



алгоритмика

# PYTHON

Занятие 10

# Сегодня на уроке

---

- Модуль Tkinter
- Классы и объекты
- Работа с окном и Canvas

# Вопрос

---

1. Как подключается модуль в Python?
2. Что такое объект?
3. Для чего используются объекты в программе?
4. Что такое метод?

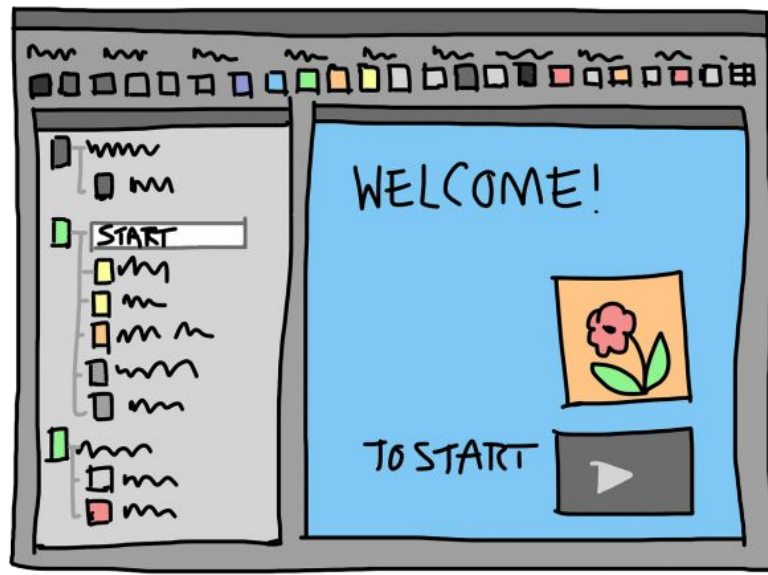
# Модуль tkinter

---

**tkinter** – **tool**kit **inter**face – инструментарий для создания пользовательского интерфейса.

Подключение модуля:

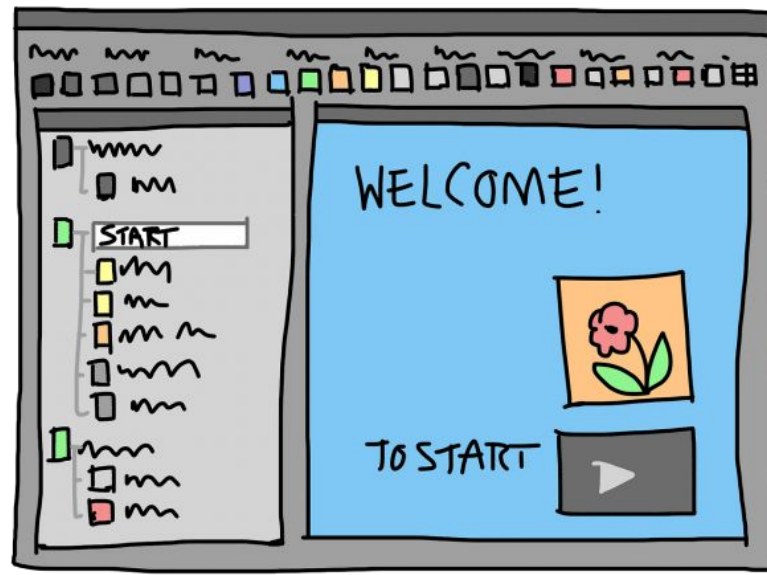
```
from tkinter import*
```



# Вопрос

---

Какие элементы используются при создании интерфейса приложения?

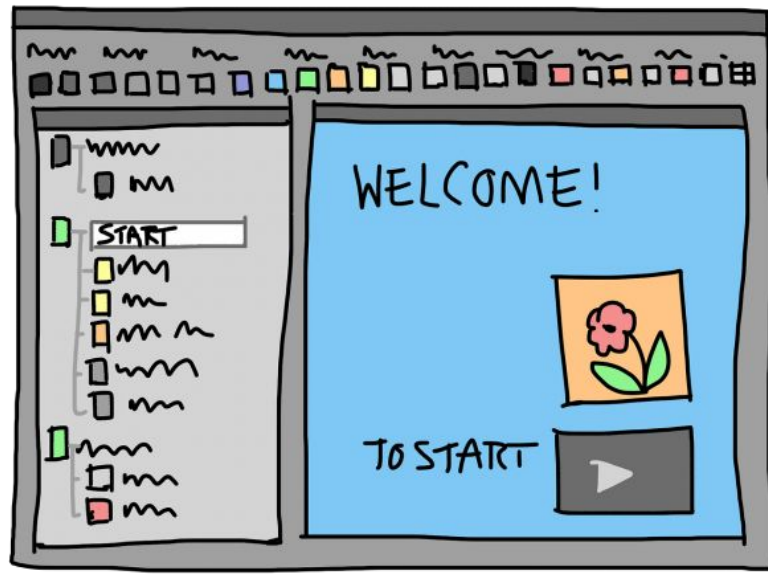


# Вопрос

---

Какие элементы используются при создании интерфейса приложения?

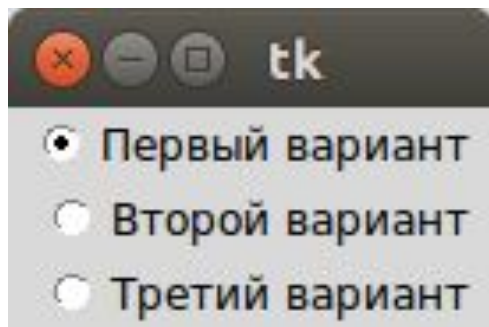
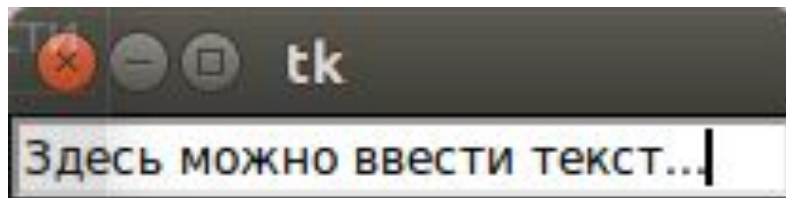
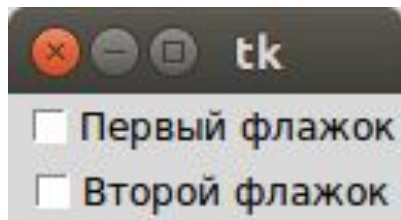
- картинки
- кнопки
- надписи



# Виджет

---

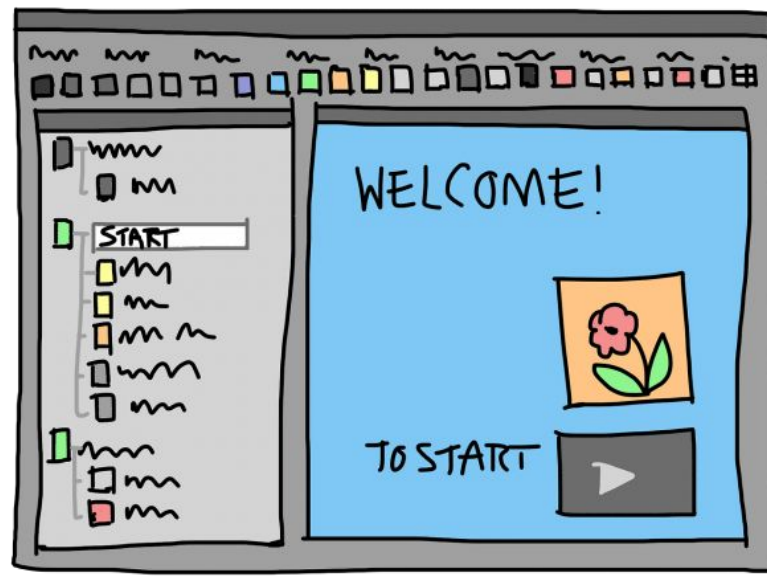
**Виджет** – элемент управления графического интерфейса, с которым взаимодействует пользователь.



# Вопрос

---

Где размещаются все элементы интерфейса?



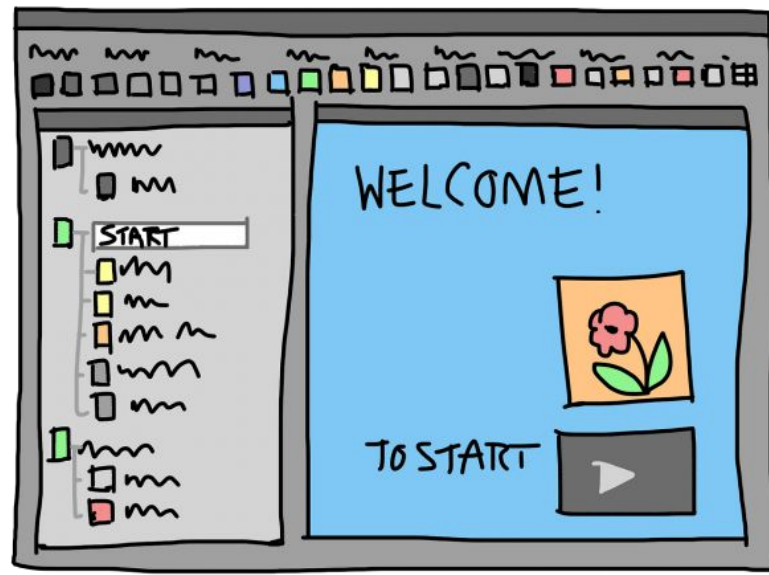


# Вопрос

---

Где размещаются все элементы интерфейса?

В окне



# Создание объекта в Python

---

Для создания объекта используется следующая конструкция:

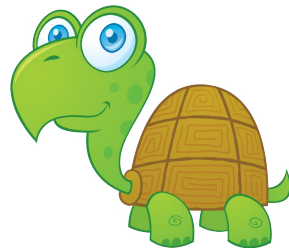
```
first=Turtle()
```



Имя объекта



Кому принадлежит этот объект



# Вопрос

---

Какой конструкцией мы будем пользоваться при создании объекта в модуле tkinter?

# Вопрос

---

Какой конструкцией мы будем пользоваться при создании объекта в модуле tkinter?

```
root = Tk()
```

имя объекта

кому принадлежит  
объект



# Класс в Python

---

**Класс** – это тип объекта, который мы создаём в программе.

**root** = **Tk()**

имя объекта

класс объекта

**Tk()** – базовый класс модуля. При создании объекта создается окно приложения.

# Задание

---

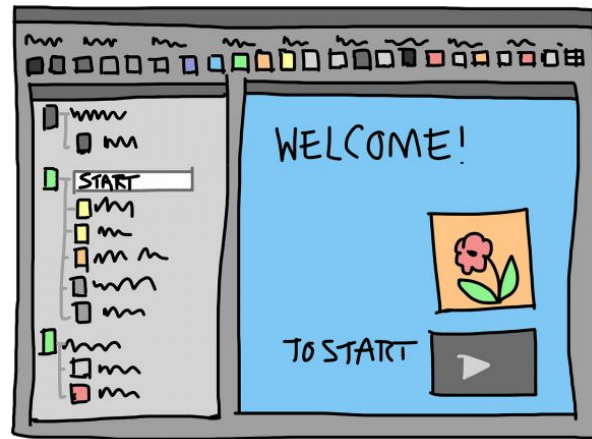
Подключите модуль tkinter и создайте объект root класса Tk().

```
from tkinter import*  
root=Tk()
```

# Вопрос

---

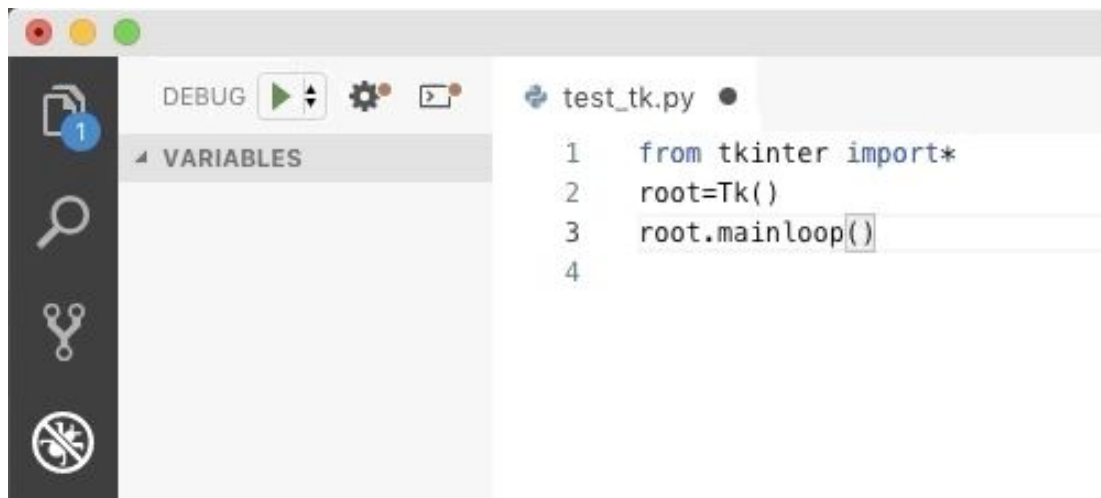
- Что произошло после запуска программы?
- Почему не появилось созданное окно?
- Встречались ли мы с таким уже ранее?



# Задержка окна

**mainloop()** – **метод** класса Tk(), который не даёт разрушиться созданному объекту.

```
from tkinter import*  
root=Tk()  
root.mainloop()
```





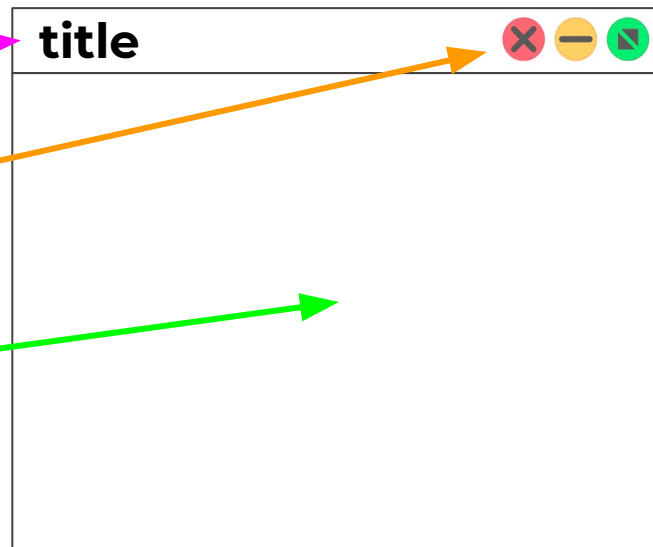
# Окно

---

**Заголовок окна** - title

**Управляющие кнопки окна** -  
заккрыть, свернуть, развернуть  
на весь экран

**Содержимое окна** - кнопки,  
текст, изображения и т.д.

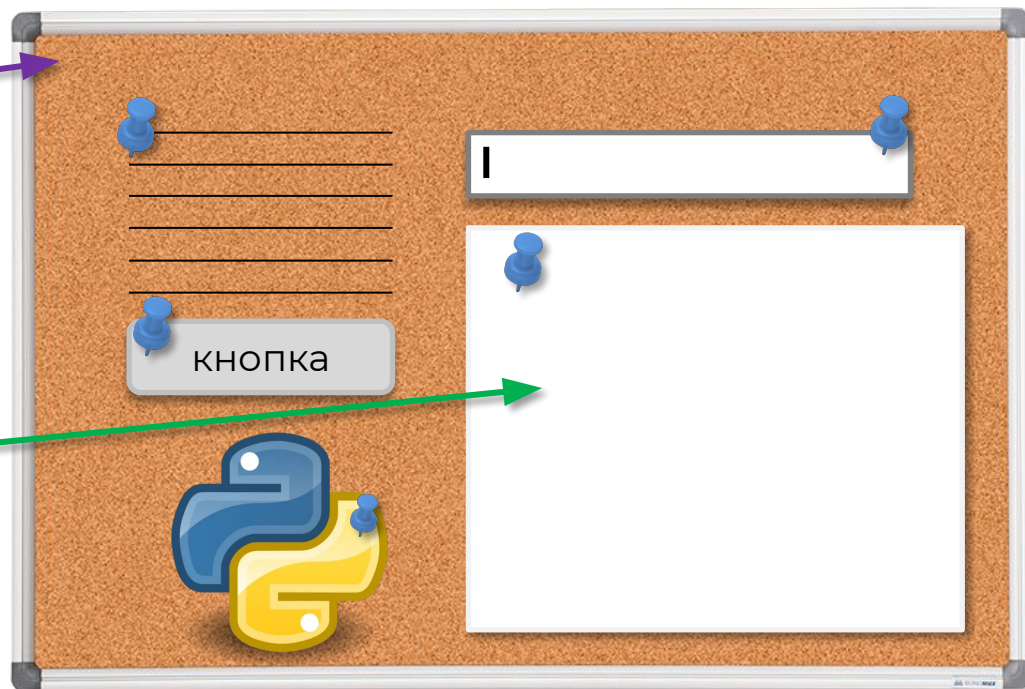


# Элементы интерфейса

---

**Окно** – основа интерфейса

**Холст** – область для графического отображения объектов



# Canvas

---

**Canvas** – холст. Класс модуля tkinter. Область для графического отображения объектов.

```
canvas = Canvas(имя_окна, ширина_холста, длина_холста)
```

```
canvas = Canvas(root, width=640, height=480)
```

# Задание

---

Добавьте в программу создание холста размером 640\*480.

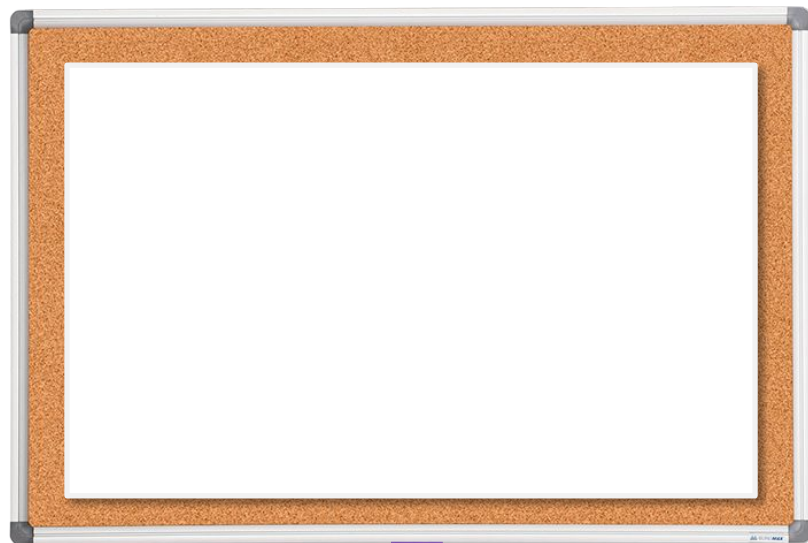
```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
root.mainloop()
```

# Вопрос

---

Изменилось ли наше окно?

```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
root.mainloop()
```



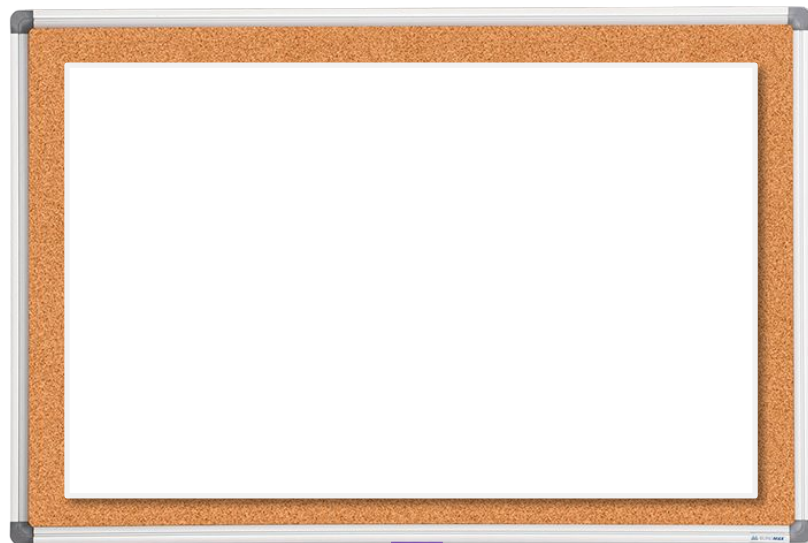
# Вопрос

---

Изменилось ли наше окно?

```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
root.mainloop()
```

**Нет окно осталось  
прежним**

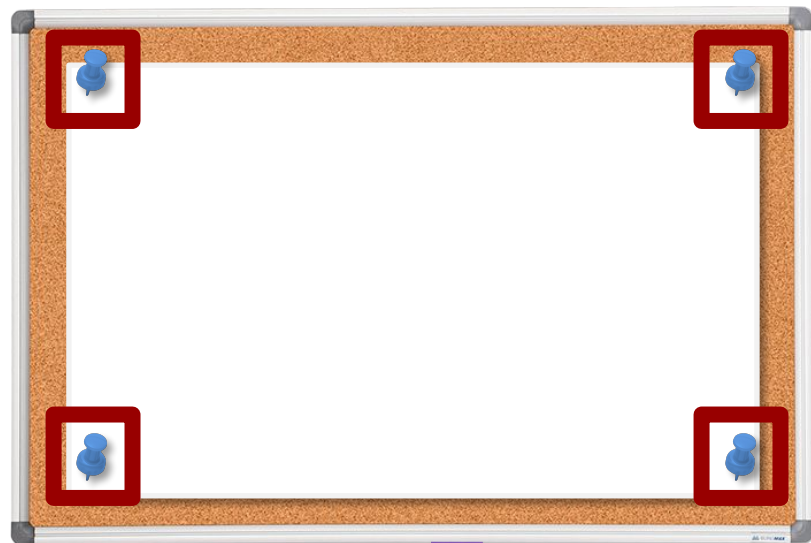


# Вопрос

---

Почему холст не появился?

```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
root.mainloop()
```



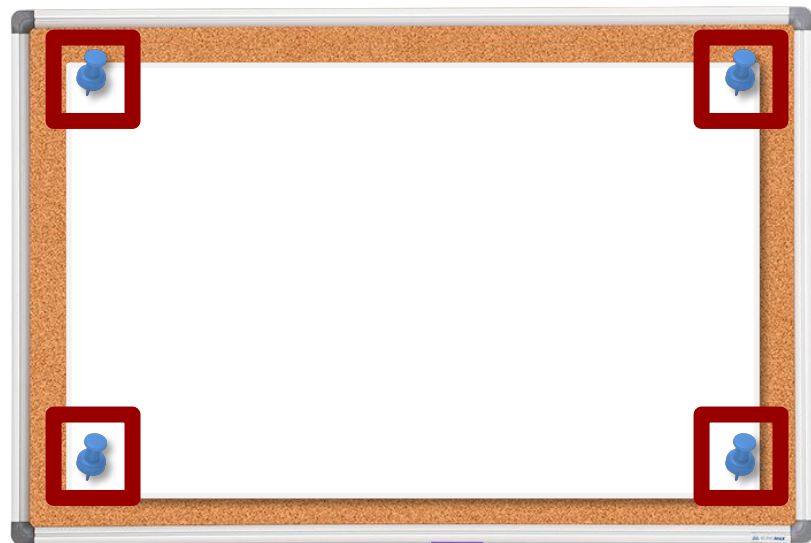
# Вопрос

---

Почему холст не появился?

```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
root.mainloop()
```

**Его не закрепили**



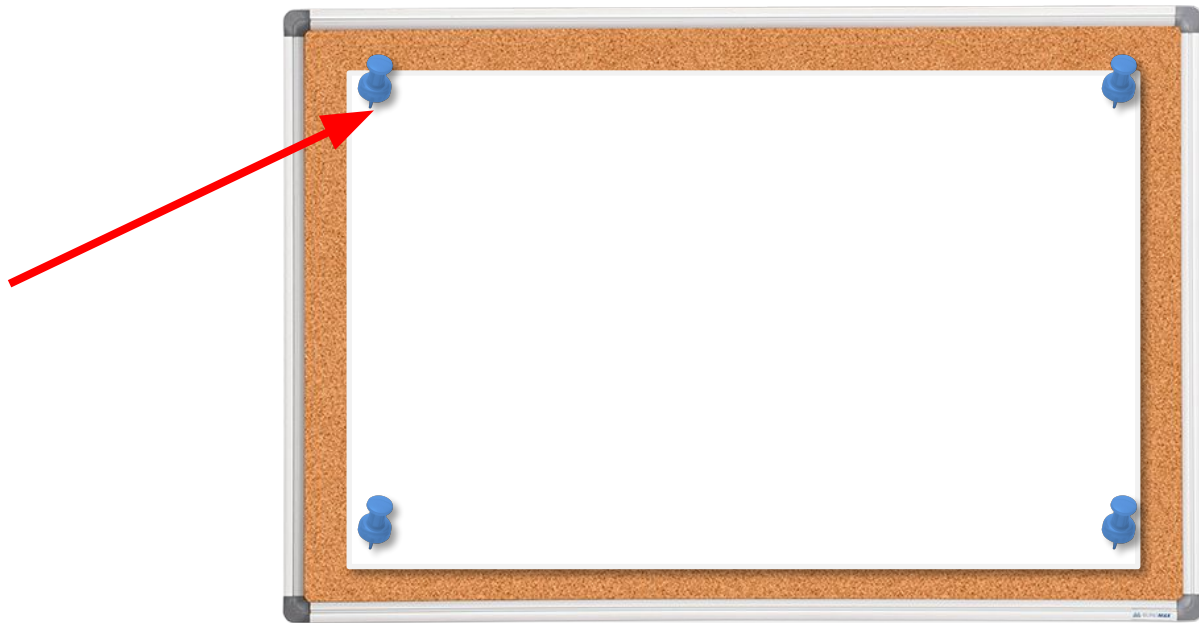


# Закрепление элемента

---

**pack()** – метод для закрепления объекта.

**canvas.pack()**



# Задание

---

Добавьте в программу закрепление холста.

```
from tkinter import*
```

```
root=Tk()
```

```
canvas = Canvas(root,width=640,height=480)
```

```
canvas.pack()
```

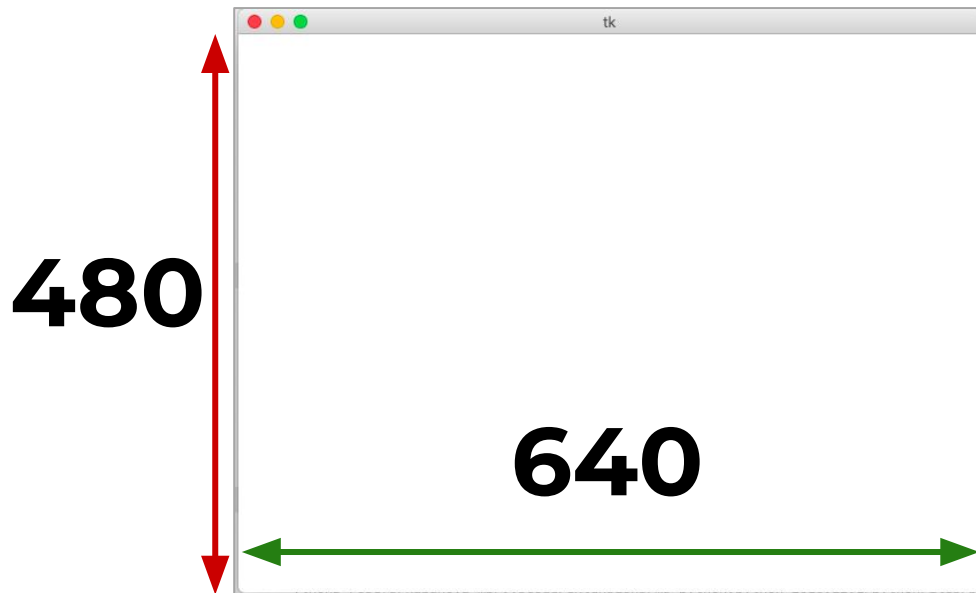
```
root.mainloop()
```

# Задание. Проверка

---

Добавьте в программу закрепление холста.

```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
canvas.pack()  
root.mainloop()
```



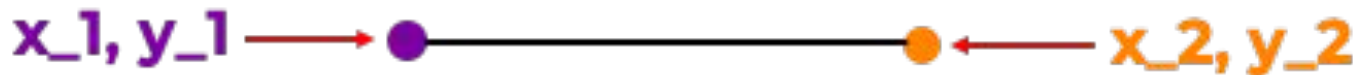
# Создание линии

---

`create_line()` – метод для создания линии.

`create_line(x_1, y_1, x_2, y_2)`

координаты **начала** линии      координаты **конца** линии




# Задание на листе

---

Нарисуйте линию.

`canvas.create_line(0,0,100,100)`



начало                      конец

# Задание

---

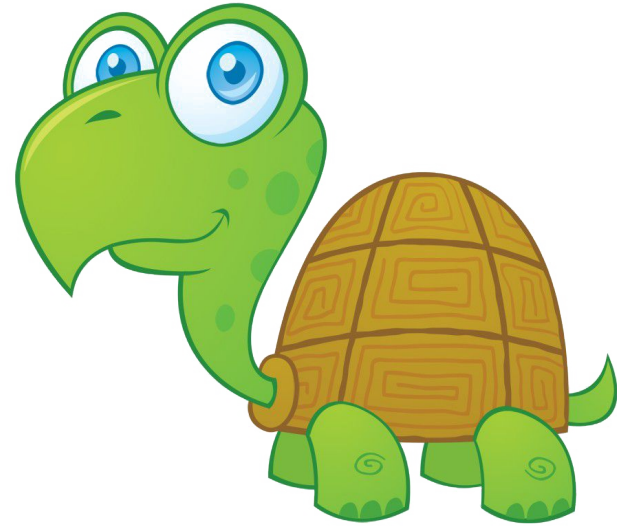
Добавьте вызов метода `create_line` в программу и проверьте свой рисунок.

```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
canvas.pack()  
canvas.create_line(0,0,100,100)  
root.mainloop()
```

# Вопрос

---

Откуда начинался отсчёт координат в модуле turtle?

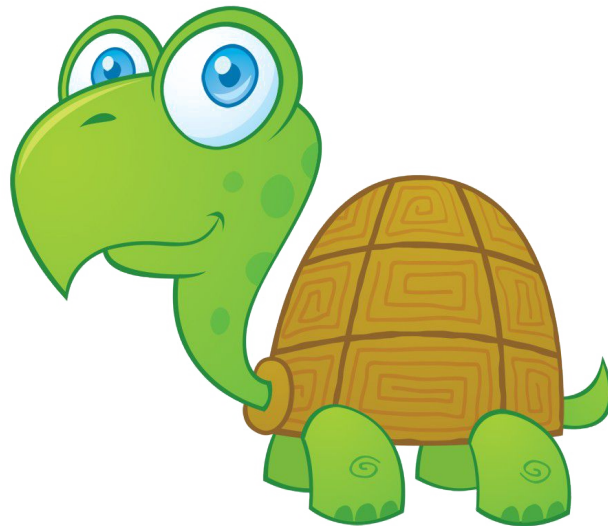
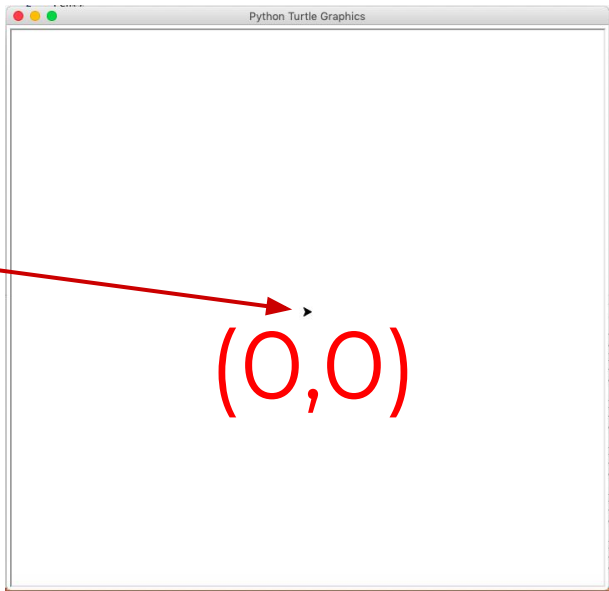


# Вопрос

---

Откуда начинался отсчёт координат в модуле turtle?

С центра



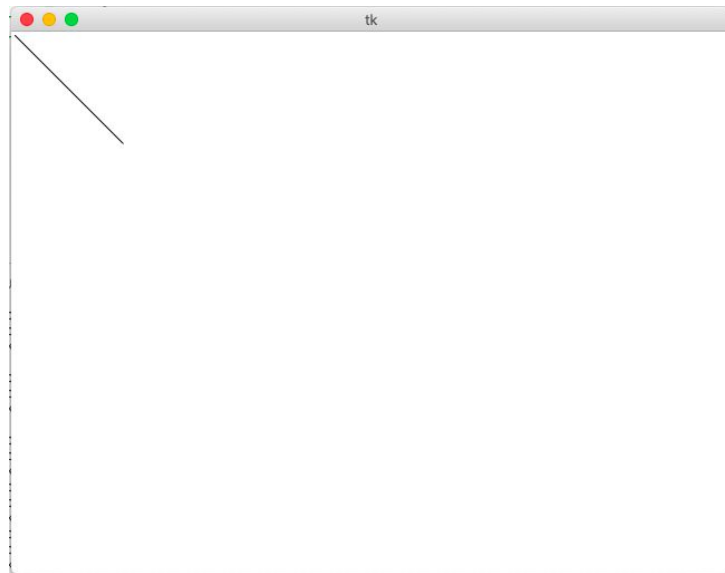


# Задание. Проверка

---

Добавьте вызов метода `create_line` в программу и проверьте свой рисунок.

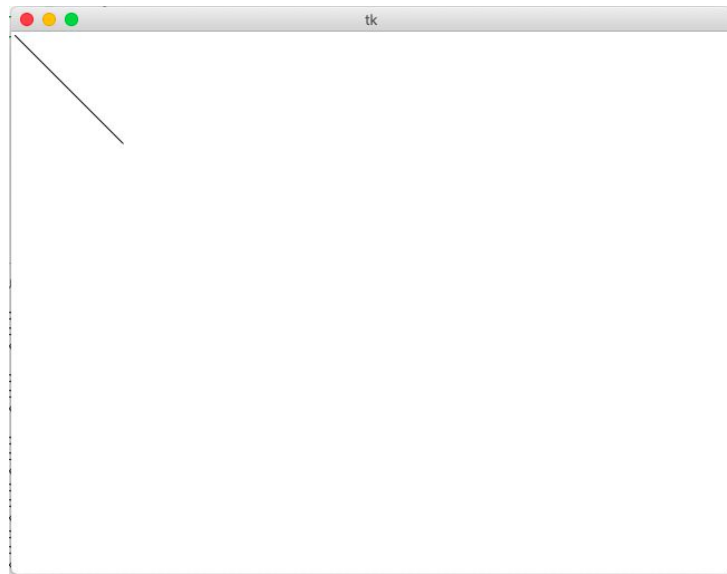
```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
canvas.pack()  
canvas.create_line(0,0,100,100)  
root.mainloop()
```



# Вопрос

---

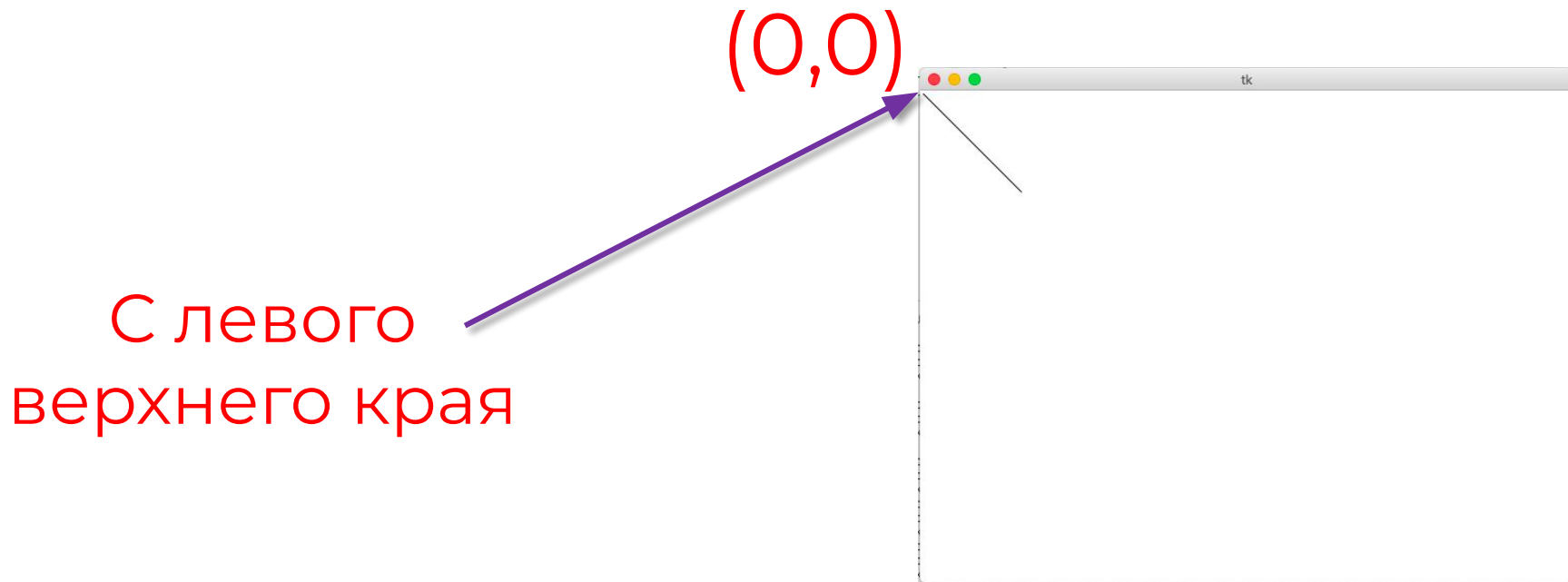
Откуда начинается отсчёт координат в окне tkinter?



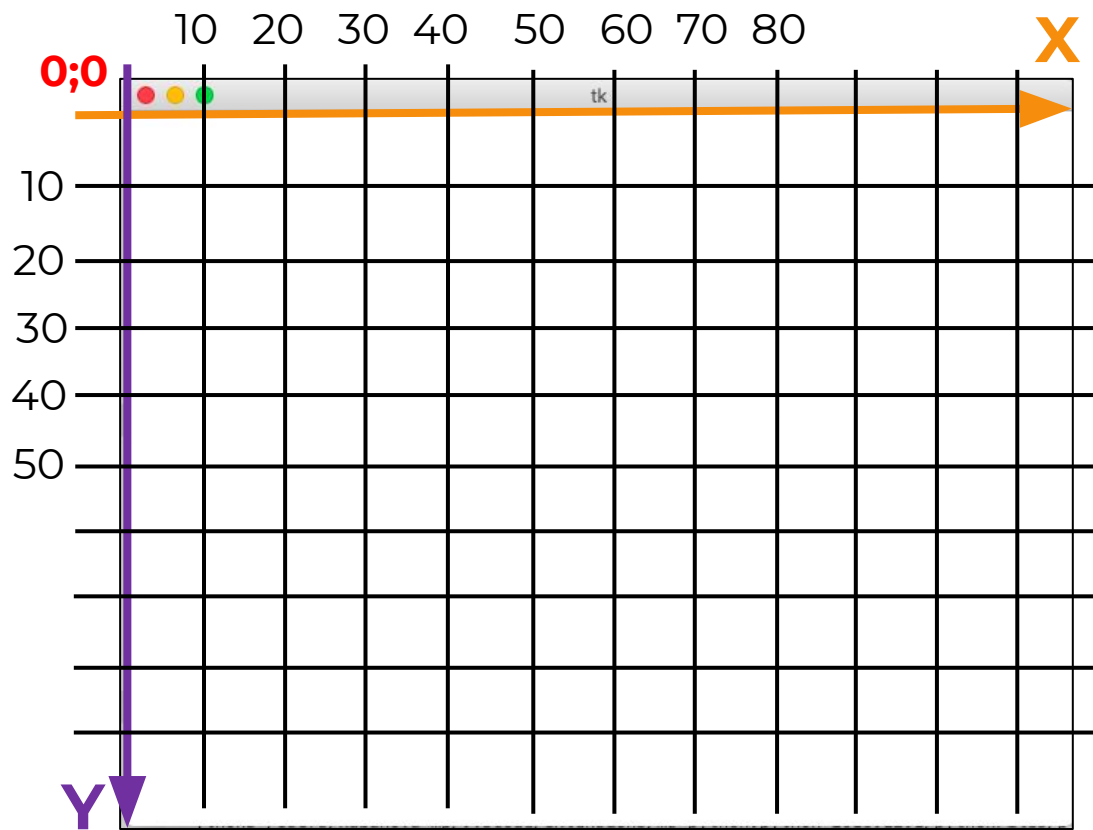
# Вопрос

---

Откуда начинается отсчёт координат в окне tkinter?



# Изменение координат

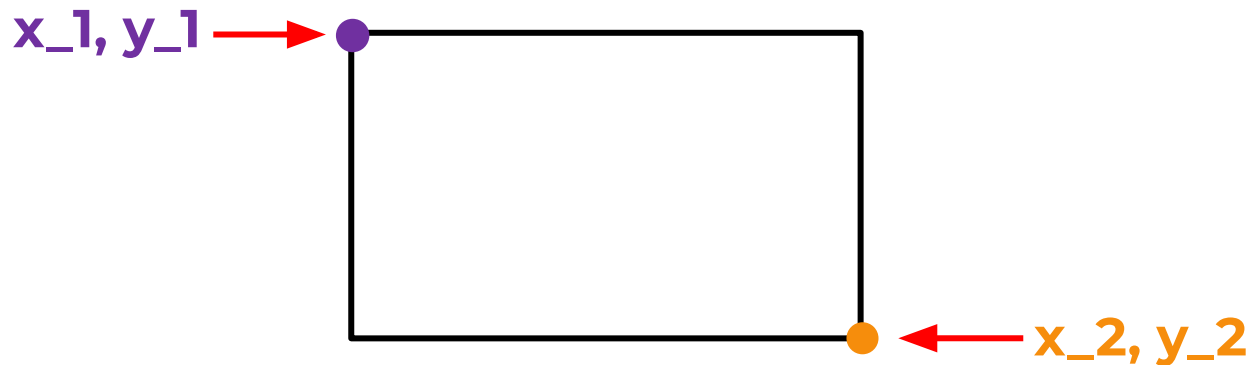


# Создание прямоугольника

---

**create\_rectangle**(x\_1, y\_1, x\_2, y\_2) – метод создаёт прямоугольник в указанных координатах.

**canvas.create\_rectangle**(x\_1, y\_1, x\_2, y\_2)



# Задание

---

Добавьте в программу создание прямоугольника:  $x_1=0$ ,  $y_1=0$ ,  $x_2=100$ ,  $y_2=100$ .

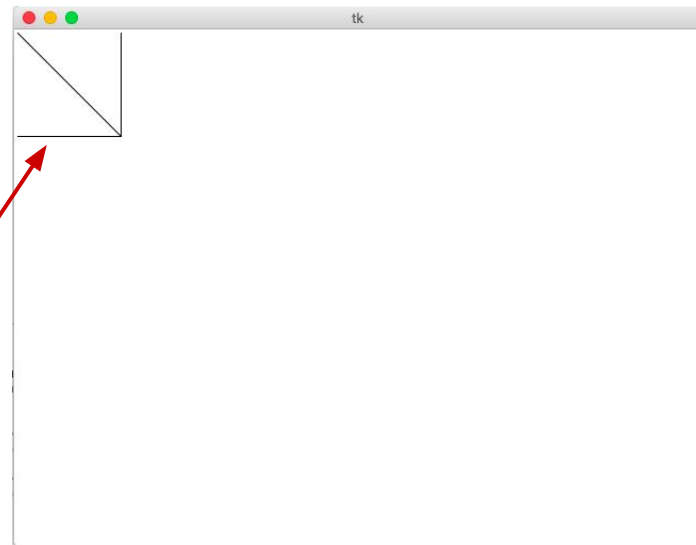
```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
canvas.pack()  
canvas.create_line(0,0,100,100)  
← -----  
root.mainloop()
```

# Задание. Решение

---

Добавьте в программу создание прямоугольника:  $x_1=0$ ,  $y_1=0$ ,  $x_2=100$ ,  $y_2=100$ .

```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
canvas.pack()  
canvas.create_line(0,0,100,100)  
canvas.create_rectangle(0,0,100,100)  
root.mainloop()
```

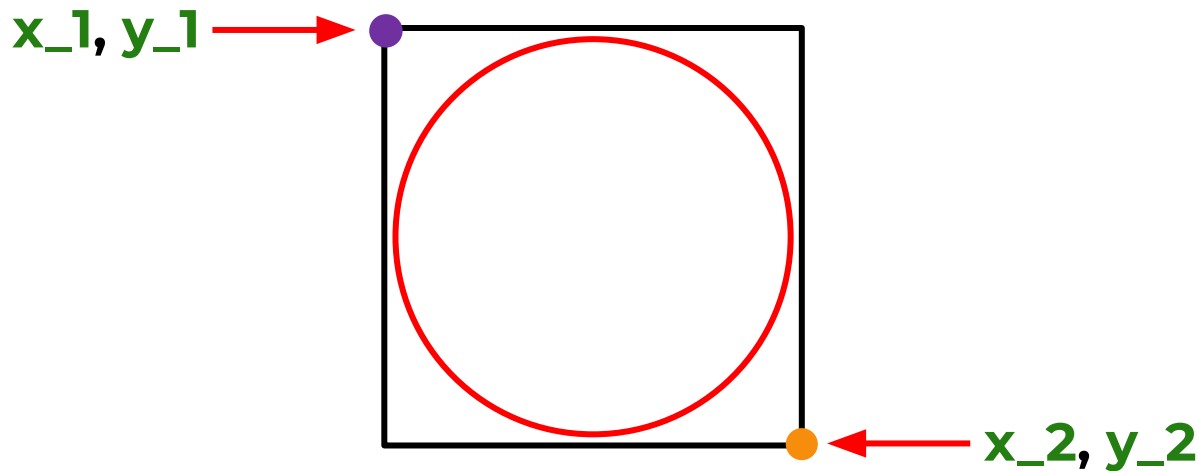


# Создание круга

---

**create\_oval**(x\_1, y\_1, x\_2, y\_2) – метод создаёт круг в указанных координатах.

**canvas.create\_oval**(x\_1, y\_1, x\_2, y\_2)





# Задание

---

Добавьте в программу создание круга:  $x_1=120$ ,  $y_1=120$ ,  $x_2=200$ ,  $y_2=200$ .

```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
canvas.pack()  
canvas.create_line(0,0,100,100)  
canvas.create_rectangle(0,0,100,100)
```



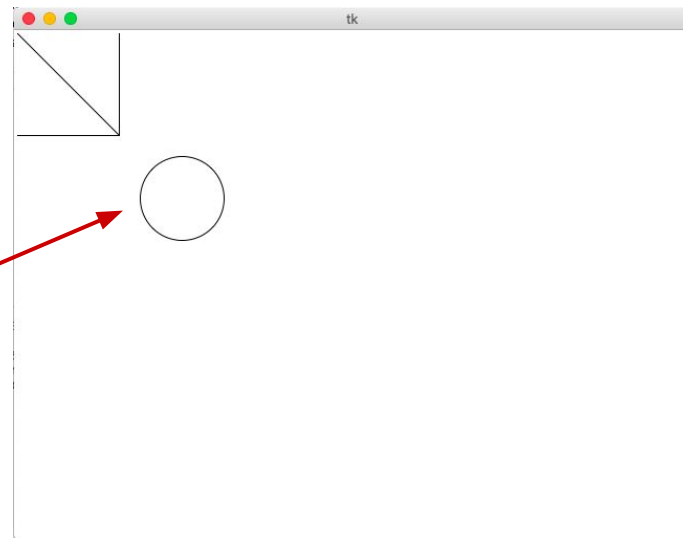
```
root.mainloop()
```

# Задание. Решение

---

Добавьте в программу создание круга:  $x_1=120$ ,  $y_1=120$ ,  $x_2=200$ ,  $y_2=200$ .

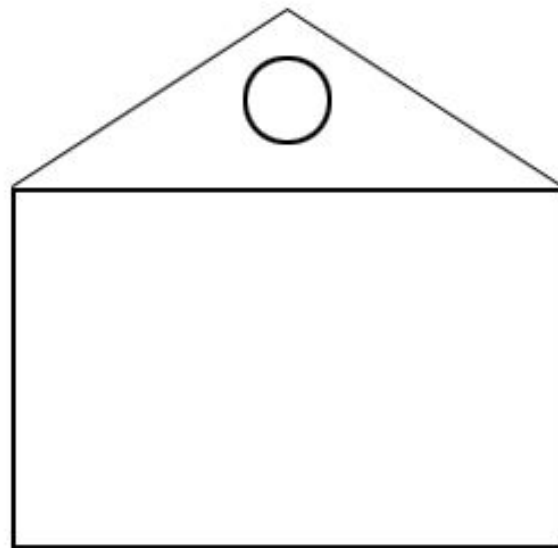
```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
canvas.pack()  
canvas.create_line(0,0,100,100)  
canvas.create_rectangle(0,0,100,100)  
canvas.create_oval(120,120,200,200)  
root.mainloop()
```



# Задание

---

Напишите программу для создания следующего изображения в указанных координатах.

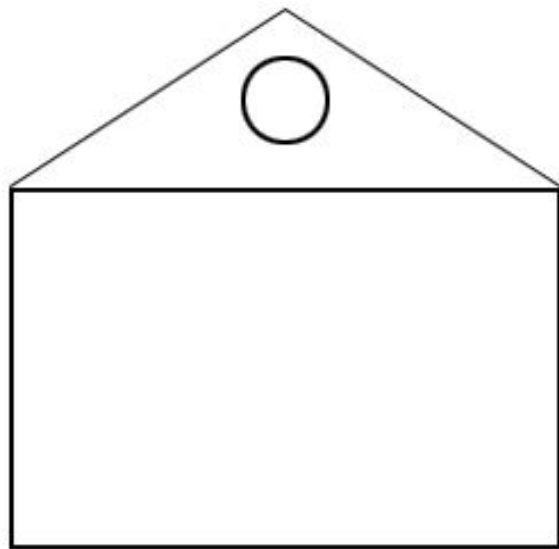


# Задание. Решение

---

Напишите программу для создания следующего изображения в указанных координатах.

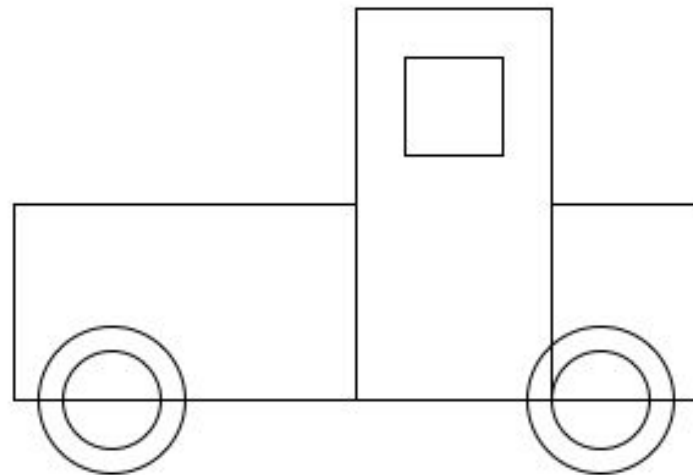
```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
canvas.pack()  
canvas.create_rectangle(40,240,280,400)  
canvas.create_oval(140,180,180,220)  
canvas.create_line(40,240,160,160)  
canvas.create_line(160,160,280,240)  
root.mainloop()
```



# Задание

---

Добавьте в программу следующее изображение в указанных координатах.



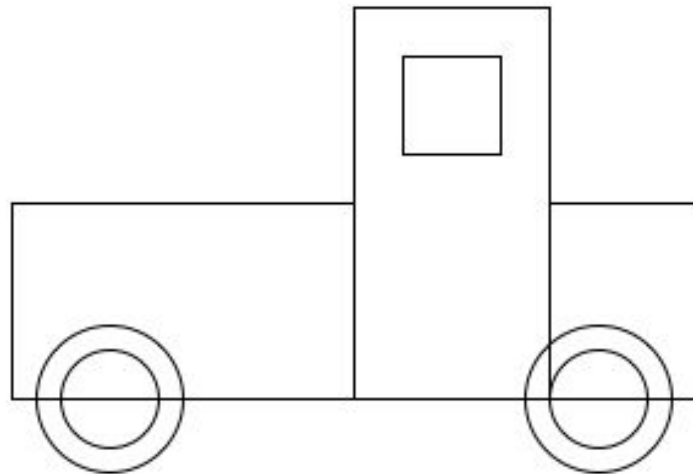
# Задание. Решение

---

Добавьте в программу следующее изображение в указанных координатах.

...

```
canvas.create_rectangle(320,320,460,400)
canvas.create_rectangle(460,240,540,400)
canvas.create_rectangle(540,320,600,400)
canvas.create_rectangle(480,260,520,300)
canvas.create_oval(340,380,380,420)
canvas.create_oval(330,370,390,430)
canvas.create_oval(540,380,580,420)
canvas.create_oval(530,370,590,430)
root.mainloop()
```



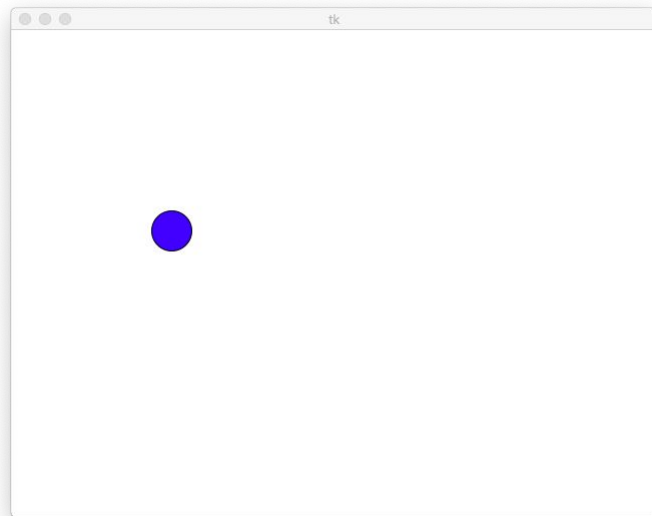
# Заливка фигуры

---

**fill** – заливка, заполняет фигуру указанным цветом.

**canvas.create\_oval(x\_1, y\_1, x\_2, y\_2, fill="color")**

```
from tkinter import*  
root=Tk()  
canvas = Canvas(root,width=640,height=480)  
canvas.pack()  
canvas.create_oval(140,180,180,220,fill="blue")  
root.mainloop()
```

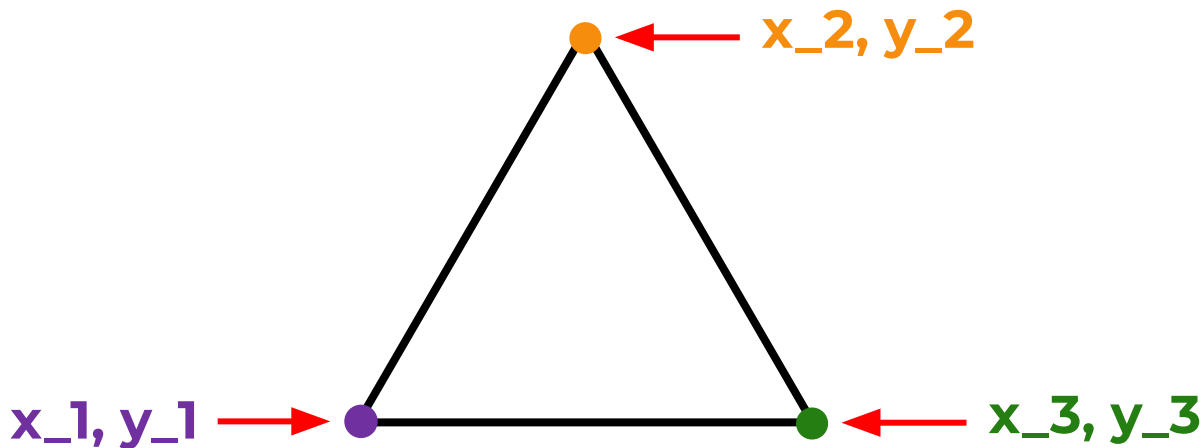


# Создание многоугольника

---

**create\_polygon**(x\_1, y\_1, x\_2, y\_2) – метод создает многоугольник в указанных координатах.

**canvas.create\_polygon**(x\_1, y\_1, x\_2, y\_2, x\_3, y\_3, ...)



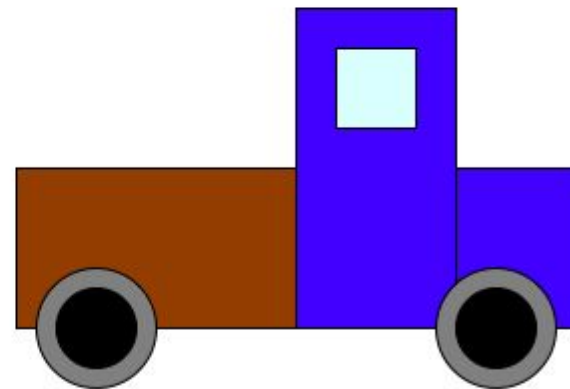
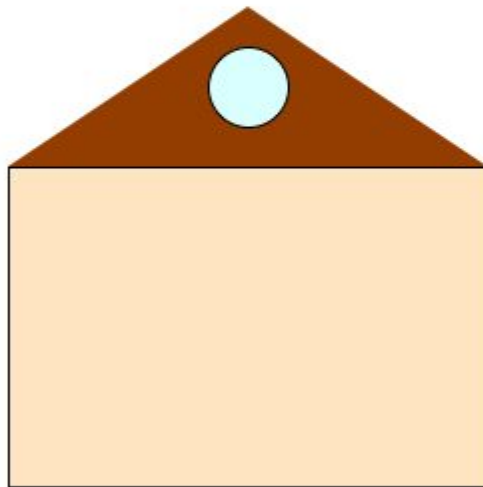


# Задание

---

Раскрасьте фигуры, как показано ниже. Для определения цвета используйте таблицу цветов.

**Обратите внимание  
на порядок создания  
элементов**

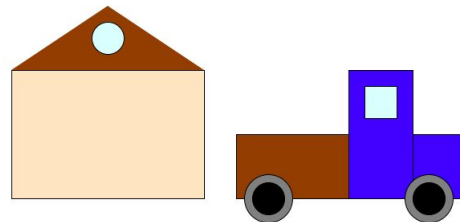


# Задание. Решение

---

```
canvas.create_rectangle(40,240,280,400,fill="bisque")  
canvas.create_polygon(40,240,160,160,280,240,fill="saddle brown")  
canvas.create_oval(140,180,180,220,fill="light cyan")
```

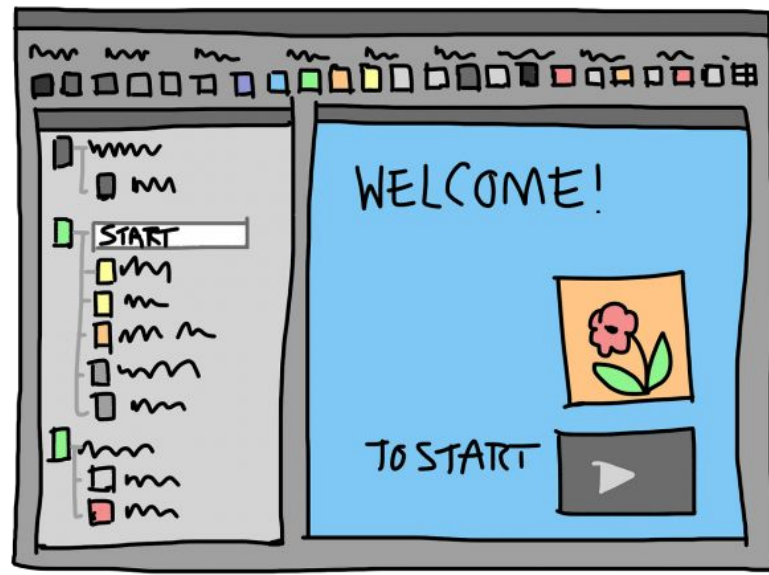
```
canvas.create_rectangle(320,320,460,400,fill="saddle brown")  
canvas.create_rectangle(460,240,540,400,fill="blue")  
canvas.create_rectangle(540,320,600,400,fill="blue")  
canvas.create_rectangle(480,260,520,300,fill="light cyan")  
canvas.create_oval(330,370,390,430,fill="gray")  
canvas.create_oval(340,380,380,420,fill="black")  
canvas.create_oval(530,370,590,430,fill="gray")  
canvas.create_oval(540,380,580,420,fill="black")
```



# Сегодня на уроке

---

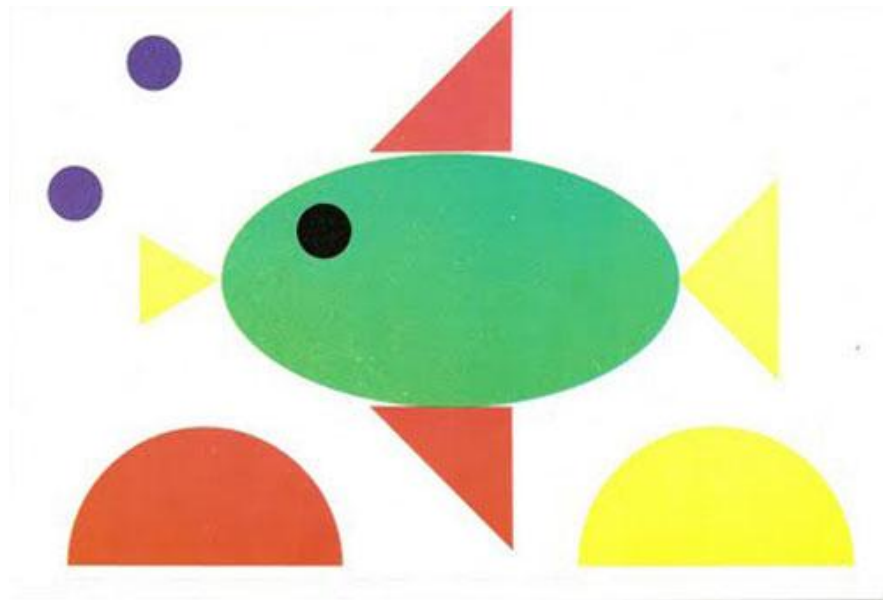
- Модуль Tkinter
- Классы и объекты
- Работа с окном и Canvas
- Быстрое появление фигур



# Задание на дом. Уровень 1

---

Напишите программу для получения следующего изображения.



# Задание на дом. Уровень 2

---

Напишите программу для  
получения следующего  
изображения

