Code Framework 代码框架

```python
class Application(Frame):
    def __init__(self,master=None):
        super().__init__(master)
        self.master = master
        self.pack()

        self.CreateEntry()
        self.CreatePicture()
        self.CreateBotton()
        # other elements

    def CreateEntry(self):
        pass

    def CreatePicture(self):
        pass

    def CreateBotton(self):
        # 创建按钮对象
        self.B = Button(self, text="Button_1", command=Bt_function)
        # 定位
        self.B.grid(row=0, column=0, rowspan=1, columnspan=1)

    def Bt_function(self):
        pass


if __name__ == '__main__':
    root = Tk()
    root.geometry(f'{300}x{900}')
    app = Application(master=root)
    root.mainloop()
```

Init

```python
import tkinter as tk

main = tk.Tk()
main.title("Title")     # 设置窗口标题
main.config(bg="#fff") # 设置背景色为白色
```

Frame

添加一块框架区域，用于放置各种元素

```
top_frame = tk.Frame(main)  # 创建一个框架区域
top_frame.pack(side=tk.TOP) # 设置框架在顶部
```

初始化完框架区域后，需要进步设置框架区域的大小和颜色，并放置元素。例如，在蓝色背景上放置一个 logo:

```
logo = PhotoImage(file="logo.png") # 读取图片
tk.Label(top_frame, image=logo, bg="#1f77b4", height=85, width=1300).pack() # 在框架中添加元素
```

Canvas

添加一块画布区域

```
canvas = tk.Canvas(main, with=1300, height=350, bg="white")
canvas.pack(side=tk.TOP) # 设置画布在顶部（此时如果前面已经有一个frame.pack(side=tk.TOP)，那么画布会跟在框架后面）
```

Label

显示文字标签

```
lb = Label(tk, text="Text Content")
```

显示图片标签

```
pic = PhotoImage(file="pic.png")
lb_pic = Label(tk, image=pic)
```

Entry

输入框

```
# 定义一个字符串变量，用于接收输入
v = StringVar()
# 定义一个长度为 10 个字符的输入框
ety = Entry(tk, textvariable=v, width=10)
```

Botton

创建按钮，并绑定触发事件

```
B = Button(tk, text="Botton", command=Bt_function)

def Bt_function:
    pass
```

Toplevel

创建一个新的窗口，常绑定到按钮上，从而实现 "点击按钮后弹出新窗口" 的功能

```
def new_window():
    toplevel = Toplevel()
    toplevel.title("New Window")
    toplevel.geometry(f'{300}x{300}')
    toplevel.config(bg="#fff")

    # 设置关闭按钮的功能（当点击窗口的关闭键后，销毁窗口）
    def on_close():
        toplevel.destroy()
    toplevel.protocol("WM_DELETE_WINDOW", on_close)

root = Tk()
button = Button(root, text="New Window", command=new_window)
button.pack()
root.mainloop()
```

当 Toplevel 结合 simpy 来使用时，我们往往会希望在打开一个窗口时开始一个进程 `env.process()`，并且在关闭这个窗口时结束这个进程，从而避免资源浪费

```
class tkUpdate:
    def __init__(self, env, canvas):
        self.env = env
        self.canvas = canvas

    def run(self):
        while True:
            if self.canvas is not None and self.canvas.winfo_exists():
                # do something change on canvas
             env.timeout(1)

def create_toplevel():
    toplevel = Toplevel()
    toplevel.title("New Window")
    canvas = Canvas(toplevel, width=700, height=240)

    # Start the SimPy process when the Toplevel window is opened
    tkU = tkUpdate(env, canvas)
    process = env.process(tkU.run())

    # Bind the <WM_DELETE_WINDOW> event to a function that stops the SimPy process
    def on_close():
        process.interrupt((simpy.events.Event("Window closed")))
        toplevel.destroy()
    toplevel.protocol("WM_DELETE_WINDOW", on_close)

env = simpy.Environment()
root = tk.Tk()
button = tk.Button(root, text="Create Toplevel", command=create_toplevel)
button.pack()
root.mainloop()
```