Борисов Илья Анатольевич

Лабораторная работа 9

Вариант 2

Знакомство с фреймворком Kivy. Разработка программ, взаимодействующих с виджетом Image

Цель работы: научиться создавать оконные приложения с помощью библиотек Python

Задания

***Задание 1***

Создать оконное приложение, содержащие следующие виджеты:

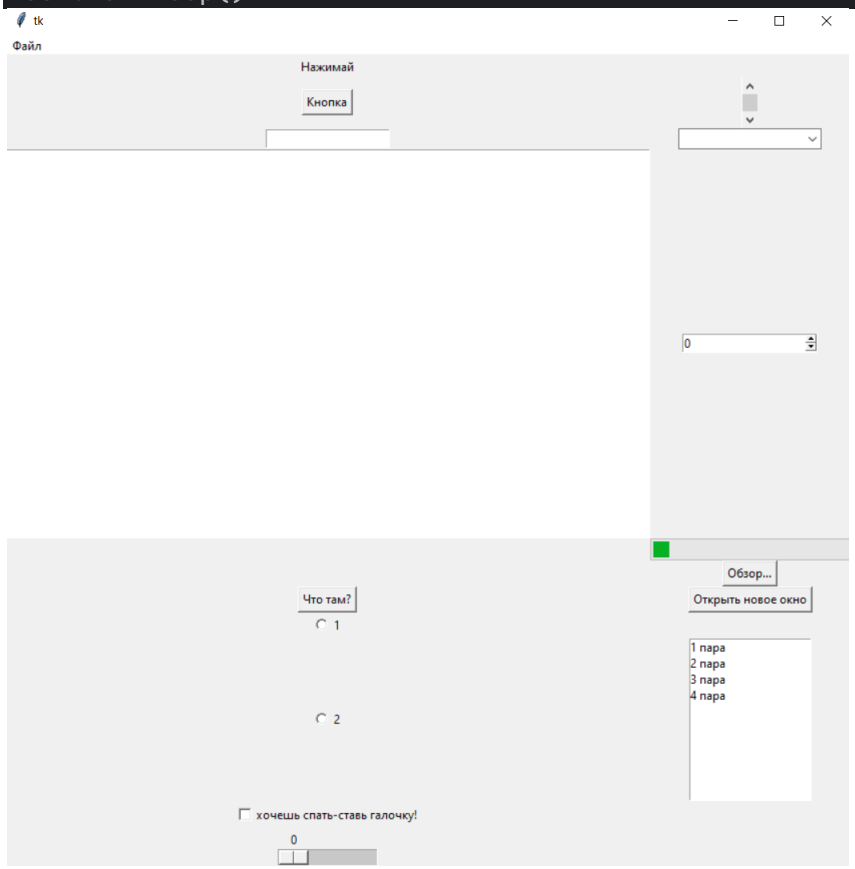
| |
|---|
| Надпись Label |
| Кнопка Button |
| Поле ввода Entry |
| Многострочное поле ввода Text |
| Рамка Frame |
| Рамка с надписью LabelFrame |
| Всплывающее окно messagebox |
| Переключатель Radiobutton |
| Независимый переключатель Checkbutton |
| Списки Listbox |
| Шкала (выбор какого-то значения из определенного диапазона) Scale |
| Прокрутка Scrollbar |
| Меню Menu |
| Раскрывающийся список Combobox |
| Поле ввода со стрелками приращения и уменьшения Spinbox |
| Полоса прогресса Progressbar |
| Диалоговые окна открытия и сохранения файлов filedialog |
| Дочерние окна Toplevel |

При создании приложения использовать разные менеджеры геометрии (`.pack()`, `.place()`, `.grid()`)

```python
import tkinter as tk
from tkinter import ttk, messagebox, filedialog

# Создание основного окна
root = tk.Tk()

# Надпись Label
label = tk.Label(root, text="Нажимай")
label.grid(row=0, column=0)

# Кнопка Button
def on_button_click():
    print("Button clicked")
button = tk.Button(root, text="Кнопка", command=on_button_click)
button.grid(row=1, column=0)

# Поле ввода Entry
entry = tk.Entry(root)
entry.grid(row=2, column=0)
```

```
# Многострочное поле ввода Text
text = tk.Text(root)
text.grid(row=3, column=0)

# Рамка Frame
frame = tk.Frame(root)
frame.grid(row=4, column=0)

# Рамка с надписью LabelFrame
labelframe = tk.LabelFrame(root, text="Рамка")
labelframe.grid(row=5, column=0)

# Всплывающее окно messagebox
def show_messagebox():
    messagebox.showinfo("Напоминание!", "Хорошего дня!!!")
messagebox_button = tk.Button(root, text="Что там?", command=show_messagebox)
messagebox_button.grid(row=6, column=0)

# Переключатель Radiobutton
v = tk.IntVar()
radiobutton1 = tk.Radiobutton(root, text="1", variable=v, value=1)
radiobutton1.grid(row=7, column=0)
radiobutton2 = tk.Radiobutton(root, text="2", variable=v, value=2)
radiobutton2.grid(row=8, column=0)

# Независимый переключатель Checkbutton
checkbutton = tk.Checkbutton(root, text="хочешь спать-ставь галочку!")
checkbutton.grid(row=9, column=0)

# Шкала Scale
scale = tk.Scale(root, from_=0, to=100, orient=tk.HORIZONTAL)
scale.grid(row=11, column=0)

# Прокрутка Scrollbar
scrollbar = tk.Scrollbar(root)
scrollbar.grid(row=1, column=1, sticky='ns')

# Меню Menu
menu = tk.Menu(root)
root.config(menu=menu)
filemenu = tk.Menu(menu)
menu.add_cascade(label="Файл", menu=filemenu)
filemenu.add_command(label="Новый")
filemenu.add_command(label="Открыть...")
filemenu.add_command(label="Сохранить в файл..")
filemenu.add_command(label="Выход")

# Раскрывающийся список Combobox
combobox = ttk.Combobox(root, values=["Евгений", "Илья", "Максим", "Леша"])
combobox.grid(row=2, column=1)

# Поле ввода со стрелками приращения и уменьшения Spinbox
spinbox = tk.Spinbox(root, from_=0, to=10)
```

```
spinbox.grid(row=3, column=1)

# Полоса прогресса Progressbar
progressbar = ttk.Progressbar(root, length=200, mode='indeterminate')
progressbar.grid(row=4, column=1)

# Диалоговые окна открытия и сохранения файлов filedialog
def open_file():
    filename = filedialog.askopenfilename()
open_file_button = tk.Button(root, text="Обзор...", command=open_file)
open_file_button.grid(row=5, column=1)

# Дочерние окна Toplevel
def open_new_window():
    new_window = tk.Toplevel(root)
    new_window.title("Новое окно")
new_window_button = tk.Button(root, text="Открыть новое окно",
command=open_new_window)
new_window_button.grid(row=6, column=1)

# Списки Listbox
listbox = tk.Listbox(root)
listbox.insert(1, "1 пара")
listbox.insert(2, "2 пара")
listbox.insert(3, "3 пара")
listbox.insert(4, "4 пара")
listbox.grid(row=8, column=1)

root.mainloop()
```



Борисов Илья Анатольевич

Задание 2

Создать оконное приложение, содержащие следующие виджеты:
При нажатии на кнопку идет загрузка Progressbar и появляется текст «Уважаемый ФИО, идет обработка ваших данных»

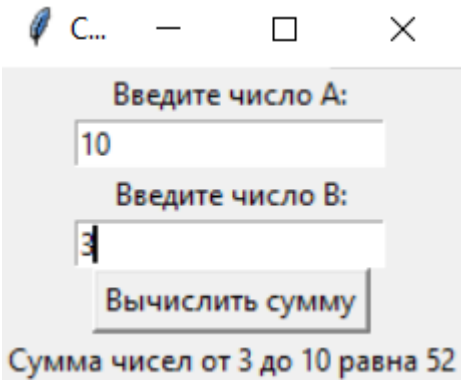
```
import tkinter as tk
from tkinter import ttk

def on_button_click():
    progressbar.start()
    label.config(text=f"Уважаемая {entry.get()}, идет обработка ваших данных")

root = tk.Tk()
root.title("Оконное приложение")
root.configure(bg='orange')

tk.Label(root, text="Ваше ФИО:").grid(row=0, column=0)
entry = tk.Entry(root)
entry.grid(row=0, column=1)

tk.Label(root, text="Увлечения:").grid(row=1, column=0)
checkbutton = tk.Checkbutton(root)
checkbutton.grid(row=1, column=1)

progressbar = ttk.Progressbar(root, length=200, mode='indeterminate')
progressbar.grid(row=2, column=0, columnspan=2)

button = tk.Button(root, text="Кнопка", command=on_button_click)
button.grid(row=3, column=0, columnspan=2)

label = tk.Label(root, text="")
label.grid(row=4, column=0, columnspan=2)

root.mainloop()
```

Задание 3

B5-6: Пользователь вводит два числа A и B. Функция суммирует все целые числа от A до B включительно. Если пользователь задаст первое число большее чем второе, поменяйте их местами и посчитайте.

```
import tkinter as tk

def calculate_sum():
    try:
        a = int(entry_a.get())
        b = int(entry_b.get())
        if a > b:
            a, b = b, a  # Поменяем местами, если a > b
        total_sum = sum(range(a, b + 1))
        result_label.config(text=f"Сумма чисел от {a} до {b} равна {total_sum}")
    except ValueError:
        result_label.config(text="Пожалуйста, введите целые числа")

# Создаем графический интерфейс
root = tk.Tk()
root.title("Сумма чисел")
```

```
label_a = tk.Label(root, text="Введите число A:")
entry_a = tk.Entry(root)
label_b = tk.Label(root, text="Введите число B:")
entry_b = tk.Entry(root)
calculate_button = tk.Button(root, text="Вычислить сумму", command=calculate_sum)
result_label = tk.Label(root, text="")

label_a.pack()
entry_a.pack()
label_b.pack()
entry_b.pack()
calculate_button.pack()
result_label.pack()

root.mainloop()
```



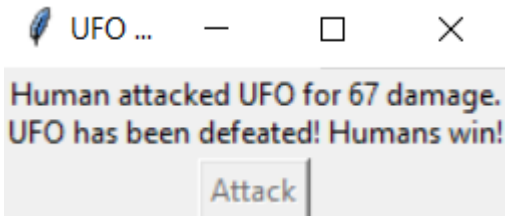

***Задание 4***

Реализовать игру, используя алгоритм из ЛР «Объектно-ориентированное программирование»

***B1-B6:*** Напишите игру по следующему описанию. Есть класс «НЛО» и класс «ЛюдиХ». У НЛО один объект, имеющий здоровье 1000hp. Создается от 1 до 5 людей Х (случайное число), имеющих здоровье 100 hp каждый. В случайном порядке НЛО и ЛюдиХ бьют друг друга. Тот, кто бьет, здоровья не теряет. У того, кого бьют, оно уменьшается на N очков от одного удара (N случайное число). После каждого удара надо выводить сообщение, кто кого атаковал, и сколько у противников осталось здоровья. Как только у НЛО заканчивается ресурс здоровья или погибают все ЛюдиХ, программа завершается сообщением о том, кто одержал победу.

```
import tkinter as tk
import random

class UFOGame:
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("UFO vs. Humans")
        self.ufo_health = 1000
        self.num_humans = random.randint(1, 5)
        self.human_health = [100] * self.num_humans

        self.label = tk.Label(self.root, text="UFO vs. Humans")
        self.label.pack()
```

```
        self.attack_button = tk.Button(self.root, text="Attack",
command=self.attack)
        self.attack_button.pack()

    def attack(self):
        attacker = random.choice(["UFO", "Human"])
        if attacker == "UFO":
            target = random.randint(0, self.num_humans - 1)
            damage = random.randint(1, 50)
            self.human_health[target] -= damage
            message = f"UFO attacked Human {target + 1} for {damage} damage."
        else:
            damage = random.randint(1, 100)
            self.ufo_health -= damage
            message = f"Human attacked UFO for {damage} damage."

        if self.ufo_health <= 0 or all(hp <= 0 for hp in self.human_health):
            if self.ufo_health <= 0:
                message += "\nUFO has been defeated! Humans win!"
            else:
                message += "\nAll humans have been defeated! UFO wins!"
            self.attack_button.config(state=tk.DISABLED)

        self.label.config(text=message)

    def run(self):
        self.root.mainloop()

if __name__ == "__main__":
    game = UFOGame()
    game.run()
```




Контрольные вопросы:

1. Порядок выполнения программ в классическом программировании определяется последовательным выполнением инструкций в коде. Программа выполняет действия в том порядке, в котором они записаны. Однако, в событийном программировании, выполнение программы зависит от событий — действий пользователя (например, нажатие клавиши, клик мыши) или сообщений от других программ и потоков. В событийно-ориентированных программах обычно используется главный цикл, который выбирает и обрабатывает события.
2. Событие — это факт, который может произойти или не произойти в результате опыта. Например, появление герба при бросании монеты, попадание в цель при выстреле или обнаружение объекта радиолокационной станцией.

3. Алгоритм работы программы, управляемой событиями:

- Создайте главный цикл приложения.
- В этом цикле:
  - Выберите событие (например, нажатие клавиши или клик мыши).
  - Обработайте это событие, выполнив соответствующие действия.
  - Повторите, пока не завершится выполнение программы.

4. Известные графические библиотеки для Python:

- PyQt5: Построен на платформе Qt и поддерживает кроссплатформенную разработку. Позволяет создавать приложения для разных платформ, таких как Mac, Windows, Linux, iOS и Android.
- Tkinter: Простая библиотека с открытым исходным кодом, предустановленная в Python. Хороший выбор для начинающих и средних пользователей.
- Matplotlib: Удобно строит графики для всех приложений, используя свой API.
- Seaborn: Универсальная библиотека на основе Matplotlib, позволяющая сравнивать несколько переменных.
- Plotly: Позволяет создавать интерактивные графики с помощью JS.

5. Принципы создания обработчика события:

- Выберите элемент (например, кнопку) для привязки обработчика события.
- Зарегистрируйте обработчик события для выбранного элемента.
- Определите функцию-обработчик, которая будет выполняться при наступлении события.
- Реализуйте функционал обработчика события.