Graphical User Interface (GUI) Programming

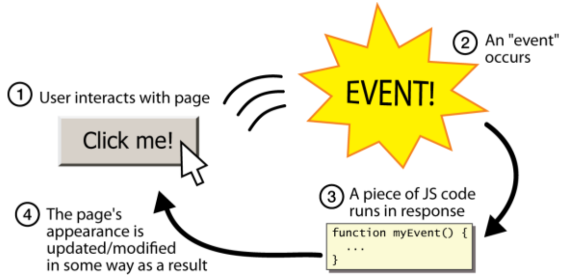
Outline

- Introduction to Python GUI programming
  – Event-driven programming
- **tkinter** widgets (program control objects)
  – **Main window**, aka the **root** widget
  – **Label** , **Entry**, **Frame**, **Radiobutton**
  – **Listbox**, **Button**, **Checkbutton**
- Geometry Management
  – Exact placement with **place()** method
  – Combining **grid()** method with frame widgets
    - Using **pack()** method to populate frames with other widgets
    - Covered in class demo

Introduction to Python GUI

- Graphical User Interfaces (**GUI**) revolutionized software industry
  - User interaction drives program execution
  - First GUI developed at Xerox Palo Alto Research Center (1979)
    - Steve Jobs picked up on the idea
    - Apple software engineers developed first GUI-based Mac in 1984
- GUI Program
  - **Design** user interface by placing widgets (aka controls: labels, entry boxes, buttons, …) on a window canvas for users to interact with
  - **UX/UI** (User Experience/Interaction) designers and developers
    - Designers - Come up with creative visual designs for optimal experience
    - Developers - code designs using a variety of software tools (like Python)

Event-Driven Programming


① User interacts with page
Click me!
② An "event" occurs
EVENT!
③ A piece of JS code runs in response
function myEvent() {
...
}
④ The page's appearance is updated/modified in some way as a result

- GUI Program
  - Main program runs an infinite loop until terminated
  - Constantly monitors widgets and detects associated events (ex. click on a button)
  - Executes Python code in “callback” functions associated with these events (click on a button calls a function)
- Traditional Programs
  - Program starts and flows in predictable fashion until it ends
- Event-Driven Programs
  - The user decides in which order the program is executed

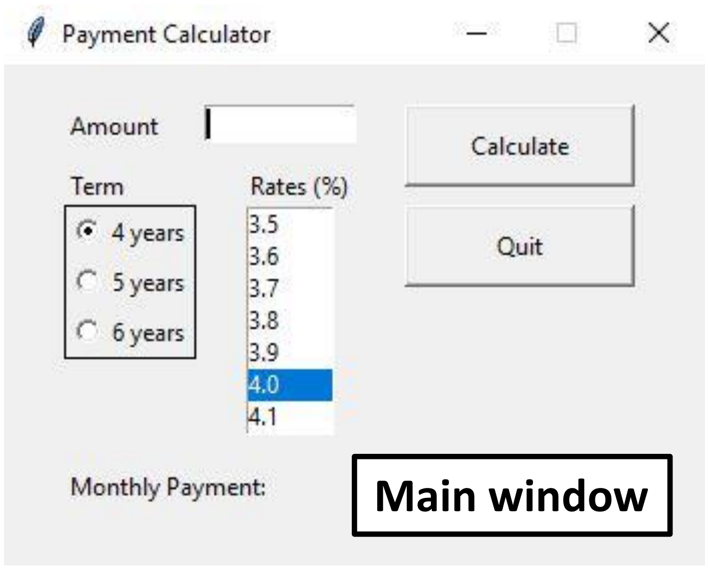
tkinter Module

Tkinter python

- Comes with standard install, relatively easy to get started
  - Other libraries: **PyQt**, **Kivy**, **wxPython**, etc..
- Overall program structure
  - See **Lect13_Payment_Calc.py**
  - import **tkinter**
  - All the code contained in a single class, ex. **PmtCalcGUI()**
  - An object (ex. **pmt_calc**) of the class instantiated in the **main()** function which calls the **__init__** constructor

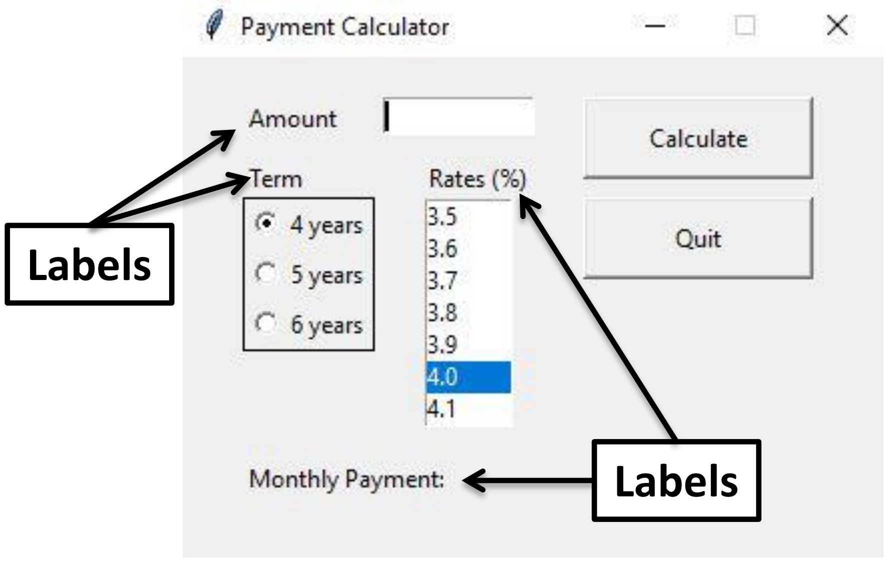
Main Window

- **PmtCalcGUI()** class
  - The **__init__** constructor acts as a main switchboard
  - Creates the main application **window**, aka, the **root** widget
    - Sets basic parameters: window title, size (geometry), etc..
  - Calls user-defined functions creating user interface and implementing functionality
    - Static and output labels, input entry box
    - Frame and radio buttons
    - List box and command buttons



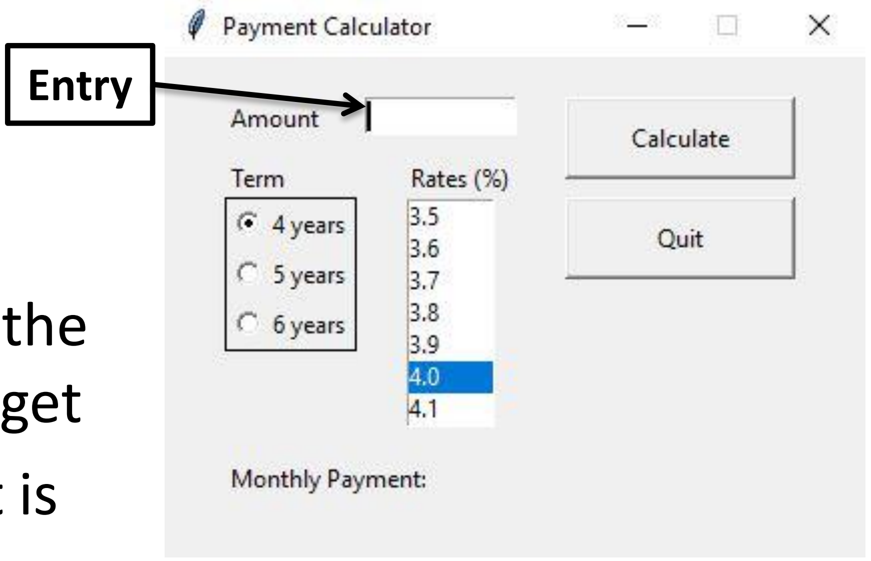
Exact Placement and Static Label Widgets

- Geometry Manager
  - **place(x, y)** method
  - **x, y**: number of pixels from the **left/top** edge of the main window
- Label widgets
  - Mostly used for static **text** display
  - Positioned using **place(x, y)** method
    - Using mostly add-hoc trial and error approach
    - Should be precisely laid out using appropriate UX design tool



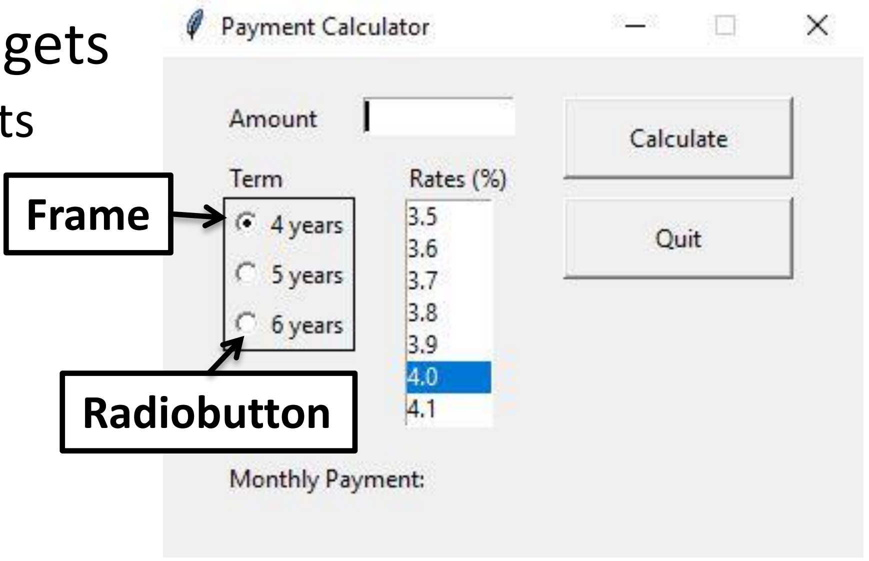
Input Entry Widget

- Generic input box
  - Content entered interpreted as text
  - There is no control over what the user enters into the **Entry** widget
  - The **width** of the Entry widget is NOT in pixels but "**text units**"
    - Ex. the width of **5** would accommodate **00000**
  - The **get()** method retrieves the content of Entry widget
  - The **focus()** method makes the Entry widget active and awaiting user input



Frame and Radiobutton Widgets

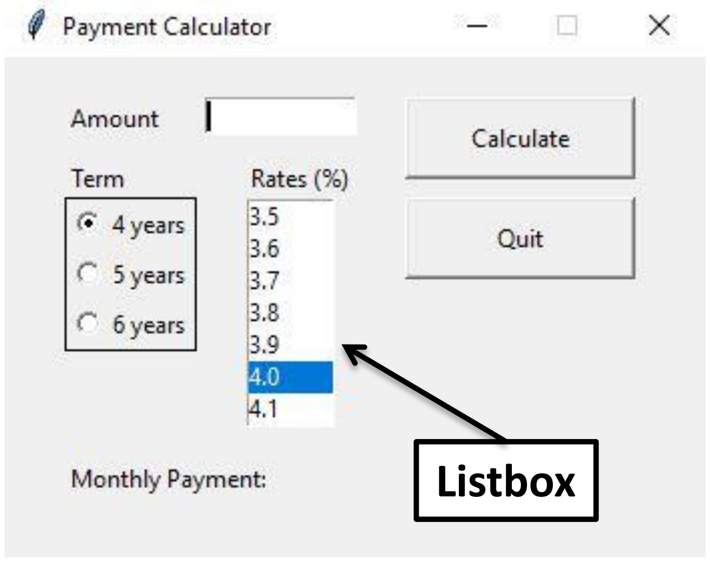
- **Frame** used to contain other widgets
  - Organization and grouping of widgets
- **Radiobutton** widgets
  - Only one radio button can be selected within a frame at one time
  - Selecting another button within a frame automatically deselects the previously selected one



- **IntVar** class
  - Used with radio buttons to assign unique values (ex., 4, 5, 6) for buttons within a single frame
  - Two-way communication between the code and radio buttons
    - Using object variable of **IntVar** class to **set()** the value, i.e., “turn on” a button
    - Using object variable to **get()** the value of the button and do something based on it

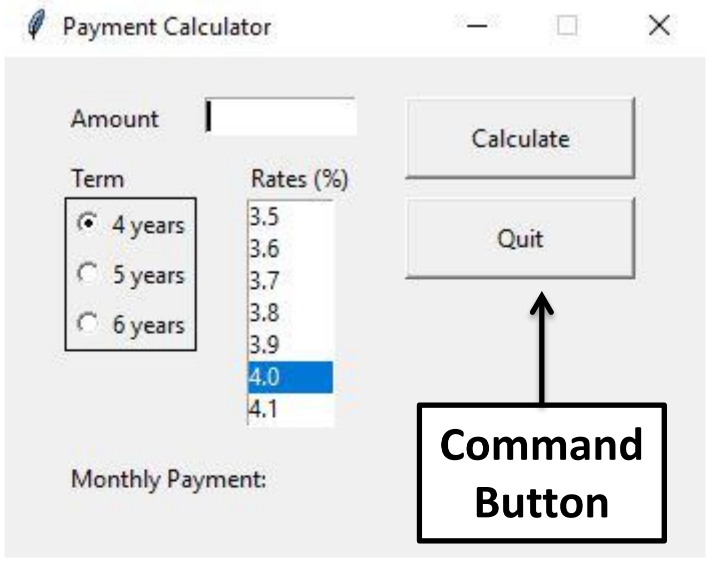
Listbox Widget

- Used to display a list of items for user selection
- Basic parameters
  - **height**: number of items shown
  - **width**: in “text units”
- Several options
  - **SINGLE**: user can select only a single item from the list
  - MULITIPLE: user can select multiple items, already selected item gets deselected
- Vertical scrolling with mouse wheel
  - Scrollbar is a separate widget (not covered)



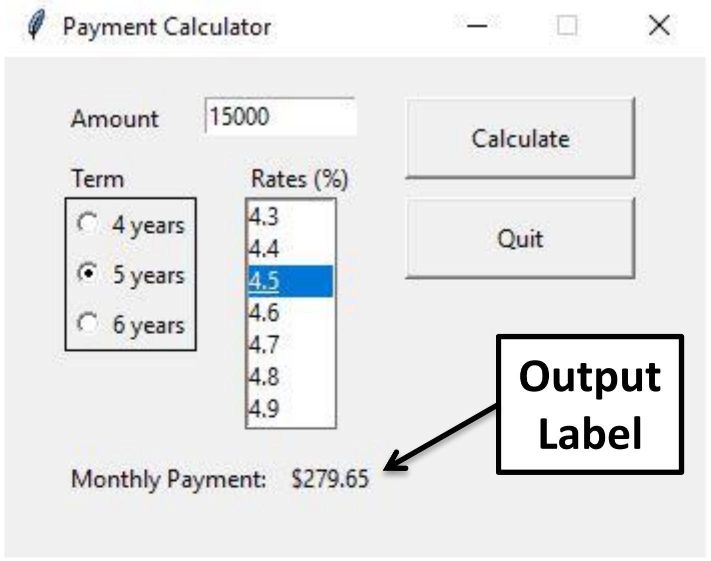
Command Button Widgets

- Used to execute program tasks
  - Calls appropriate functions
- **Callback** functions
  - Attached to the **command** attribute
  - Pre-defined or user-defined
- **User-defined** callback function
  - Gathers input from other widgets
  - Performs the needed tasks and stores the output
  - Ex. attached to the **Calculate** command button
- Pre-defined **callback** function
  - Ex: **destroy** method of the main window root widget
  - Typically attached to the **Quit** command button



Output Label Widget

- Initially empty and typically not visible
- **StringVar** class
  - Used to dynamically display output
  - Uses object variable of **StrVar** class
  - One-way communication
    - Uses **textvariable** property of **Label** widget
    - As soon as the variable of **StrVar** class is updated elsewhere in the program, the new value appears in the label
    - Variable typically updated in the callback function of a command button


Payment Calculator

Amount: 15000

Term: 4 years / [✓ 5 years] / 6 years

Rates (%): 4.3, 4.4, [✓ 4.5], 4.6, 4.7, 4.8, 4.9

Monthly Payment: $279.65

Summary

- Defined event-driven programming
  - Create Graphical User Interface (GUI) using various widgets
  - Write the code that executes when events occur (ex. button click)
  - Run the program, test interface and functionality
- Demonstrated the structure of Python's GUI program
  - Imported `tkinter` module (mentioned other alternatives)
  - Entire program is a single class, constructor acts as a switchboard
  - All the necessary **widgets** created and **callback functions** programmed to perform desired tasks
  - Widgets covered : **Label**, **Entry**, **Frame**, **Radiobutton**, **Listbox**, **Button**
  - **Checkbutton** and alternative **grid()** geometry covered in class demo