.StringVar()
        self.futureValue = tk.StringVar()

        self.initComponents()
```

The ui module (continued)

```
def initComponents(self):
    self.pack()
    ttk.Label(self, text="Monthly Investment:").grid(
        column=0, row=0, sticky=tk.E)
    ttk.Entry(self, width=25,
              textvariable=self.monthlyInvestment).grid(
        column=1, row=0)

    ttk.Label(self, text="Yearly Interest Rate:").grid(
        column=0, row=1, sticky=tk.E)
    ttk.Entry(self, width=25,
              textvariable=self.yearlyInterestRate).grid(
        column=1, row=1)

    ttk.Label(self, text="Years:").grid(
        column=0, row=2, sticky=tk.E)
    ttk.Entry(self, width=25, textvariable=self.years).grid(
        column=1, row=2)
```

The ui module (continued)

```
        ttk.Label(self, text="Future Value:").grid(
            column=0, row=3, sticky=tk.E)
        ttk.Entry(self, width=25, textvariable=self.futureValue,
                  state="readonly").grid(column=1, row=3)

        self.makeButtons()

        for child in self.winfo_children():
            child.grid_configure(padx=5, pady=3)

    def makeButtons(self):
        # Create a frame to store the two buttons
        buttonFrame = ttk.Frame(self)
        buttonFrame.grid(column=0, row=4, columnspan=2, sticky=tk.E)

        ttk.Button(buttonFrame, text="Calculate",
                   command=self.calculate).grid(column=0, row=0,
                                                padx=5)
        ttk.Button(buttonFrame, text="Exit",
                   command=self.parent.destroy).grid(column=1, row=0)
```

The ui module (continued)

```
def calculate(self):
    self.investment.monthlyInvestment = float(
        self.monthlyInvestment.get())
    self.investment.yearlyInterestRate = float(
        self.yearlyInterestRate.get())
    self.investment.years = int(self.years.get())

    self.futureValue.set(locale.currency(
        self.investment.calculateFutureValue(), grouping=True))

if __name__ == "__main__":
    root = tk.Tk()
    root.title("Future Value Calculator")
    FutureValueFrame(root)
    root.mainloop()
```