

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

Институт перспективной инженерии  
Департамента цифровых, роботехнических систем и электроники

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №8**  
**дисциплины «Объектно-ориентированное программирование»**

Выполнила:  
Гайчук Дарья Дмитриевна  
3 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика  
и вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем», очная  
форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А.-доцент департамента  
цифровых, роботехнических систем и  
электроники института перспективной  
инженерии

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2024 г.

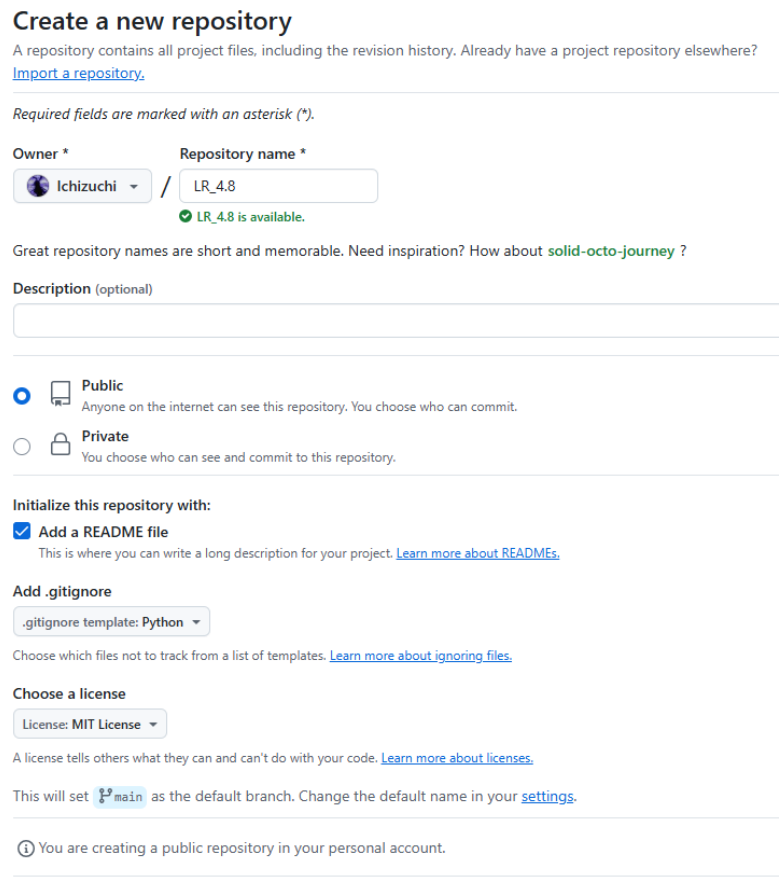
Тема: «Основы работы с Tkinter»

Цель работы: приобретение навыков построения графического интерфейса пользователя GUI с помощью пакета Tkinter языка программирования Python версии 3.x.

Ссылка на репозиторий: [https://github.com/Ichizuchi/LR\\_4.8](https://github.com/Ichizuchi/LR_4.8)

Порядок выполнения работы:

1. Создала новый репозиторий и клонировала его на свой компьютер.



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, there's a section for 'Required fields are marked with an asterisk (\*)'. The 'Owner' field is set to 'Ichizuchi' and the 'Repository name' field is set to 'LR\_4.8'. A green checkmark indicates that 'LR\_4.8 is available'. There's a note about great repository names being short and memorable, with a link to 'solid-octo-journey'. The 'Description' field is optional and 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'. At the bottom, it says 'This will set main as the default branch. Change the default name in your settings.' and a note that 'You are creating a public repository in your personal account.'

Рисунок 1. Создание репозитория

2. Клонировала репозиторий на свой компьютер.

```
● @Ichizuchi →/workspaces/LR_4.8 (main) $ git branch develop
● @Ichizuchi →/workspaces/LR_4.8 (main) $ git checkout develop
Switched to branch 'develop'
```

Рисунок 2. Модель ветвления git-flow

3. Создала виртуальное окружение Anaconda с именем репозитория.

```

@Ichizuchi →/workspaces/LR 4.8 (develop) $ conda create -n myenv python=3.10
Retrieving notices: ...working... done
Channels:
  - defaults
Platform: linux-64
Collecting package metadata (repodata.json): \

```

Рисунок 3. Создание виртуального окружения

## Выполнение заданий

### Вариант 3

Задание №1. 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.

```

Tasks > Task_1.py > main
4  def move_items(source_listbox, target_listbox):
5      selected_items = source_listbox.curselection()
6      if not selected_items:
7          messagebox.showinfo("Информация", "Выберите элементы для перемещения.")
8          return
9
10     for index in selected_items[::-1]:
11         item = source_listbox.get(index)
12         target_listbox.insert(tk.END, item)
13         source_listbox.delete(index)
14
15 def main():
16     root = tk.Tk()
17     root.title("Список покупок")
18
19     # Фреймы
20     frame_lists = tk.Frame(root)
21     frame_lists.pack(pady=10)
22
23     frame_buttons = tk.Frame(root)
24     frame_buttons.pack()
25
26     # перечень товаров
27     listbox_items = tk.Listbox(frame_lists, selectmode=tk.MULTIPLE, width=30, height=15)
28     listbox_items.pack(side=tk.LEFT, padx=5)
29
30     items = ["Печенье", "Сок", "Бананы", "Творог", "Овсянка", "Морковь", "Картофель", "Курица", "Салат", "Гречка"]
31     for item in items:
32         listbox_items.insert(tk.END, item)
33
34     # перечень покупок
35     listbox_shopping = tk.Listbox(frame_lists, selectmode=tk.MULTIPLE, width=30, height=15)
36     listbox_shopping.pack(side=tk.RIGHT, padx=5)
37
38     # Кнопки управления
39     button_add = tk.Button(frame_buttons, text="Добавить в покупки",
40                           command=lambda: move_items(listbox_items, listbox_shopping))
41     button_add.pack(side=tk.LEFT, padx=5, pady=5)
42
43     button_remove = tk.Button(frame_buttons, text="Удалить из покупок",
44                              command=lambda: move_items(listbox_shopping, listbox_items))
45     button_remove.pack(side=tk.RIGHT, padx=5, pady=5)
46
47     root.mainloop()

```

Рисунок 6. Код программы

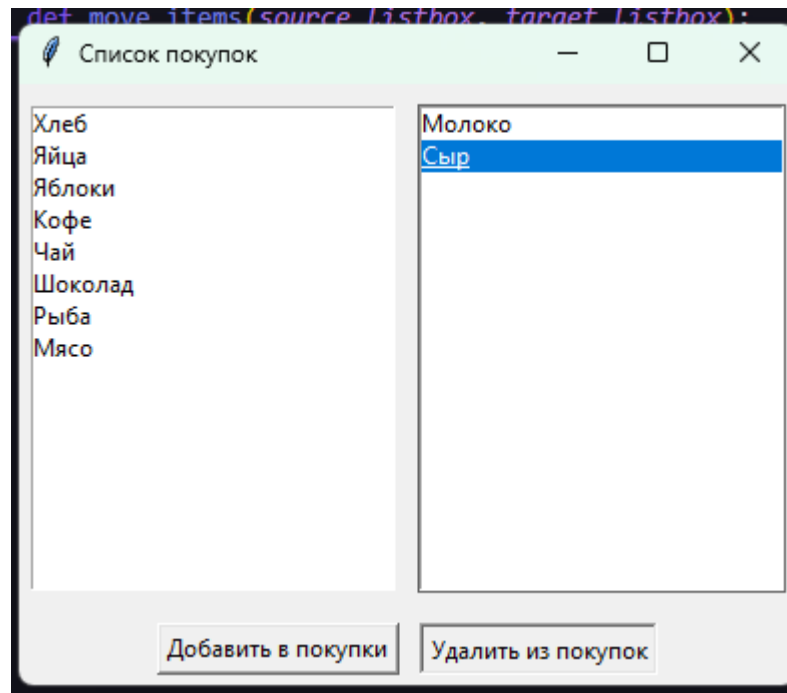


Рисунок 7. Вывод задания №1

Задание №2. 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.

```

Tasks > Task_2.py > main
1  import tkinter as tk
2
3  def add_to_listbox(event):
4      """Добавляет текст из текстового поля в список при нажатии Enter."""
5      text = entry.get().strip()
6      if text:
7          listbox.insert(tk.END, text)
8          entry.delete(0, tk.END)
9
10 def copy_to_entry(event):
11     """Копирует выбранный элемент из списка в текстовое поле при двойном клике."""
12     selection = listbox.curselection()
13     if selection:
14         entry.delete(0, tk.END)
15         entry.insert(0, listbox.get(selection[0])) # Копирование текста
16
17 def main():
18     root = tk.Tk()
19     root.title("Текстовое поле и список")
20
21     # Однострочное текстовое поле
22     global entry
23     entry = tk.Entry(root, width=40)
24     entry.pack(pady=10)
25     entry.bind("<Return>", add_to_listbox)
26
27     # Список (Listbox)
28     global listbox
29     listbox = tk.Listbox(root, width=40, height=10)
30     listbox.pack(pady=10)
31     listbox.bind("<Double-Button-1>", copy_to_entry)
32
33     root.mainloop()
34
35 if __name__ == "__main__":
36     main()
37

```

Рисунок 8. Код программы

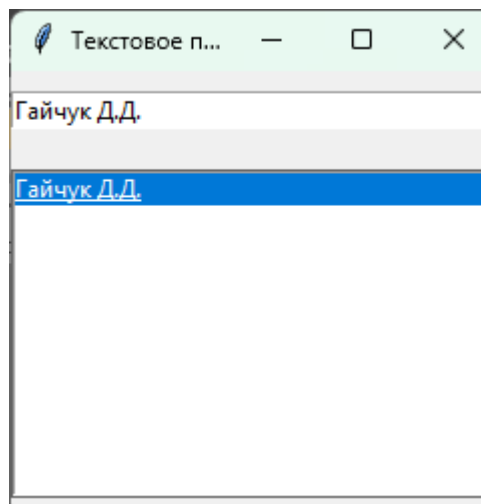


Рисунок 9. Вывод задания №2

Задание №3. 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).

```
Tasks > Task_3.py > ...
1  import tkinter as tk
2
3  def resize_text_widget(event=None):
4      """Изменяет размер многострочного текстового поля в соответствии со значениями полей ввода."""
5      try:
6          width = int(entry_width.get())
7          height = int(entry_height.get())
8          text_widget.config(width=width, height=height)
9      except ValueError:
10         tk.messagebox.showerror("Ошибка", "Введите корректные числовые значения!")
11
12 def set_focus_color(event):
13     text_widget.config(bg="white")
14
15 def unset_focus_color(event):
16     text_widget.config(bg="lightgrey")
17
18 root = tk.Tk()
19 root.title("Изменение размеров Text")
20
21 # Поля для ввода ширины и высоты
22 entry_width = tk.Entry(root, width=5)
23 entry_width.insert(0, "25")
24 entry_width.grid(row=0, column=0, padx=5, pady=5)
25
26 entry_height = tk.Entry(root, width=5)
27 entry_height.insert(0, "12")
28 entry_height.grid(row=1, column=0, padx=5, pady=5)
29
30 # Кнопка для изменения размеров
31 btn_resize = tk.Button(root, text="Изменить", command=resize_text_widget)
32 btn_resize.grid(row=0, column=1, rowspan=2, padx=5, pady=5)
33
34 # Многострочное текстовое поле
35 text_widget = tk.Text(root, width=25, height=12, bg="lightgrey")
36 text_widget.grid(row=2, column=0, columnspan=2, padx=5, pady=5)
37
38 entry_width.bind("<Return>", resize_text_widget)
39 entry_height.bind("<Return>", resize_text_widget)
40 text_widget.bind("<FocusIn>", set_focus_color)
41 text_widget.bind("<FocusOut>", unset_focus_color)
42
43 root.mainloop()
44
```

Рисунок 10. Код программы

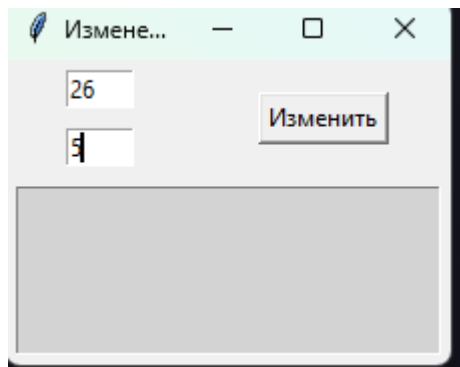


Рисунок 11. Вывод задания №3

Задание №4. Создайте на холсте подобное изображение:



Рисунок 12. Пример изображения

```

Tasks > Task_4.py > main
1  import tkinter as tk
2
3  def main():
4      root = tk.Tk()
5      root.title("Рисунок на холсте")
6
7      # холст
8      canvas = tk.Canvas(root, width=300, height=300, bg="white")
9      canvas.pack()
10
11     # прямоугольник
12     canvas.create_rectangle(100, 150, 200, 250, fill="lightblue", outline="")
13
14     # крыша
15     canvas.create_polygon(80, 150, 220, 150, 150, 90, fill="lightblue", outline="")
16
17     # солнце
18     canvas.create_oval(220, 40, 260, 80, fill="orange", outline="")
19
20     # трава
21     for i in range(0, 300, 10):
22         canvas.create_line(i, 250, i + 15, 230, fill="green", width=2)
23
24     root.mainloop()
25
26 if __name__ == "__main__":
27     main()
28

```

Рисунок 13. Код программы

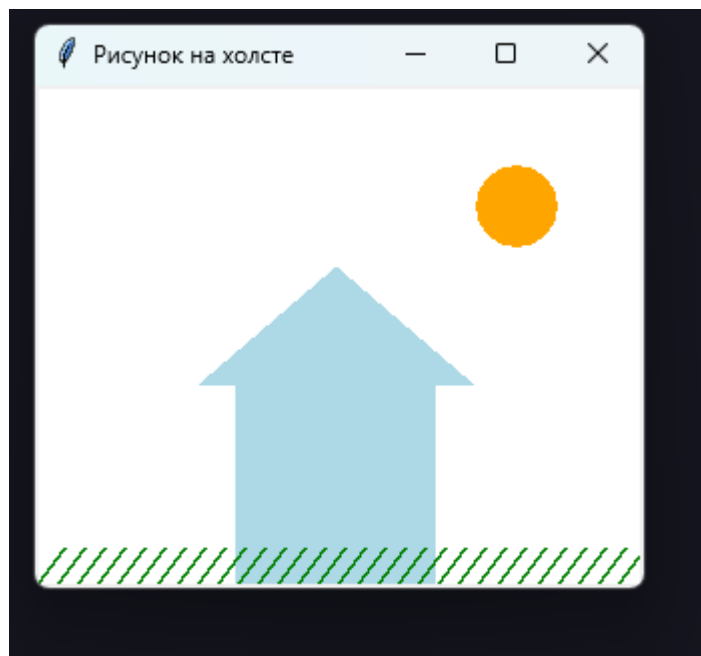


Рисунок 14. Вывод задания №4

Задание №5. 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 `y`. 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`).

```

Tasks > Task_5.py > ...
3  def motion():
4      # координаты центра круга
5      x0, y0, x1, y1 = c.coords(ball)
6      ball_center_x = (x0 + x1) / 2
7      ball_center_y = (y0 + y1) / 2
8
9
10     # разница между текущими и целевыми координатами
11     dx = target_x - ball_center_x
12     dy = target_y - ball_center_y
13
14     # Скорость
15     step_x = 1 if dx > 0 else -1 if dx < 0 else 0
16     step_y = 1 if dy > 0 else -1 if dy < 0 else 0
17
18     c.move(ball, step_x, step_y)
19
20     if abs(dx) > 1 or abs(dy) > 1:
21         root.after(10, motion)
22
23  def on_click(event):
24      global target_x, target_y
25      target_x = event.x
26      target_y = event.y
27      motion()
28
29  root = Tk()
30
31  # холст
32  c = Canvas(root, width=300, height=200, bg="white")
33  c.pack()
34
35  # круг
36  ball = c.create_oval(0, 100, 40, 140, fill='green')
37
38  # Начальные координаты
39  target_x = 20
40  target_y = 120
41
42  # Привязка клика мыши
43  c.bind("<Button-1>", on_click)
44  |
45  root.mainloop()
46

```

Рисунок 15. Код программы

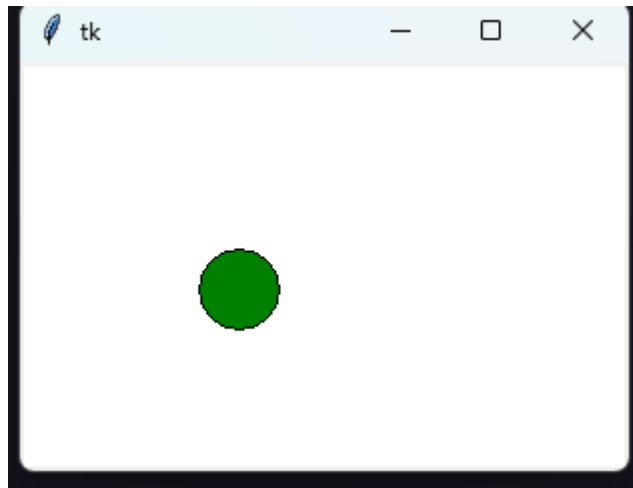


Рисунок 16. Вывод задания №5

```

• @Ichizuchi →/workspaces/LR_4.8 (develop) $ conda env export > environment.yml
• @Ichizuchi →/workspaces/LR_4.8 (develop) $ pip freeze > requirements.txt
• @Ichizuchi →/workspaces/LR_4.8 (develop) $ conda init
no change      /opt/conda/condabin/conda
no change      /opt/conda/bin/conda
no change      /opt/conda/bin/conda-env
no change      /opt/conda/bin/activate
no change      /opt/conda/bin/deactivate
no change      /opt/conda/etc/profile.d/conda.sh
no change      /opt/conda/etc/fish/conf.d/conda.fish
no change      /opt/conda/shell/condabin/Conda.psm1
no change      /opt/conda/shell/condabin/conda-hook.ps1
no change      /opt/conda/lib/python3.12/site-packages/xontrib/conda.xsh
no change      /opt/conda/etc/profile.d/conda.csh
modified       /home/codespace/.bashrc

==> For changes to take effect, close and re-open your current shell. <==

• @Ichizuchi →/workspaces/LR_4.8 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
• @Ichizuchi →/workspaces/LR_4.8 (main) $ git merge develop
Updating a5a94d9..dc8257a
Fast-forward
 Tasks/Task_1.py | 50 +++++++++++++++++++++++++++++++++++++++++++++++++++++
 Tasks/Task_2.py | 36 +++++++++++++++++++++++++++++++++++++
 Tasks/Task_3.py | 43 +++++++++++++++++++++++++++++++++++++
 Tasks/Task_4.py | 27 +++++++++++++++++++++
 Tasks/Task_5.py | 45 +++++++++++++++++++++++++++++++++++++
 5 files changed, 201 insertions(+)
 create mode 100644 Tasks/Task_1.py
 create mode 100644 Tasks/Task_2.py
 create mode 100644 Tasks/Task_3.py
 create mode 100644 Tasks/Task_4.py
 create mode 100644 Tasks/Task_5.py

```

Рисунок 17. Слияние веток

## Ответы на контрольные вопросы

### 1. Каково назначение виджета «Listbox»?

«Listbox» — это виджет Tkinter для отображения списка элементов, из которого можно выбирать один или несколько элементов. Используется, когда необходимо предоставить пользователю возможность выбора из списка, например, для выбора опций, файлов или категорий.

2. Каким образом осуществляется связывание события или действия с виджетом Tkinter?

Связывание события с виджетом осуществляется с помощью метода «`widget.bind(event, handler)`», где «event» — это строка, описывающая событие (например, `<Button-1>` для клика мыши), а `handler` — функция-обработчик, которая вызывается при возникновении события.

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

Основные типы событий в Tkinter включают:

- Клавиатурные события: `<KeyPress>` (нажатие клавиши), `<KeyRelease>` (отпускание клавиши)
- Мышиные события: `<Button-1>` (левая кнопка мыши), `<Button-3>` (правая кнопка мыши), `<Double-Button-1>` (двойной клик)
- События окна: `<Configure>` (изменение размера окна), `<FocusIn>` и `<FocusOut>` (получение и потеря фокуса)
- Системные события: `<Destroy>` (заккрытие окна)

Пример: «`button.bind(<Button-1>, handler_function)`» связывает клик левой кнопки мыши с «button».

### 4. Как обрабатываются события в Tkinter?

В Tkinter события обрабатываются функциями-обработчиками, которые привязываются к виджетам через «`bind`». Когда событие происходит, вызовется привязанная функция-обработчик. Аргументом функции будет объект события («event»), содержащий информацию о типе события, координатах и других параметрах.

### 5. Как обрабатываются события мыши в Tkinter?

События мыши обрабатываются с помощью «`bind`», указывая тип события (например, `<Button-1>`, `<Motion>` для перемещения курсора). Объект события передаёт данные, такие как «`event.x`» и «`event.y`» — координаты указателя мыши, что позволяет определить точное местоположение курсора при возникновении события.

### 6. Каким образом можно отображать графические примитивы в Tkinter?

Графические примитивы отображаются с помощью виджета «Canvas», на котором можно рисовать линии, окружности, прямоугольники, многоугольники, текст и изображения. «Canvas» предоставляет методы для создания и управления этими объектами.

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

Основные методы «Canvas» для отображения графических примитивов:

- «create\_line()» — рисует линию
- «create\_rectangle()» — рисует прямоугольник
- «create\_oval()» — рисует эллипс или круг
- «create\_polygon()» — рисует многоугольник
- «create\_text()» — добавляет текст
- «create\_image()» — отображает изображение

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

Для обращения к фигуре используется её идентификатор, который возвращается при создании. Фигурам также можно назначить тэги, что позволяет обращаться к ним группами. Методы, такие как «itemconfig» и «coords», позволяют изменять свойства и положение фигур по их идентификатору или тэгу.

9. Каково назначение тэгов в Tkinter?

Тэги позволяют группировать объекты на «Canvas», чтобы управлять ими вместе. Например, можно применить одинаковое действие ко всем фигурам с заданным тэгом, изменять их цвет, позицию или удалять их из холста одним вызовом команды.

Вывод: в ходе работы были приобретены навыки по построению графического интерфейса пользователя GUI с помощью пакета Tkinter языка программирования Python версии 3.x.

