**Documentación Detallada de la Aplicación Kivy**

**Descripción General**

Esta aplicación es un recurso educativo interactivo desarrollado en Python utilizando el framework Kivy. Su objetivo es proporcionar una interfaz gráfica para aprender sobre conceptos matemáticos y de programación, incluyendo funciones, grafos, árboles, algoritmos y conjuntos. La aplicación está estructurada en varias pantallas, cada una dedicada a un tema específico.

La aplicación se compone de dos archivos principales:

1. **Código Python**: Define la lógica de la aplicación, incluyendo la gestión de pantallas y la interacción del usuario.
2. **Código Kivy (KV)**: Define la interfaz gráfica de la aplicación, estructurando la disposición de los elementos visuales y sus propiedades.

**Funcionamiento del Framework Kivy**

Kivy es un framework de Python para el desarrollo de aplicaciones multitáctiles y de escritorio. Permite crear interfaces gráficas de usuario (GUI) de manera sencilla y rápida. Kivy utiliza un enfoque basado en eventos, lo que significa que la aplicación responde a las acciones del usuario (como clics de botones o entradas de texto) mediante la invocación de métodos específicos.

**Colaboración entre Python y KV**

El código Python y el código KV trabajan juntos para crear una experiencia de usuario fluida. El código Python maneja la lógica de la aplicación, mientras que el código KV se encarga de la presentación visual. Aquí se detalla cómo se complementan:

1. **Definición de Pantallas**: Cada pantalla en el código KV se corresponde con una clase en el código Python. Por ejemplo, MainScreen en Python tiene su representación en KV.
2. **Interacción de Eventos**: Los eventos en el código KV (como on\_press o on\_release) están vinculados a métodos en el código Python. Por ejemplo, el botón "Entrar" en MainScreen llama al método go\_to\_topics en Python.
3. **Actualización de Contenido**: El código Python puede actualizar el contenido de los elementos visuales definidos en KV utilizando sus IDs. Por ejemplo, el método update\_title en TopicsScreen actualiza el texto de title\_label.
4. **Gestión de Datos**: El código Python maneja la lógica de negocio, como cálculos y manipulación de datos, mientras que el código KV se encarga de mostrar esos datos al usuario.

**Documentación del Código Python**

**Importaciones**

El código comienza con la importación de las bibliotecas necesarias:

python

import kivy

from kivy.app import App

from kivy.uix.screenmanager import ScreenManager, Screen

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.label import Label

from kivy.uix.textinput import TextInput

from kivy.uix.button import Button

from kivy.uix.scrollview import ScrollView

from kivy.uix.gridlayout import GridLayout

from kivy.uix.widget import Widget

from kivy.graphics import Color, Rectangle, Ellipse

from kivy.clock import Clock

from kivy.uix.image import Image

from matplotlib import \*

from matplotlib\_venn import venn2

from kivy.properties import StringProperty, NumericProperty, ReferenceListProperty, ObjectProperty, ListProperty, BooleanProperty

import random

from kivy.core.window import Window

from kivy.uix.slider import Slider

import matplotlib.pyplot as plt

from kivy.vector import Vector

from kivy.config import Config

from kivy.uix.videoplayer import VideoPlayer

from kivy.lang import Builder

from kivymd.app import MDApp

from kivy.uix.anchorlayout import AnchorLayout

from kivy.uix.stacklayout import StackLayout

**Clases y Métodos**

1. **MainScreen**
   * **Método go\_to\_topics**: Este método se activa cuando el usuario presiona el botón "Entrar". Obtiene el nombre ingresado en el campo de texto y cambia a la pantalla de temas, actualizando el título con el nombre proporcionado.
2. **TopicsScreen**
   * **Método update\_title**: Este método actualiza el título de la pantalla de temas con el nombre del usuario.
   * **Método show\_topic**: Este método imprime en la consola el tema seleccionado por el usuario.
   * **Método exit\_app**: Este método cierra la aplicación.
3. **FuncionesScreen**
   * **Método on\_enter**: Este método se ejecuta cuando se entra en la pantalla de funciones. Actualiza las etiquetas con información sobre funciones y relaciones.
4. **GrafosScreen**
   * **Método on\_enter**: Similar al anterior, este método actualiza las etiquetas con información sobre grafos.
5. **ArbolesScreen**
   * **Método on\_enter**: Actualiza las etiquetas con información sobre árboles.
6. **AlgoritmosScreen**
   * **Método on\_enter**: Este método se ejecuta al entrar en la pantalla de algoritmos, actualizando las etiquetas con información sobre algoritmos y sus tipos.
7. **BolitaScreen**
   * Contiene una clase interna CanvasExample5 que maneja la animación de una bolita que rebota en la pantalla. Utiliza el método update para cambiar la posición de la bolita y el método change\_color para cambiar su color al rebotar.
8. **ConjuntosScreen**
   * **Método on\_enter**: Este método se ejecuta al entrar en la pantalla de conjuntos, mostrando información sobre diagramas de Venn y permitiendo al usuario realizar cálculos de conjuntos. Incluye métodos para calcular la unión, intersección, complemento y diferencia simétrica de conjuntos.

**Clase Principal de la Aplicación**

* **PapuApp**: Esta clase inicializa el ScreenManager y agrega todas las pantallas a la aplicación. El método build configura la estructura de la aplicación y establece la pantalla inicial.

**Documentación del Código Kivy (KV)**

**Estructura del Código KV**

1. **ScreenManager**
   * Define el ScreenManager, que es responsable de gestionar las diferentes pantallas de la aplicación. Cada pantalla se agrega al ScreenManager con un nombre único.

kv

<ScreenManager>:

MainScreen:

name: 'Main'

TopicsScreen:

name: 'Topics'

ConjuntosScreen:

name: 'ConjuntoS'

FuncionesScreen:

name: 'FuncioneS'

GrafosScreen:

name: 'GrafoS'

ArbolesScreen:

name: 'ArboleS'

AlgoritmosScreen:

name: 'AlgoritmoS'

1. **MainScreen**
   * Contiene un BoxLayout que organiza los elementos verticalmente. Incluye un Label para mostrar un saludo, un TextInput para que el usuario ingrese su nombre y un Button que llama al método go\_to\_topics en el código Python.

kv

<MainScreen>:

canvas.before:

Rectangle:

source: 'Bienvenida.jpg'

pos: self.pos

size: self.size

BoxLayout:

orientation: 'vertical'

Label:

text: 'Hola Precioso UwU'

size\_hint\_y: None

height: 60

font\_size: 60

halign: 'center'

valign: 'top'

text\_size: self.size

TextInput:

id: name\_input

hint\_text: 'Ingrese su nombre'

multiline: False

size\_hint\_y: None

height: 40

Button:

text: 'Entrar'

size\_hint\_y: None

height: 40

on\_press: root.go\_to\_topics()

background\_color: (1, 1, 1, 1)

1. **TopicsScreen**
   * Presenta una lista de temas en un ScrollView que permite al usuario desplazarse. Cada Button en el GridLayout está vinculado a un método en el código Python que cambia a la pantalla correspondiente cuando se presiona.

kv

<TopicsScreen>:

BoxLayout:

orientation: 'vertical'

Image:

source: 'Portada.jpg'

allow\_stretch: True

keep\_ratio: False

Label:

id: title\_label

text: 'Temas'

font\_size: '30sp'

halign: 'center'

size\_hint\_y: None

height: 50

ScrollView:

GridLayout:

id: topics\_grid

cols: 1

size\_hint\_y: None

height: self.minimum\_height

Button:

text: 'Conjuntos'

size\_hint\_y: None

height: 40

on\_release: root.manager.current = 'ConjuntoS'

Button:

text: 'Funciones y Relaciones'

size\_hint\_y: None

height: 40

on\_press: root.manager.current = 'FuncioneS'

Button:

text: 'Grafos'

size\_hint\_y: None

height: 40

on\_press: root.manager.current = 'GrafoS'

Button:

text: 'Árboles'

size\_hint\_y: None

height: 40

on\_press: root.manager.current = 'ArboleS'

Button:

text: 'Algoritmos'

size\_hint\_y: None

height: 40

on\_press : root.manager.current = 'AlgoritmoS'

Button:

text: 'WoW'

size\_hint\_y: None

height: 40

on\_press: root.manager.current = 'Bolita'

Button:

text: 'Salir'

size\_hint\_y: None

height: 40

on\_press: root.exit\_app()

1. **FuncionesScreen, GrafosScreen, ArbolesScreen, AlgoritmosScreen, BolitaScreen**
   * Cada una de estas pantallas sigue una estructura similar, donde se utilizan Labels para mostrar información y VideoPlayer para reproducir videos relevantes. Los métodos on\_enter en el código Python actualizan el contenido de las etiquetas cuando se accede a cada pantalla.

kv

<FuncionesScreen>:

name: 'Funciones y Relaciones'

BoxLayout:

orientation: 'vertical'

canvas.before:

Rectangle:

pos: self.pos

size: self.size

source: 'FONDO2.png'

BoxLayout:

orientation: 'vertical'

ScrollView:

size\_hint: (1, 1)

BoxLayout:

orientation: 'vertical'

size\_hint\_y: None

height: self.minimum\_height

padding: 10

spacing: 10

Label:

text: 'Funciones y Relaciones'

size\_hint\_y: None

height: 60

font\_size: 60

halign: 'center'

valign: 'middle'

text\_size: self.size

Label:

id: funciones1\_label

size\_hint\_y: None

height: self.texture\_size[1] + 20

text\_size: self.width, None

halign: 'left'

valign: 'top'

padding: [10, 10]

Image:

source: 'Funciones.png'

size\_hint\_x: 1

size\_hint\_y: None

height: 300

allow\_stretch: True

keep\_ratio: True

Button:

text: 'Menu'

size\_hint: None, None

size: 100, 20

pos\_hint: {'center\_x': 0.5, 'center\_y': 0.1}

on\_release: root.manager.current = 'Topics'

1. **ConjuntosScreen**
   * Esta pantalla permite al usuario realizar cálculos de conjuntos. Los TextInput permiten ingresar los elementos de los conjuntos, y los botones llaman a los métodos correspondientes en el código Python para calcular la unión, intersección, etc.

kv

<ConjuntosScreen>:

name: 'Conjuntos'

BoxLayout:

orientation: 'vertical'

canvas.before:

Rectangle:

pos: self.pos

size: self.size

source: 'FONDO1.jpg'

BoxLayout:

orientation: 'vertical'

ScrollView:

size\_hint: (1, 1)

BoxLayout:

orientation: 'vertical'

size\_hint\_y: None

height: self.minimum\_height

padding: 10

spacing: 10

Label:

text: 'Conjuntos'

size\_hint\_y: None

height: 60

font\_size: 60

halign: 'center'

valign: 'middle'

text\_size: self.size

Label:

text: "Universo (separar por comas):"

size\_hint\_y: None

height: 30

font\_size: 25

TextInput:

id: input\_universo

size\_hint\_y: None

height: 40

Button:

text: 'Calcular Unión'

on\_release: root.calcular\_union()

Button:

text: 'Menu'

size\_hint: None, None

size: 100, 20

pos\_hint: {'center\_x': 0.5, 'center\_y': 0.1}

on\_release: root.manager.current = 'Topics'

**Conclusión**

La interacción entre el código Python y el código Kivy (KV) es fundamental para el funcionamiento de la aplicación. El código Python maneja la lógica y la interacción del usuario, mientras que el código KV se encarga de la presentación visual y la disposición de los elementos en la interfaz. Cada pantalla en el código KV se corresponde con una clase en el código Python, y los métodos definidos en las clases Python son invocados a través de eventos en el código KV, creando una experiencia de usuario fluida y coherente.