**Constants in Python¶**

In Python, we cannot declare a variable or value as constant in Python.

But we can indicate to programmers that a variable is a constant by using an all upper case variable name.

This is the common practice to create a constant in Python:

**MAX\_VALUE = 99**

**//////////////////////////////////////////////////////////////////////////////////////////////////**

**Kivy** python takes the same path, allowing developers to develop multi-platform applications for several operating systems and devices. Thus, applications developed using this library a similar across all the platforms on which they run.

One advantage of Kivy is providing a graphical user interface that the developer interacts with as he develops the application.

In this tutorial, the reader will learn to set up Kivy for development and creating a basic application using the same library. Then, we will build a graphical user interface application for a user to do quick basic age calculations.

The development of this project will enable the reader to understand the entry-level programming with the Kivy library, create a graphical user interface using components such as images, text, buttons, and labels.

This tutorial will also teach the user how to interact with various layout components.

Prerequisites

The reader should have a good understanding of the following in order to follow along with this tutorial.

* Python programming language
* Objects and Classes in Python

Project setup

It would be best if you had the Kivy library installed in your system to get started. Use the command below to install Kivy on your machine.

pip install kivy

Library import

Create a folder and call it the calculator app. This name is the application’s name. In the folder created, create a new file, call it main.py, and add the snippet below.

from kivy.app import App

from kivy.uix.gridlayout import GridLayout

from kivy.uix.label import Label

from kivy.uix.image import Image

from kivy.uix.button import Button

from kivy.uix.text input import TextInput

Coding the application widgets

The snippet above imports the component widgets we require to build the Age Calculatorapplication.

Next, create a window object for the application by initializing a class called AgeCalculator. Kivy Python is based on Classes and Objects, so if the reader is not well-equipped with the knowledge of Python classes, they can look at that before the tutorial.

class AgeCalculator(App):

def build(self):

self.window = GridLayout()

return self.window

Lastly, we specify the function to run as soon as the program executes the file.

if \_\_name\_\_ == "\_\_main\_\_":

AgeCalculator().run()

We need to style our application for a more initutive user interface and interaction. For instance, we will begin by setting the number of columns that the grid layout takes. Then, we will do the app’s further styling after we have implemented full functionality.

self.window.cols = 1

We will set the logo for our application using the snippet below. Again, save the application’s logo in the same directory as the main.py file, so the application will easily find it.

self.window.add\_widget(Image(source("logo\_image.png")))

Since this application aims to find the user’s age when they provide the date of birth, we need to provide an input where they insert the date of birth and give it a label stating “enter your date of birth”.

self.ageRequest = Label(text = "Enter your year of birth...")

self.window.add\_widget(self.ageRequest)

self.date = TextInput(multiline=False)

self.window.add\_widget(self.date)

Collecting user input

Let us set up an application to connect the user input, which is the date of birth.

self.date = TextInput()

Bind to calculate age function to the button

We need to bind the callback function that calculates the age to the button such that when the Calculate Age button is clicked, the getAge function is called to calculate the age from the input year of birth.

self.button = Button(text = "Calculate Age")

self.button.bind(on\_press = self.getAge)

self.window.add\_widget(self.button)

Calculate age function

This function takes the input year of birth and subtracts it from the current year. We use the DateTime Python module and extract the year part from it.

Additionally, we convert the user input year from a string to an integer to calculate a uniform data type. Lastly, we convert the result to a string and display it to the user.

def getAge(self, event):

today = datetime.today().year

dob = self.date.text

age = int(today) - int(dob)

self.ageRequest.text = "You are " + str(int(age)) + " years old"

Styling the application’s UI.

To style our application correctly, we can begin by adding margins to the application’s sides and bottom. Margin is the space around a given component in the application.

For our case, the margin will be the space around the container layout. Subsequently, we play around with all other components of the application using the snippet below.

Another advantage of the module is that the application developed automatically responds to screen size and gives a clear outline of the subjected screen size.

def build(self):

self.window = GridLayout()

self.window.cols = 1

self.window.size\_hint = (0.6, 0.7)

self.window.pos\_hint = {"center\_x": 0.5, "center\_y": 0.5 }

self.window.add\_widget(Image(source("logo\_image.png")))

self.ageRequest = Label(

text = "Enter your date of birth...",

font\_size = 50,

color = "#ffffff",

bold = True

)

self.window.add\_widget(self.ageRequest)

self.date = TextInput(

multiline=False,

padding\_y = (30, 30),

size\_hint = (1, 0.7),

font\_size = 30

)

self.window.add\_widget(self.date)

self.button = Button(

text = "Calculate Age",

size\_hint = (0.5, 0.5),

bold = True,

font\_size = 30

)

self.button.bind(on\_press = self.getAge)

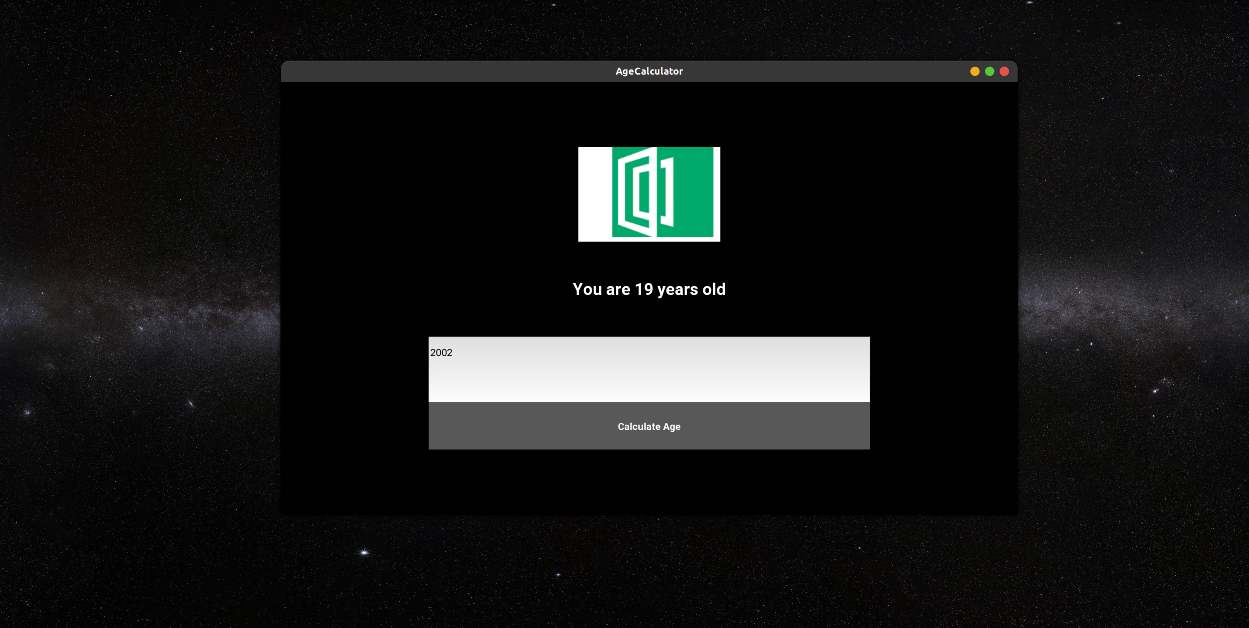
self.window.add\_widget(self.button)

return self.window

Running the application

For the application to run, open the terminal, then navigate to the project directory. Next, run the command below in the project directory.

python main.py



Conclusion

In this article, the reader learned the mobile application creation process using the Kivy python library. We developed an age calculator application using the multi-platform python module and tested our applications’ functionality.

We realized how every component of development using Kivy is based on understanding Python Classes and Objects.

Happy coding!