edunet
foundation

Unit - 2
Python GUI

Objective

- GUI – Graphical User Interface
- Different Python GUI Tools
- Introduction to Tkinter
- Working with Image in Tkinter

GUI – Graphical User Interface

What is a GUI ?

A **GUI** (graphical user interface) is a system of interactive visual components for computer software. A GUI displays objects that convey information and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them.

Python GUI

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications.

Different Python GUI Tools

Tkinter

Tkinter is commonly bundled with Python, using Tk and is Python's standard GUI framework. It is popular for its simplicity and graphical user interface. It is open source and available under the Python License.

Kivy

Kivy is an OpenGL ES 2 accelerated framework for the creation of new user interfaces. It supports multiple platforms namely Windows, MacOS X, Linux, Android iOS and Raspberry Pi.

Different Python GUI Tools

PyQT is one of the favored cross-platform Python bindings implementing the Qt library for the Qt (owned by Nokia) application development framework .

WxPython is an open-source wrapper for cross-platform GUI library WxWidgets (earlier known as WxWindows) and implemented as a Python extension module.

Introduction to Tkinter

Tkinter is an open source, portable graphical user interface (GUI) library designed for use in Python scripts.

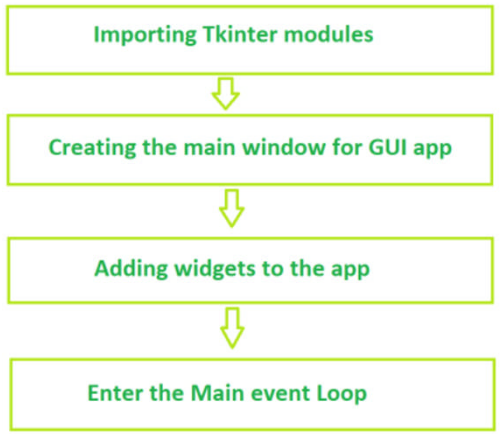
Advantages of Tkinter

- Tkinter is easy and fast to implement as compared to any other GUI toolkit.
- Tkinter is more flexible and stable.
- Tkinter is really easy to understand and master

Disadvantages of using Tkinter

- Tkinter does not include advanced widgets
- Sometime, it is hard to debug in Tkinter.

Fundamental Structure of Tkinter Program

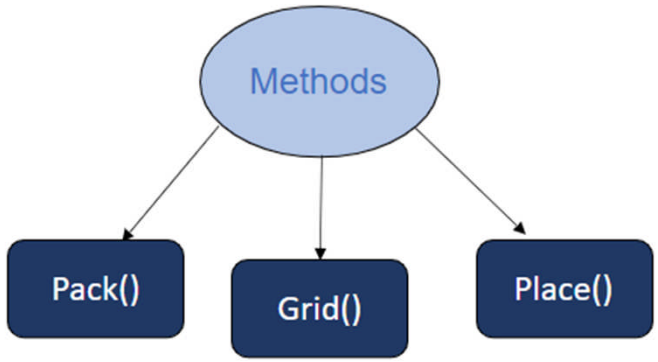


Basic Tkinter Widgets

- Label
- Button
- Canvas
- ComboBox
- CheckButton
- RediButton
- Frame
- Message
- Scale

Python Tkinter Geometry Manager

In order to **organize or arrange or place all the widgets** in the parent window, Tkinter provides us the **geometric configuration** of the widgets.



1. Tkinter pack() Geometry Manager

The pack() method mainly uses a packing algorithm to place widgets in a Frame or window in a specified order.

This method is mainly used to **organize the widgets in a block**.

Packing Algorithm:

The steps of Packing algorithm are as follows:

Firstly, this algorithm will **compute a rectangular area** known as a **Parcel** which is tall (or wide) enough to hold the widget and then it will fill the remaining width (or height) in the window with **blank space**.

It will **center the widget** until any different location is specified.
This method is powerful, but it is difficult to visualize.
Here is the **syntax** for using pack() function:

```
widget.pack(options)
```

Tkinter grid() Geometry Manager

The most used geometry manager is grid() because it provides all the power of pack() function but in an easier and maintainable way.

- You can **easily specify the location of a widget** just by calling grid() function and passing the **row** and **column indices** to the row and column keyword arguments, respectively.

Here is the **syntax** of the grid() function:

```
widget.grid(options)
```

Tkinter place() Geometry Manager

The place() Geometry Manager organizes the widgets to **place them in a specific position** as directed by the programmer.

- This method basically **organizes the widget** in accordance with its **x and y coordinates**. Both x and y coordinates are in **pixels**.
- Thus, the origin (where x and y are both 0) is the **top-left corner** of the Frame or the window.

Here is the **syntax** of the place() method:

```
widget.place(options)
```

Working with Image in Tkinter

Python Tkinter Image

Python Tkinter has the method PhotoImage which allows **reading images in Python Tkinter**.

There are ways of adding images on Python Tkinter.

- **PhotoImage** **Pillow Module**

How to use Pillow with Tkinter?

Tkinter relies on Pillow for working with images. Pillow is a fork of the Python Imaging Library and can be imported in a Python console as *PIL*.

To check if Pillow is already installed, enter the following command in a Python console:

help('PIL')

If Pillow is not installed, you can install it with:

python3 -m pip install pillow

edunet foundation

How to manage images with PIL and Tkinter?

To import *ImageTk* and *Image* in a Python console, enter:

```
from PIL import ImageTk, Image
```

An image can be opened with the following code snippet:

```
image1 = Image.open("<path/image_name>")
```

The *resize()* option can be used to set an image's height and width.

```
image1 = img.resize((50, 50), Image.ANTIALIAS)
```

Introduction to Tkinter Themes & Styles

In Tkinter, a theme determines the “look & feel” of all the widgets. It’s a collection of styles for all the ttk widgets.

A style specifies the appearance of a widget class e.g., a Button. Each theme comes with a set of styles. It’s possible to change the appearance of widgets by:

- Modifying the built-in styles
- Creating new styles

First, create a new instance of the ttk.Style class:

```
style = ttk.Style(root)
```

Second, get the available themes by calling the theme_names() method:

```
style.theme_names()
```

To get the current theme, you use the theme_use() method:

```
current_theme = style.theme_use()
```

To change the current theme to a new one, you pass the new theme name to the theme_use() method:

```
style.theme_use(theme_name)
```

Tkinter Example

Working with Iris Dataset

Importing Packages

In [21]:
```
import tkinter as tk
from tkinter import ttk
import pickle
import joblib
import numpy as np
import seaborn as sns
from sklearn.utils import shuffle
from sklearn.linear_model import LogisticRegression
```

Basics of tkinter

Creating a Window

In [22]:
```
window=tk.Tk()
tk.mainloop()
```

Creating Label widget in window

In [23]:
```
window=tk.Tk()
label1=tk.Label(window,text="Hello")
label1.pack()
tk.mainloop()
```

Creating Button widget in window

In [25]:
```
window=tk.Tk()
tk.Button(window,text="click",command= lambda : print("Clicked")).pack()
tk.mainloop()
```

```
Clicked
```

Working on Iris Dataset

In [26]:
```
# Importing Dataset and printing head
iris=sns.load_dataset("iris")
iris.head()
```

Out[26]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [27]:
```
# unique species
iris.species.unique()
```

Out[27]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

```
num=[]
for i in iris.species:
    if i == "setosa":
        num.append(0)
    elif i=='versicolor':
        num.append(1)
    elif i=='virginica':
        num.append(2)
    else:
        pass
iris['species_num']=num
iris.head()
```

Out[28]:

	sepal_length	sepal_width	petal_length	petal_width	species	species_num
0	5.1	3.5	1.4	0.2	setosa	0
1	4.9	3.0	1.4	0.2	setosa	0
2	4.7	3.2	1.3	0.2	setosa	0
3	4.6	3.1	1.5	0.2	setosa	0
4	5.0	3.6	1.4	0.2	setosa	0

Out[29]: array([0, 1, 2], dtype=int64)

In [30]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
 5   species_num   150 non-null    int64
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [31]:
```
# dropping the species name column
iris.drop('species',axis=1,inplace=True)
iris.head()
```

Out[31]:

	sepal_length	sepal_width	petal_length	petal_width	species_num
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [32]:
```
# shuffling data
iris=shuffle(iris)
iris.reset_index(drop=True)
iris.head()
```

Out[32]:

	sepal_length	sepal_width	petal_length	petal_width	species_num
71	6.1	2.8	4.0	1.3	1
76	6.8	2.8	4.8	1.4	1
129	7.2	3.0	5.8	1.6	2
45	4.8	3.0	1.4	0.3	0
141	6.9	3.1	5.1	2.3	2

References

1. https://www.geeksforgeeks.org/python-gui-tkinter/#:~:text=Python%20offers%20multiple%20options%20for,to%20create%20the%20GUI%20applications.
2. https://medium.com/teamresellerclub/the-6-best-python-gui-frameworks-for-developers-7a3f1a41ac73
3. https://www.computerhope.com/jargon/g/gui.htm
4. https://www.cs.mcgill.ca/~hv/classes/MS/TkinterPres/#Overview
5. https://www.geeksforgeeks.org/introduction-to-tkinter/
6. https://www.studytonight.com/tkinter/python-tkinter-geometry-manager
7. https://www.activestate.com/resources/quick-reads/how-to-add-images-in-tkinter/