

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4.8
дисциплины «Объектно-ориентированное программирование»

Выполнил:
Магомедов Имран Борисович
3 курс, группа «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., доцент департамента
цифровых, робототехнических систем и
электроники

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

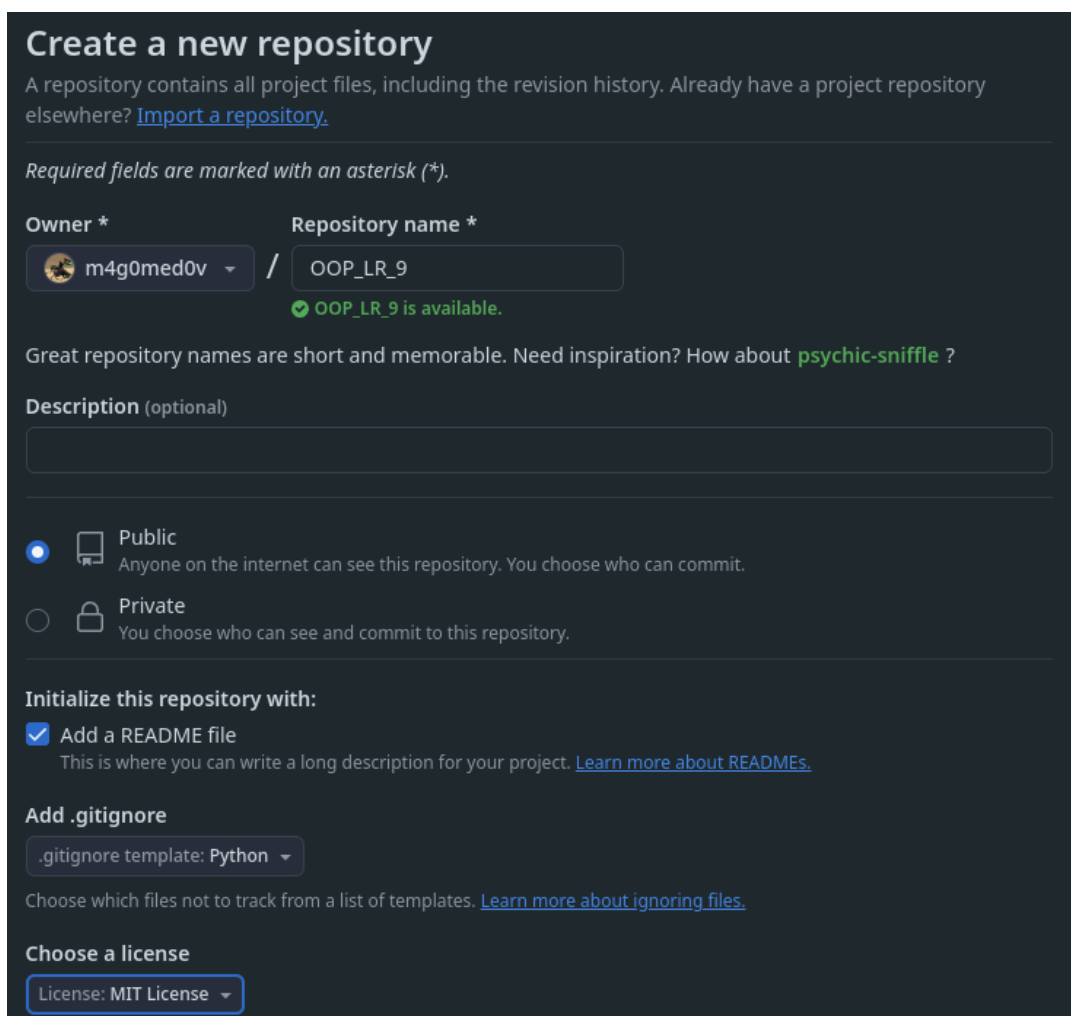
Тема: Обработка событий и рисование в Tkinter.

Цель работы: приобретение навыков улучшения графического интерфейса пользователя GUI с помощью обработки событий и рисования, реализованных в пакете Tkinter языка программирования Python версии 3.x.

Методика выполнения работы

[Ссылка на репозиторий](#)

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.



The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, it states 'Required fields are marked with an asterisk (*)'. The form has two main sections: 'Owner' and 'Repository name'. The 'Owner' is set to 'm4g0med0v' and the 'Repository name' is 'OOP_LR_9'. A green checkmark indicates that 'OOP_LR_9' is available. Below this, there is a suggestion for repository names: 'Great repository names are short and memorable. Need inspiration? How about **psychic-sniffle** ?'. The 'Description (optional)' field is empty. The 'Visibility' section has two options: 'Public' (selected) and 'Private'. The 'Initialize this repository with:' section has a checked box for 'Add a README file'. The 'Add .gitignore' section has a dropdown menu set to '.gitignore template: Python'. The 'Choose a license' section has a dropdown menu set to 'License: MIT License'.

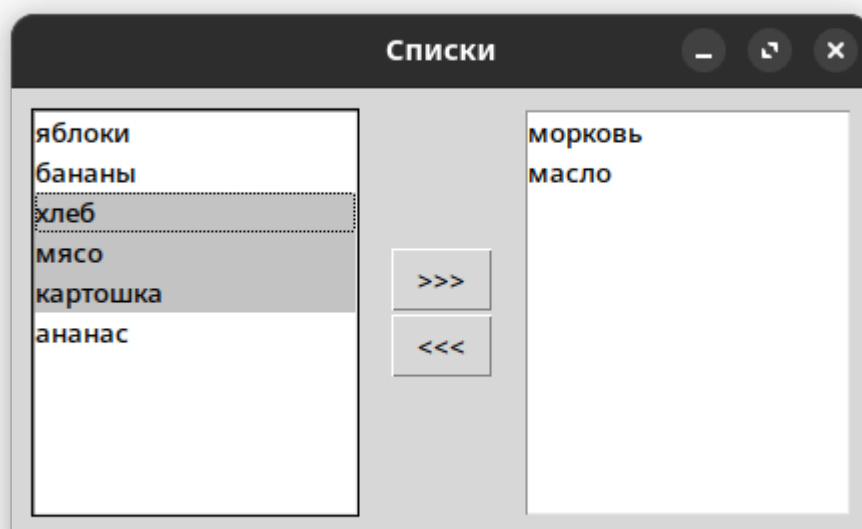
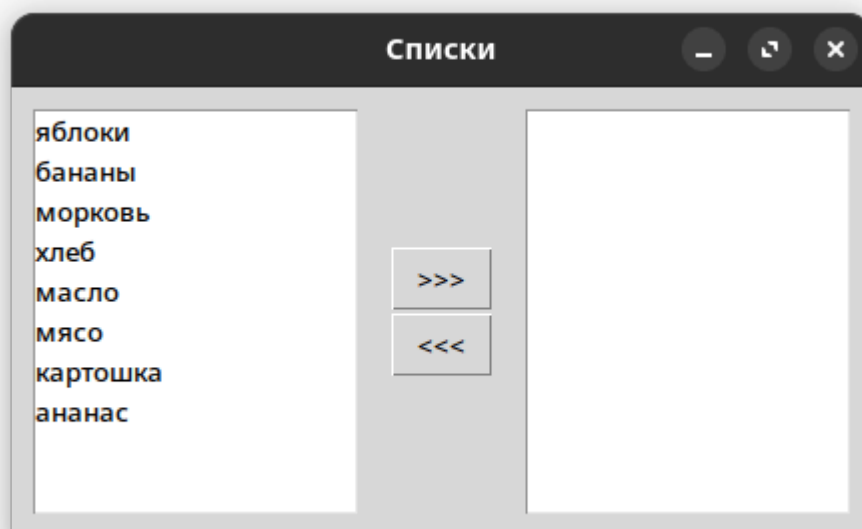
3. Выполните клонирование созданного репозитория.
4. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```
(.venv) → OOP_LR_9 (main) git branch develop *  
(.venv) → OOP_LR_9 (main) git switch develop *
```

5. Решите задачу: напишите программу, состоящую из двух списков Listbox . В первом будет, например, перечень товаров, заданный программно. Второй изначально пуст, пусть это будет перечень покупок. При клике на одну кнопку товар должен переходить из одного списка в другой. При клике на вторую кнопку – возвращаться (человек передумал покупать). Предусмотрите возможность множественного выбора элементов списка и их перемещения.



Решение:



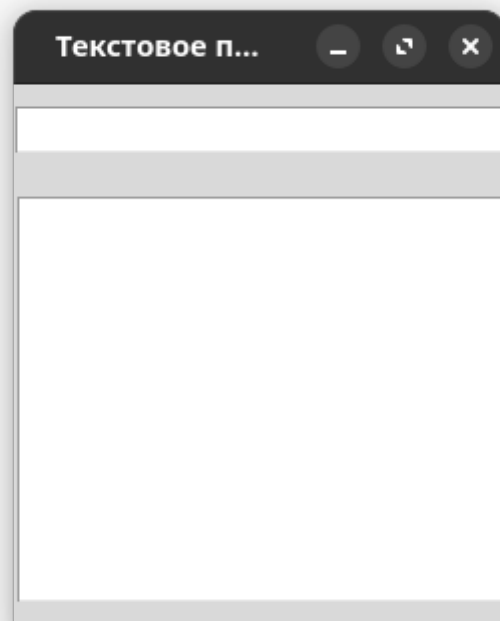
```
25 def move_to_list_2():
26     # Получаем индексы выбранных элементов в listbox_1
27     selected_items = list(listbox_1.curselection())
28
29     # Проверяем, выбраны ли элементы
30     if not selected_items:
31         messagebox.showinfo("Информация", "Выберите элементы для перемещения.")
32         return
33
34     # Перемещаем элементы из listbox_1 в listbox_2
35     for i in selected_items[::-1]:
36         listbox_2.insert(END, listbox_1.get(i))
37         listbox_1.delete(i)
38
39
40 def move_to_list_1():
41     # Получаем индексы выбранных элементов в listbox_2
42     selected_items = list(listbox_2.curselection())
43     print(selected_items)
44
45     # Проверяем, выбраны ли элементы
46     if not selected_items:
47         messagebox.showinfo("Информация", "Выберите элементы для перемещения.")
48         return
49
50     # Перемещаем элементы из listbox_2 в listbox_1
51     for i in selected_items[::-1]:
52         listbox_1.insert(END, listbox_2.get(i))
53         listbox_2.delete(i)
```

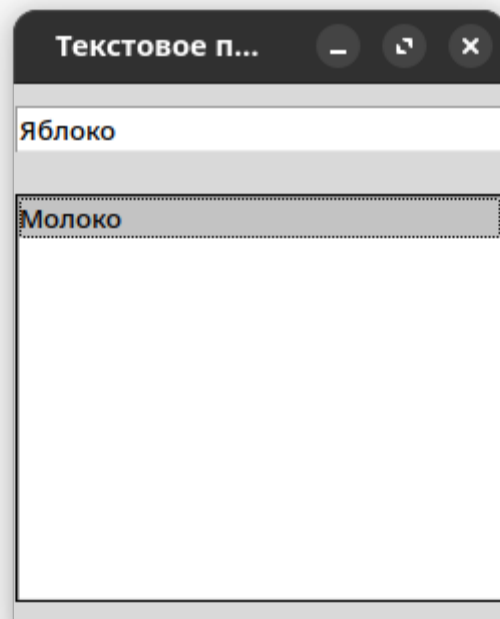
```
56 root = Tk()
57 root.title("Списки")
58
59 # Создаем фреймы для списков и кнопок
60 leftFrame = Frame(root)
61 leftFrame.grid(row=0, column=0)
62
63 midFrame = Frame(root)
64 midFrame.grid(row=0, column=1)
65
66 rightFrame = Frame(root)
67 rightFrame.grid(row=0, column=2)
68
69 # Создаем список товаров
70 listbox_1 = Listbox(leftFrame, selectmode=EXTENDED)
71 listbox_1.pack(padx=10, pady=10)
72
73 # Добавляем товары в список
74 items = [
75     "яблоки",
76     "бананы",
77     "морковь",
78     "хлеб",
79     "масло",
80     "мясо",
81     "картошка",
82     "ананас",
83 ]
84 for item in items:
85     listbox_1.insert(END, item)
86
87 # Создаем список покупок
88 listbox_2 = Listbox(rightFrame, selectmode=EXTENDED)
89 listbox_2.pack(padx=10, pady=10)
```

```
91 # Создаем кнопки для перемещения элементов
92 button_1 = Button(
93     midFrame,
94     text=" >>> ",
95     command=move_to_list_2,
96 )
97 button_1.pack(padx=5)
98
99 button_2 = Button(
100     midFrame,
101     text=" <<< ",
102     command=move_to_list_1,
103 )
104 button_2.pack(padx=5)
105
106
107 if __name__ == "__main__":
108     root.mainloop()
```

6. Решите задачу: напишите программу по следующему описанию. Нажатие Enter в однострочном текстовом поле приводит к перемещению текста из него в список (экземпляр Listbox). При двойном клике (<Double-Button-1>) по элементу-строке списка, она должна копироваться в текстовое поле.

Решение:





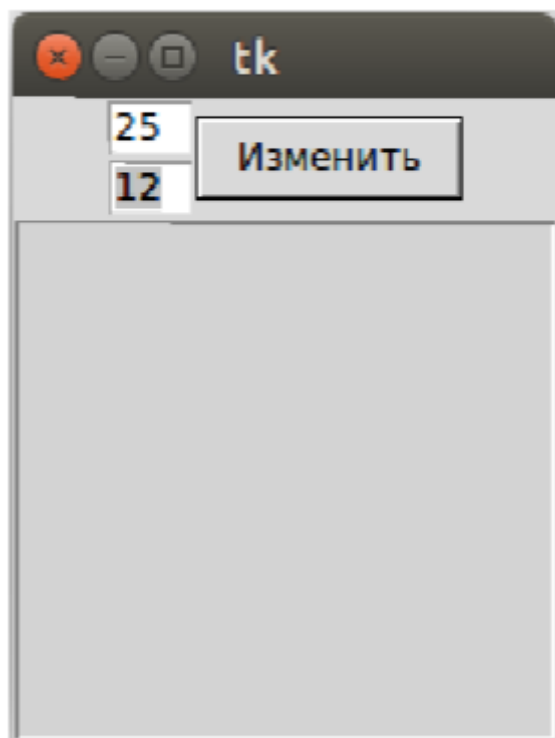

```

14 # Функция для добавления текста из текстового поля в список
15 def add_to_list(event):
16     text = entry.get()
17     if text.strip():
18         listbox.insert(END, text)
19         entry.delete(0, END)
20
21
22 # Функция для копирования выбранного элемента из списка в текстовое поле
23 def copy_to_entry(event):
24     selected_item_index = listbox.curselection()
25     if selected_item_index:
26         text = listbox.get(selected_item_index[0])
27         entry.delete(0, END)
28         entry.insert(0, text)
29
30
31 # Создаем главное окно приложения
32 root = Tk()
33 root.title("Текстовое поле и список")
34
35 # Создаем текстовое поле и.pack его в окно
36 entry = Entry(root, width=30)
37 entry.pack(pady=10)
38 # Привязываем нажатие Enter к функции add_to_list
39 entry.bind("<Return>", add_to_list)
40
41 # Создаем список и.pack его в окно
42 listbox = Listbox(root, width=30, height=10)
43 listbox.pack(pady=10)
44 # Привязываем двойной клик к функции copy_to_entry
45 listbox.bind("<Double-Button-1>", copy_to_entry)
46
47
48 # Запуск главного цикла приложения
49 if __name__ == "__main__":
50     root.mainloop()

```

7. Решите задачу: напишите программу по описанию. Размеры многострочного текстового поля определяются значениями, введенными в однострочные текстовые поля. Изменение размера происходит при нажатии мышью на кнопку, а также при нажатии клавиши Enter. Цвет фона экземпляра Text светлосерый (lightgrey), когда поле не в фокусе, и белый, когда имеет фокус. Событие получения фокуса обозначается как <FocusIn> , потери – как <FocusOut> . Для справки: фокус перемещается по виджетам

при нажатии Tab, Ctrl+Tab, Shift+Tab, а также при клике по ним мышью (к кнопкам последнее не относится).



Решение:

Изменени...

—

↗

×

Ширина:

25

Изменить

Высота:

12

Изменени...

—

↗

×

Ширина:

25

Изменить

Высота:

12

Привет

```

18 # Функция для обновления размера текстового поля
19 def update_text_size(event=None):
20     try:
21         width = int(width_entry.get())
22         height = int(height_entry.get())
23
24         text_widget.config(width=width, height=height)
25     except ValueError:
26         messagebox.showerror("Ошибка", "Введите корректные числовые значения!")
27
28
29 # Функция для изменения цвета фона текстового поля при получении фокуса
30 def on_focus_in(event):
31     text_widget.config(bg="white")
32
33
34 # Функция для изменения цвета фона текстового поля при потере фокуса
35 def on_focus_out(event):
36     text_widget.config(bg="lightgrey")

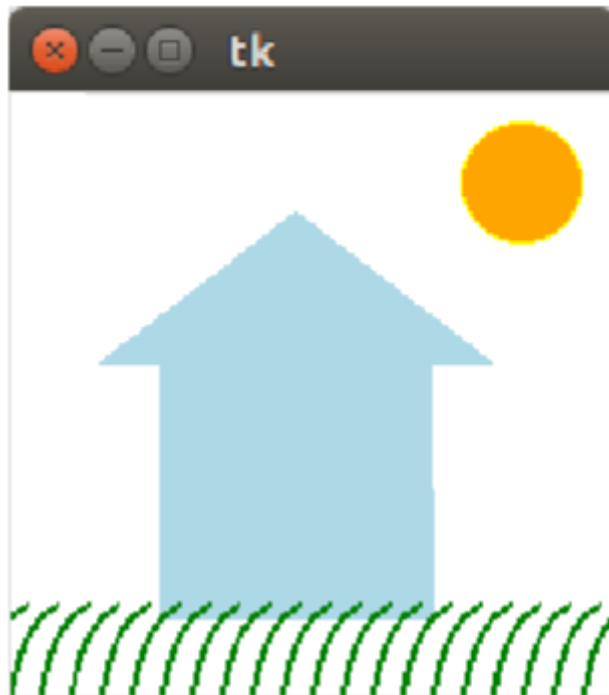
```

```

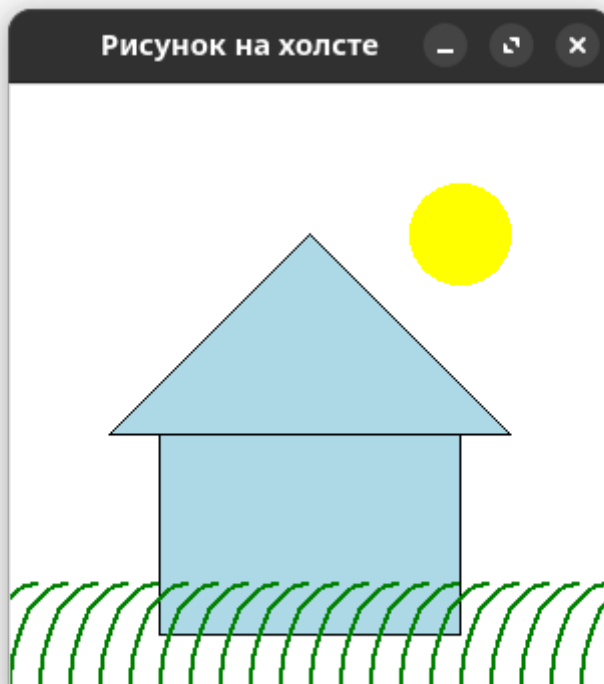
39 # Создаем главное окно приложения
40 root = Tk()
41 root.title("Изменение размера текста")
42
43 # Создаем метку и поле ввода для ширины
44 width_label = Label(root, text="Ширина: ")
45 width_label.grid(row=0, column=0, padx=5, pady=5, sticky="e")
46 width_entry = Entry(root, width=5)
47 width_entry.insert(0, "25")
48 width_entry.grid(row=0, column=1, padx=5, pady=5)
49
50 # Создаем метку и поле ввода для высоты
51 height_label = Label(root, text="Высота: ")
52 height_label.grid(row=1, column=0, padx=5, pady=5, sticky="e")
53 height_entry = Entry(root, width=5)
54 height_entry.insert(0, "12")
55 height_entry.grid(row=1, column=1, padx=5, pady=5)
56
57 # Создаем кнопку для изменения размера текстового поля
58 resize_button = Button(root, text="Изменить", command=update_text_size)
59 resize_button.grid(row=0, column=2, rowspan=2, padx=5, pady=5)
60
61 # Создаем многострочное текстовое поле
62 text_widget = Text(root, width=25, height=12, bg="lightgrey")
63 text_widget.grid(row=2, column=0, columnspan=3, padx=5, pady=5)
64
65 # Привязываем события получения и потери фокуса к соответствующим функциям
66 text_widget.bind("<FocusIn>", on_focus_in)
67 text_widget.bind("<FocusOut>", on_focus_out)
68
69 # Привязываем нажатие Enter к функции обновления размера текстового поля
70 text_widget.bind("<Return>", update_text_size)
71
72
73 # Запуск главного цикла приложения
74 if __name__ == "__main__":
75     root.mainloop()

```

8. Решите задачу: Создайте на холсте подобное изображение:



Решение:



```
8 # Создаем главное окно приложения
9 root = Tk()
10 root.title("Рисунок на холсте")
11
12 # Создаем холст (Canvas) для рисования
13 canvas = Canvas(root, width=300, height=300, bg="white")
14 canvas.pack()
15
16 # Рисуем солнце (круг)
17 canvas.create_oval(200, 50, 250, 100, fill="yellow", outline="yellow")
18 # Рисуем крышу домика (треугольник)
19 canvas.create_polygon(
20     50, 175, 150, 75, 250, 175, fill="lightblue", outline="black"
21 )
22 # Рисуем основание домика (прямоугольник)
23 canvas.create_rectangle(75, 175, 225, 275, fill="lightblue", outline="black")
24
25 # Рисуем траву (серия линий)
26 for x in range(-45, 300, 15):
27     canvas.create_line(
28         x, 300, x + 5, 250, x + 30, 250, fill="green", smooth=True, width=2
29     )
30
31 # Запуск главного цикла приложения
32 if __name__ == "__main__":
33     root.mainloop()
```

Для создания травы используется цикл.

9. Решите задачу: в данной программе создается анимация круга, который движется от левой границы холста до правой:

```
from tkinter import *

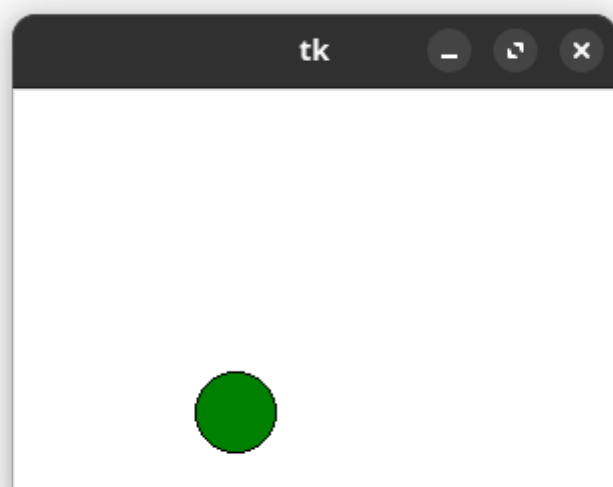
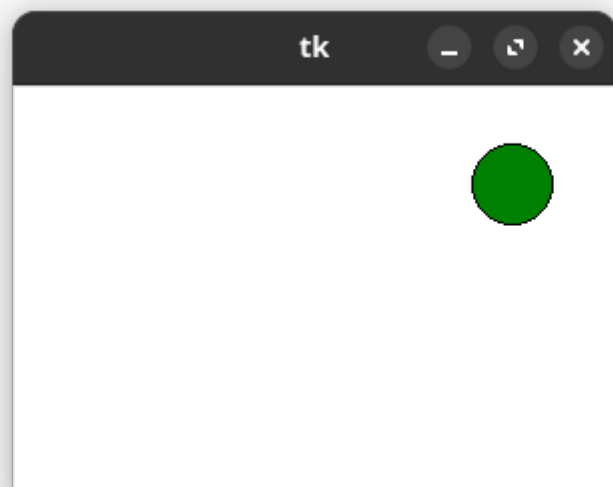
def motion():
    c.move(ball, 1, 0)
    if c.coords(ball)[2] < 300:
        root.after(10, motion)

root = Tk()
c = Canvas(root, width=300, height=200,
bg="white")
c.pack()
ball = c.create_oval(0, 100, 40, 140,
fill='green')
motion()
root.mainloop()
```

Выражение `c.coords(ball)` возвращает список текущих координат объекта (в данном случае это `ball`). Третий элемент списка соответствует его второй координате `x`. Метод `after` вызывает функцию, переданную вторым аргументом, через количество миллисекунд, указанных первым аргументом.

Изучите приведенную программу и самостоятельно запрограммируйте постепенное движение фигуры в ту точку холста, где пользователь кликает левой кнопкой мыши. Координаты события хранятся в его атрибутах `x` и `y` (`event.x` , `event.y`).

Решение:




```

21 # Функция для движения круга к целевой точке
22 def motion():
23     # Расчет разницы координат между текущей позицией круга и целевой точкой
24     dx = target_x - c.coords(ball)[0]
25     dy = target_y - c.coords(ball)[1]
26
27     # Расчет расстояния между текущей позицией и целевой точкой
28     distance = (dx**2 + dy**2) ** 0.5
29
30     # Если расстояние больше 1 пикселя, продолжаем движение
31     if distance > 1:
32         # Нормализация вектора движения
33         dx, dy = dx / distance, dy / distance
34         # Перемещение круга на небольшое расстояние в направлении целевой точки
35         c.move(ball, dx * speed, dy * speed)
36         # Вызов функции motion через 10 миллисекунд для продолжения движения
37         root.after(10, motion)
38
39
40 # Функция для обработки клика левой кнопкой мыши
41 def on_click(event):
42     global target_x, target_y
43     # Обновление целевых координат по клику мыши
44     target_x, target_y = event.x, event.y
45     # Запуск движения круга к новой целевой точке
46     motion()
47
48
49 # Создание главного окна приложения
50 root = Tk()
51
52 # Создание холста (Canvas) для рисования
53 c = Canvas(root, width=300, height=200, bg="white")
54 c.pack()
55
56 # Создание круга в левой части холста
57 ball = c.create_oval(0, 100, 40, 140, fill="green")
58
59 # Инициализация целевых координат как начальной позиции круга
60 target_x, target_y = c.coords(ball)[0], c.coords(ball)[1]
61
62 # Скорость движения круга
63 speed = 2
64
65 # Привязка события клика левой кнопкой мыши к функции on_click
66 c.bind("<Button-1>", on_click)
67
68 # Запуск главного цикла приложения
69 if __name__ == "__main__":
70     root.mainloop()

```

10. Зафиксируйте сделанные изменения в репозитории.
11. Выполните слияние ветки для разработки с веткой main (master).

```

(.venv) → OOP_LR_9 (main) git merge develop
Обновление 8b50def..371d65d
Fast-forward
 .pre-commit-config.yaml | 24 ++++++
 pyproject.toml           |  3 +++
 requirements.txt         |  2 ++
 src/task_1.py            | 107 ++++++
 src/task_2.py            |  51 ++++++
 src/task_3.py            |  75 ++++++
 src/task_4.py            |  31 ++++++
 src/task_5.py            |  70 ++++++
8 files changed, 363 insertions(+)
create mode 100644 .pre-commit-config.yaml
create mode 100644 pyproject.toml
create mode 100644 requirements.txt
create mode 100644 src/task_1.py
create mode 100644 src/task_2.py
create mode 100644 src/task_3.py
create mode 100644 src/task_4.py
create mode 100644 src/task_5.py

```

12. Отправьте сделанные изменения на сервер GitHub.

Контрольные вопросы

1. Каково назначение виджета ListBox?

Виджет Listbox в Tkinter используется для создания списков, внутри которых элементы перечисляются в столбик. Этот виджет позволяет выбирать один или несколько элементов списка.

- Элементы можно добавлять с помощью метода insert.
- По умолчанию можно выбирать только один элемент, но установив selectmode в EXTENDED, можно выбирать несколько элементов, используя клавиши Ctrl или Shift

2. Каким образом осуществляется связывание событие или действие с виджетом Tkinter?

Связывание события или действия с виджетом Tkinter можно осуществить несколькими способами:

- Использование параметра `command`: Этот способ применяется для виджетов, у которых есть такой параметр, например, для кнопок. При создании виджета можно указать функцию, которая будет вызвана при событии (например, при нажатии кнопки)].
- Использование метода `bind`: Этот метод доступен для всех виджетов и позволяет привязать любое событие (например, нажатие клавиш, клики мыши) к конкретной функции-обработчику

3. Какие существуют типы событий в Tkinter? Приведите примеры.

В Tkinter поддерживаются различные типы событий, включая:

Клики мыши:

- `<Button-1>`: Левая кнопка мыши.
- `<Button-2>`: Средняя кнопка мыши (колесико).
- `<Button-3>`: Правая кнопка мыши].

Движение мыши:

- `<Motion>`: Движение мыши.
- `<B1-Motion>`: Движение мыши с нажатой левой кнопкой.

Нажатия клавиш:

- `<Key>`: Нажатие любой клавиши.
- `<Return>`: Нажатие клавиши Enter.
- `<space>`: Нажатие пробела..

Фокус и активность:

- <FocusIn>: Получение фокуса виджетом.
- <FocusOut>: Потеря фокуса виджетом.

4. Как обрабатываются события в Tkinter?

События в Tkinter обрабатываются с помощью функций-обработчиков, которые можно связать с виджетами следующими способами:

- Через параметр `command`: Для виджетов, поддерживающих этот параметр, например, для кнопок.
- Через метод `bind`: Для любых виджетов, позволяя привязать событие к конкретной функции.

5. Как обрабатываются события мыши в Tkinter?

События мыши обрабатываются аналогичным образом, используя метод `bind`. Например, чтобы обработать клик левой кнопки мыши:

```
1 from tkinter import Tk, Frame
2
3 def mouse_click(event):
4     print("Левая кнопка мыши нажата")
5
6 root = Tk()
7 frame = Frame(root)
8 frame.bind("<Button-1>", mouse_click)
9 frame.pack()
10 root.mainloop()
```

6. Каким образом можно отображать графические примитивы в Tkinter?

Для отображения графических примитивов в Tkinter можно использовать виджет Canvas. Основные методы для отображения графических примитивов на холсте включают:

- `create_line` для рисования линий.
- `create_rectangle` для рисования прямоугольников.
- `create_oval` для рисования эллипсов.
- `create_polygon` для рисования полигонов.
- `create_text` для отображения текста.
- `create_image` для отображения изображений.

7. Перечислите основные методы для отображения графических примитивов в Tkinter.

Основные методы для отображения графических примитивов в Tkinter

- `create_line`: Рисование линий.
- `create_rectangle`: Рисование прямоугольников.
- `create_oval`: Рисование эллипсов.
- `create_polygon`: Рисование полигонов.
- `create_text`: Отображение текста.
- `create_image`: Отображение изображений.

8. Каким образом можно обратиться к ранее созданным фигурам на холсте?

На холсте Canvas можно обращаться к ранее созданным фигурам используя их идентификаторы, которые возвращаются методами создания фигур.

9. Каково назначение тэгов в Tkinter?

Тэги в Tkinter используются для группировки и манипуляции с несколькими элементами на холсте Canvas. Тэги можно присвоить при создании элементов и затем использовать для изменения или удаления этих элементов.