

Графика в Python

В этом уроке мы разберём графику в Python с помощью модуля **Turtle черепашка**. **Turtle** это модуль для Python, позволяющий создавать графические объекты, рисунки в специальном окне.

Модуль **Turtle** можно использовать для создания игр на питоне.

Чтобы присоединить (импортировать) модуль **turtle** необходимо выбрать один из следующих способов:

1. **import turtle**
2. **from turtle import ***
3. **from turtle import open as t**

Первый способ импортирует модуль, но команды нужно писать с упоминанием модуля (как ссылку на атрибут) в стиле ООП. Например, **turtle.reset()**, что загромождает исходный код и создаёт неудобочитаемость программы.

Второй способ позволяет обращаться к функциям модуля непосредственно без упоминания имени модуля.

Третий способ позволяет подменить название модуля своим именем, т. е. вместо того, чтобы писать полностью имя модуля мы будем использовать только один символ, например:

```
t.reset()
t.fd(100)
```

К особенностям работы с модулем **turtle** относится предварительный вызов графического окружения — так называемого **холста**, т.е. поля по которому перемещается исполнитель. Для этого используется команда без параметров **reset()**. Она же сбрасывает все настройки по умолчанию и возвращает исполнителя в исходное состояние. (Рис. 1).

Внешний вид исполнителя (по умолчанию стиль **classic** черного цвета) — наконечник стрелы всегда направленный в сторону движения.

Существует несколько команд определяющих внешний вид исполнителя:

1. **turtle.shape(«Style»)** - изменяет внешний вид
2. **turtle.shapesize(m,n)** — управляет размерами исполнителя
3. **turtle.tilt(alpha)** — изменяет ориентацию исполнителя
4. **turtle.hideturtle()** - скрывает черепаху
5. **turtle.showturtle()** - показывает черепаху и др.

Параметр **«Style»** в функции **shape()** изменяет внешний вид исполнителя. Поддерживаются следующие стили:

- **arrow**
- **turtle**
- **circle**
- **square**
- **triangle**
- **classic**

Чтобы программа с модулем **turtle** на Python работала корректно, в самом конце программы всегда нужно прописывать две команды:

1. **t.screen.exitonclick()**
2. **t.screen.mainloop()**

С помощью команды **t.screen.exitonclick()** программа на Python реагирует на нажатие кнопки мыши после исполнения программы. Если пользователь нажмёт на левую кнопку мыши, пока курсор находится в окне для графики модуля **turtle**, то окно закроется.

С помощью команды **t.screen.mainloop()** останавливает выполнение программы.

Запустив программу, вы увидите окно для графики с «**черепашкой**» по центру (рис.1):



Рис.1.

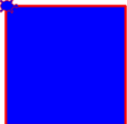
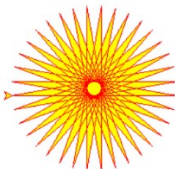
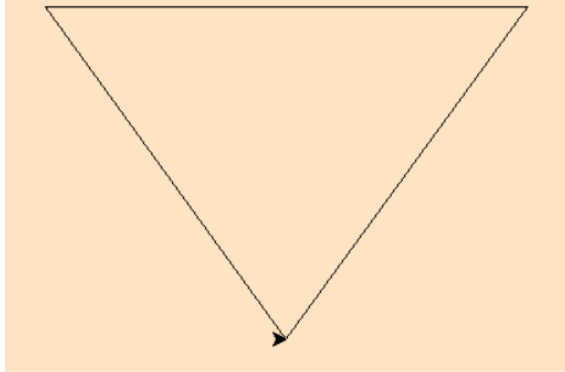
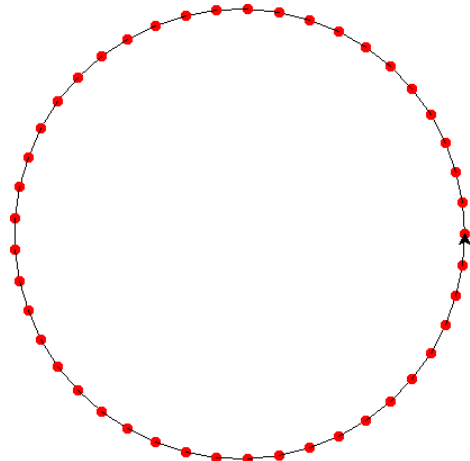
Начало координат в окне для графики модуля **turtle** находится в центре окна. Положительное направление оси X определяется слева направо, положительное направление оси Y определяется снизу вверх, чем больше X, тем правее черепашка, чем больше Y, тем выше черепашка.

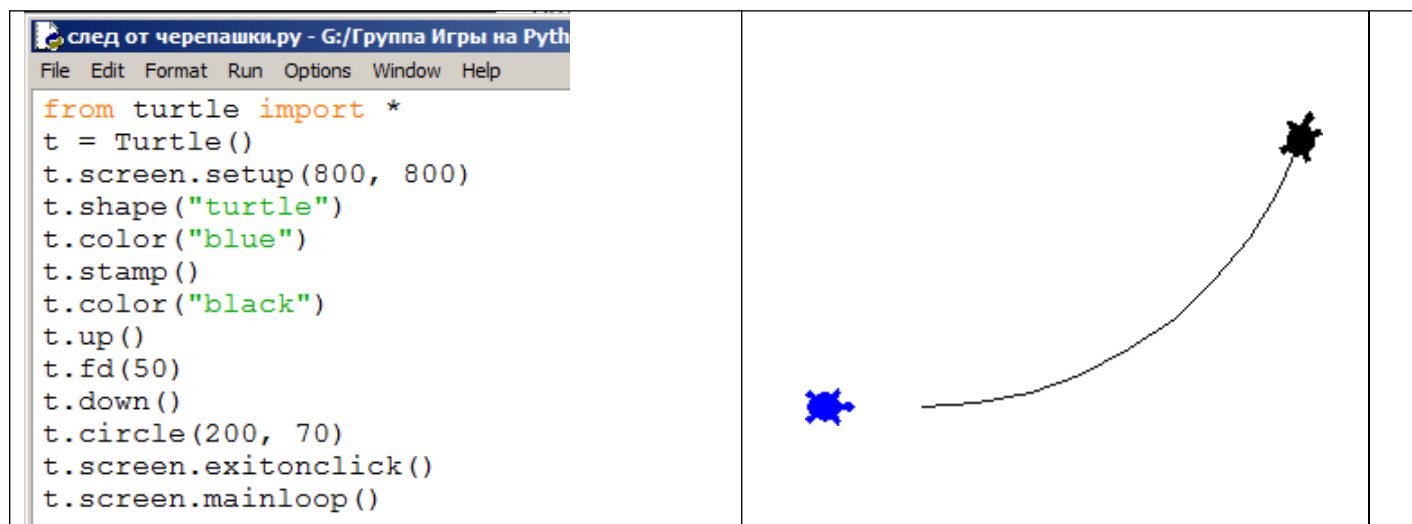
Рисунки на экране появляются с помощью перемещения «черепашки» в окне для графики модуля **turtle**, черепашка рисует линию.

Чтобы черепашка в окне для графики модуля **turtle** в Питоне двигалась вперёд, используется команда **t.fd(x)**, где x – количество пикселей, на которое сдвигается черепашка. Для движения назад используется команда **t.bk(x)**. Для передвижения черепашки в заданную точку использовать координаты **t.goto(x, y)**, где x и y – координаты точки, в которую должна переместиться черепашка.

Примеры программ:

Текст программы	Рисунок
<p>пунктир.py - G:/Группа Игры на Python/Черепашка</p> <pre> File Edit Format Run Options Window Help from turtle import * t = Turtle() t.screen.setup(800, 800) for i in range(20): t.fd(8) t.up() t.fd(8) t.down() t.screen.exitonclick() t.screen.mainloop() </pre>	
<p>gr12.py - F:/Питон/Питон_уроки Полякова/gr12.py (3.7.0)</p> <pre> File Edit Format Run Options Window Help import turtle # Подключаем модуль turtle turtle.reset() # Приводим черепашку в начальное положение turtle.down() # Опускаем перо (начало рисования) turtle.forward(150) # Проползти 20 пикселей вперед turtle.left(90) # Поворот влево на 90 градусов turtle.forward(150) # Рисуем вторую сторону квадрата turtle.left(90) turtle.forward(150) # Рисуем третью сторону квадрата turtle.left(90) turtle.forward(150) # Рисуем четвертую сторону квадрата turtle.up() # Поднять перо (закончить рисовать) turtle.forward(100) # Отвести черепашку от рисунка в сторону turtle._root.mainloop() # Задержать окно на экране </pre>	

<div data-bbox="140 208 727 241">gr2.py - F:\Питон\Питон_уроки Полякова\gr2.py (3.7.0)</div> <div data-bbox="140 257 641 284">File Edit Format Run Options Window Help</div> <pre data-bbox="140 293 430 611">import turtle from turtle import * reset() shape("turtle") color("red", "blue") pensize(3) begin_fill() for i in range(4): fd(150) rt(90) end_fill()</pre>	<div data-bbox="874 208 983 224">Python Turtle Graphics</div> 
<div data-bbox="140 651 662 678">gr3.py - F:\Питон\Питон_уроки Полякова\gr3.py (3.7.0)</div> <div data-bbox="140 694 585 719">File Edit Format Run Options Window Help</div> <pre data-bbox="140 728 419 983">from turtle import * color('red', 'yellow') begin_fill() while True: forward(200) left(170) if abs(pos()) < 1: break end_fill() done()</pre>	<div data-bbox="874 633 983 649">Python Turtle Graphics</div> 
<div data-bbox="140 1052 751 1077">z10_треугольник.py - F:\Группа Игры на Python\Черепашка\z10_треугольник.py (3.7.0)</div> <div data-bbox="140 1086 499 1106">File Edit Format Run Options Window Help</div> <pre data-bbox="140 1111 702 1458">from turtle import * iter=4 # Количество итераций screensize(1200,1200,'Bisque') reset() # сброс настроек черепашки и очистка окна setworldcoordinates(-50,-50,1150,1150) speed(1) # устанавливаем самую медленную скорость пера # Процедура строит тр-к по координатам вершин def tr(x1,y1,x2,y2,x3,y3): penup() goto(x1,y1) pendown() goto(x2,y2) goto(x3,y3) goto(x1,y1) down() tr(320,10,600,470,40,470)</pre>	
<div data-bbox="140 1500 761 1525">окружность с точками.py - G:\Группа Игры на Python\Черепашка\окружность с точками.py (3.7.0)</div> <div data-bbox="140 1532 587 1554">File Edit Format Run Options Window Help</div> <pre data-bbox="140 1563 678 1991">from turtle import * t = Turtle() t.screen.setup(800, 800) def circ(d, r, rBig): for i in range(d): t.circle(rBig, 360 / d) t.dot(r, "red") t.up() t.goto(350, 0) t.setheading(90) t.down() circ(45, 10, 200) t.screen.exitonclick() t.screen.mainloop()</pre>	



Рассмотрите, пожалуйста, рисунки и постарайтесь понять, что означает каждая команда.

Замечание: Оператор **for i in range (4)** означает, что цикл выполнится 4 раза.

Ниже приведен список основных команд модуля **turtle**.

Список основных команд черепашки

Команды перемещения черепашки	
forward(n) / fd(n)	Проползти вперед n шагов (пикселей)
backward(n)	Проползти назад n шагов (пикселей)
left(angle)	Повернуться налево на angle градусов
right(angle)	Повернуться направо на angle градусов
circle(r)	Нарисовать окружность радиуса r , центр которой находится слева от черепашки, если r>0 и справа, если r<0
circle(r, angle)	Нарисовать дугу радиуса r и градусной мерой angle. Дуга рисуется против часовой стрелки, если r>0 и по часовой стрелке, если r<0
goto(x, y)	Переместить черепашку в точку с координатами (x,y)
Команды рисования	
down()	Опустить перо. После этой команды черепашка начнет оставлять след при любом своем передвижении
up()	Поднять перо
width(n)	Установить ширину следа черепашки в n пикселей
color(s)	Установить цвет следа черепашки в s. s должно быть текстовой строкой, заключенной в кавычки, с названием цвета (по-английски), например: "red", "yellow", "green" и т.д.
Bgcolor(s)	Устанавливает цвет фона в s. s должно быть текстовой строкой, заключенной в кавычки, с названием цвета (по-английски), например: "red", "yellow", "green" и т.д.
fill(f)	Используется для рисования закрашенных областей. Начиная рисовать закрашенную область, дайте команду turtle.fill(1), а закончив рисование области — turtle.fill(0)
stamp()	После выполнения этой команды в окне для графики в месте, на котором была черепашка, останется рисунок этой черепашки
setheading(x)	где x – угол поворота в градусах относительно начального положения

	черепашки при запуске программы
Прочие команды	
reset()	Возврат черепашки в исходное состояние: очищается экран, сбрасываются все параметры, черепашка устанавливается в начало координат, глядя вправо
write(s)	Вывести текстовую строку s в точке нахождения черепашки
dot(r, color)	Команда ставит точку, где r – радиус точки и color – цвет точки
radians()	Установить меру измерения углов (во всех командах черепашки) в радианы
degrees()	Установить меру измерения углов (во всех командах черепашки) в градусы. Этот режим включен по умолчанию
tracer(f)	Включить режим отладки (трассировки) программы черепашки, если значение f равно 1. Если значение f равно 0, отключить режим отладки. В режиме отладки черепашка перемещается медленнее и на экране нарисована сама черепашка. По умолчанию режим отладки включен

Вот еще несколько примеров:

1. Программа рисует оси координат (смотри рис.2).

```
# -*- coding: utf-8 -*-
import turtle
#
turtle.reset()
turtle.tracer(0)
turtle.color('#0000ff')
#
turtle.write('0,0')
#
turtle.up()
x=-170
y=-120
coords=str(x)+", "+str(y)
turtle.goto(x, y)
turtle.write(coords)
#
x=130
y=100
coords=str(x)+", "+str(y)
turtle.goto(x, y)
turtle.write(coords)
#
x=0
y=-100
coords=str(x)+", "+str(y)
turtle.goto(x, y)
turtle.write(coords)
#
turtle.down()
x=0
y=100
coords=str(x)+", "+str(y)
turtle.goto(x, y)
turtle.write(coords)
#
turtle.up()
x=-150
y=0
```

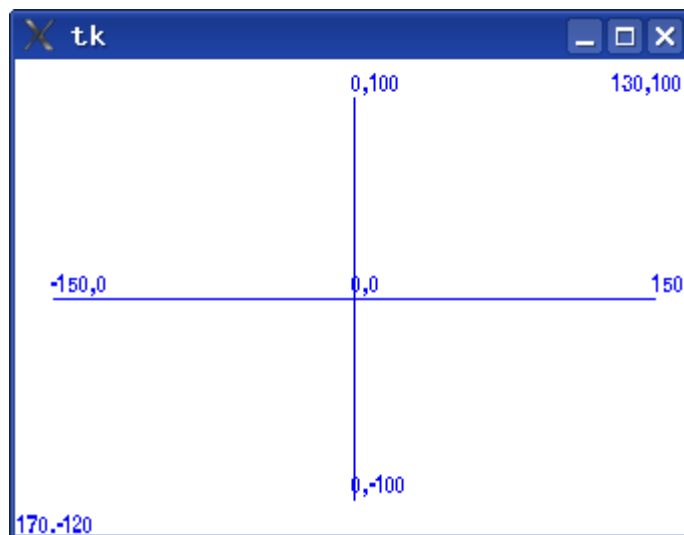


Рис.2

```

coords=str(x)+","+str(y)
turtle.goto(x, y)
turtle.write(coords)
#
turtle.down()
x=150
y=0
coords=str(x)+","+str(y)

turtle.goto(x, y)
turtle.write(coords)
turtle.up()
#
turtle.mainloop()

```

2. Программа рисует смайлик (рис.3)

```

# -*- coding: utf-8 -*-
import turtle
from turtle import *
#
reset()
tracer(0)
width(2)
#
up()
x=0
y=-100
goto(x, y)
begin_fill()
color('#ffaa00')
down()
circle(100)
end_fill()
color('black')
circle(100)
up()
#
x=-45
y=50
goto(x, y)
down()
color('#0000aa')
begin_fill()
circle(7)
up()
end_fill()
#
x=45
y=50
goto(x, y)
down()
color('#0000aa')
begin_fill()
circle(7)
up()
end_fill()
#

```

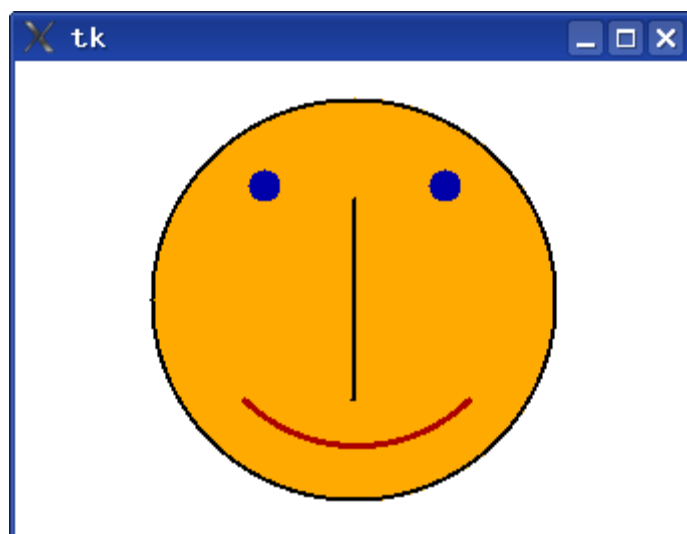


Рис. 3.

```

x=-55
y=-50
goto(x, y)
right(45)
width(3)
down()
color('#aa0000')

circle(80, 90)
up()
#
x=0
y=50
goto(x, y)
right(135)
width(3)
down()
color('#000000')
forward(100)
up()
#
turtle.mainloop()

```

Самостоятельно изучите приведенные ниже примеры

Пример 1. Полет черепахи:

```

from turtle import *
color("orange")
dot(10)
center = pos()
color("blue")
shape("turtle")
speed(0)
penup()
goto(200, 0)
pendown()
G = 800
v = Vec2D(0, 1)
t = 0
dt = 1
while t < 1100:
    goto(pos() + v * dt)
    setheading(towards(center))
    r = distance(center)
    acc = (-G / r ** 3) * pos()
    v += acc * dt
    t += dt
mainloop()

```

Пример 2 Радиоактивный знак:

```

from turtle import *

def square(length):
    for i in range(4):
        forward(length)

```

```

        left(90)

def sector(radius, angle):
    forward(radius)
    left(90)
    circle(radius, angle)
    left(90)
    forward(radius)
    left(180-angle)

def move(x, y):
    up()
    forward(x)
    left(90)
    forward(y)
    right(90)
    down()

def radioactive(radius1, radius2, side, angle=60, outlinecol="black",
fillcol="yellow"):
    color(outlinecol)
    move(-(side/2), -(side/2))

    begin_fill()
    square(side)
    color(fillcol)
    end_fill()
    move((side/2), (side/2))
    color(outlinecol)
    right(90 + angle/2)

    for i in range(3):
        begin_fill()
        sector(radius1,angle)
        left(120)
        color(outlinecol)
        end_fill()

    up()
    forward(radius2)
    left(90)
    down()

    color(fillcol)
    begin_fill()
    circle(radius2)
    color(outlinecol)
    end_fill()

    up()
    left(90)
    forward(radius2)
    width(1)

reset()
width(5)

```



```
speed(1)
radioactive(160, 36, 400)
mainloop()
```

Пример 3 Пример разработчиков:

```
from turtle import *
def switchpen():
    if isdown():
        pu()
    else:
        pd()

def demo2():
    """Demo of some new features."""
    speed(1)
    st()
    pensize(3)
    setheading(towards(0, 0))
    radius = distance(0, 0)/2.0
    rt(90)
    for _ in range(18):
        switchpen()
        circle(radius, 10)
    write("wait a moment...")
    while undobufferentries():
        undo()
    reset()
    lt(90)
    colormode(255)
    laenge = 10
    pencolor("green")
    pensize(3)
    lt(180)
    for i in range(-2, 16):
        if i > 0:
            begin_fill()
            fillcolor(255-15*i, 0, 15*i)
            for _ in range(3):
                fd(laenge)
                lt(120)
            end_fill()
            laenge += 10
            lt(15)
            speed((speed()+1)%12)
        #end_fill()

    lt(120)
    pu()
    fd(70)
    rt(30)
    pd()
    color("red", "yellow")
    speed(0)
    begin_fill()
```

```
for _ in range(4):  
    circle(50, 90)  
    rt(90)
```

```

        fd(30)
        rt(90)
    end_fill()
    lt(90)
    pu()
    fd(30)
    pd()
    shape("turtle")

    tri = getturtle()
    tri.resizemode("auto")
    turtle = Turtle()
    turtle.resizemode("auto")
    turtle.shape("turtle")
    turtle.reset()
    turtle.left(90)
    turtle.speed(0)
    turtle.up()
    turtle.goto(280, 40)
    turtle.lt(30)
    turtle.down()
    turtle.speed(6)
    turtle.color("blue", "orange")
    turtle.pensize(2)
    tri.speed(6)
    setheading(towards(turtle))
    count = 1
    while tri.distance(turtle) > 4:
        turtle.fd(3.5)
        turtle.lt(0.6)
        tri.setheading(tri.towards(turtle))
        tri.fd(4)
        if count % 20 == 0:
            turtle.stamp()
            tri.stamp()
            switchpen()
        count += 1
    tri.write("CAUGHT! ", font=("Arial", 16, "bold"), align="right")
    tri.pencolor("black")
    tri.pencolor("red")

    def baba(xdummy, ydummy):
        clearscreen()
        bye()

    while undobufferentries():
        tri.undo()
        turtle.undo()
    tri.fd(50)
    tri.write(" Click me!", font = ("Courier", 12, "bold") )
    tri.onclick(baba, 1)

```

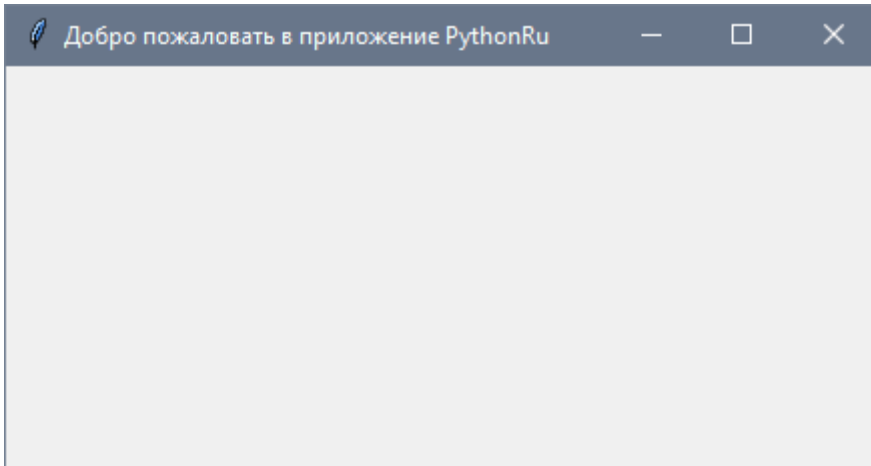
demo2()

Создание графического интерфейса

Для начала, следует импортировать Tkinter и создать окно, в котором мы зададим его название:

```
from tkinter import *  
window = Tk()  
window.title("Добро пожаловать в приложение PythonRu")  
window.mainloop()
```

Результат будет выглядеть следующим образом:



Прекрасно! Наше приложение работает.

Последняя строка вызывает функцию `mainloop`. Эта функция вызывает бесконечный цикл окна, поэтому окно будет ждать любого взаимодействия с пользователем, пока не будет закрыто.

В случае, если вы забудете вызвать функцию `mainloop`, для пользователя ничего не отобразится.

Создание виджета Label

Чтобы добавить текст в наш предыдущий пример, мы создадим `lbl`, с помощью класса `Label`, например:

```
lbl = Label(window, text="Привет")
```

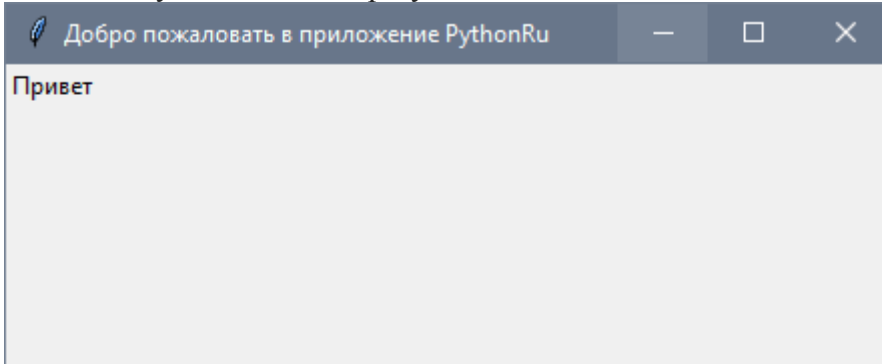
Затем мы установим позицию в окне с помощью функции `grid` и укажем ее следующим образом:

```
lbl.grid(column=0, row=0)
```

Полный код, будет выглядеть следующим образом:

```
from tkinter import *  
  
window = Tk()  
window.title("Добро пожаловать в приложение PythonRu")  
lbl = Label(window, text="Привет")  
lbl.grid(column=0, row=0)  
window.mainloop()
```

И вот как будет выглядеть результат:

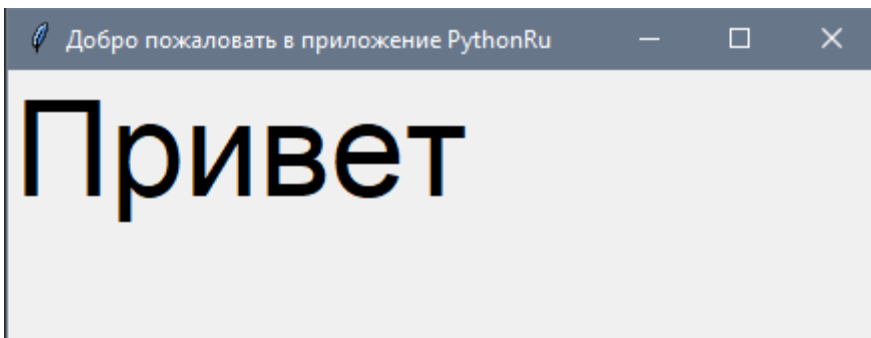


Если функция `grid` не будет вызвана, текст не будет отображаться.

Настройка размера и шрифта текста

Вы можете задать шрифт текста и размер. Также можно изменить стиль шрифта. Для этого передайте параметр `font` таким образом:

```
lbl = Label(window, text="Привет", font=("Arial Bold", 50))
```



Обратите внимание, что параметр `font` может быть передан любому виджету, для того, чтобы поменять его шрифт, он применяется не только к `Label`.

Отлично, но стандартное окно слишком мало. Как насчет настройки размера окна?

Настройка размеров окна приложения

Мы можем установить размер окна по умолчанию, используя функцию `geometry` следующим образом:

```
window.geometry('400x250')
```

В приведенной выше строке устанавливается окно шириной до 400 пикселей и высотой до 250 пикселей.

Попробуем добавить больше виджетов GUI, например, кнопки и посмотреть, как обрабатывается нажатие кнопок.

Добавление виджета Button

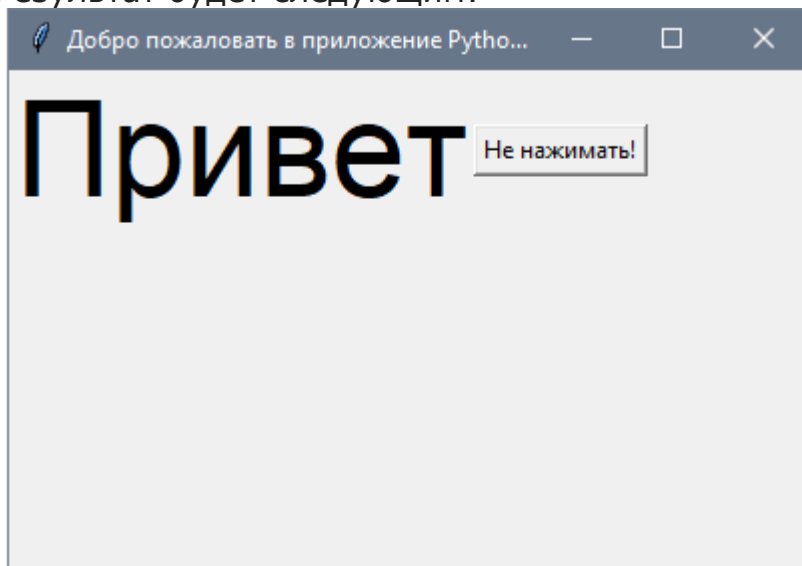
Начнем с добавления кнопки в окно. Кнопка создается и добавляется в окно так же, как и метка:

```
btn = Button(window, text="Не нажимать!")  
btn.grid(column=1, row=0)
```

Наш код будет выглядеть вот так:

```
from tkinter import *  
window = Tk()  
window.title("Добро пожаловать в приложение PythonRu")  
window.geometry('400x250')  
lbl = Label(window, text="Привет", font=("Arial Bold", 50))  
lbl.grid(column=0, row=0)  
btn = Button(window, text="Не нажимать!")  
btn.grid(column=1, row=0)  
window.mainloop()
```

Результат будет следующим:

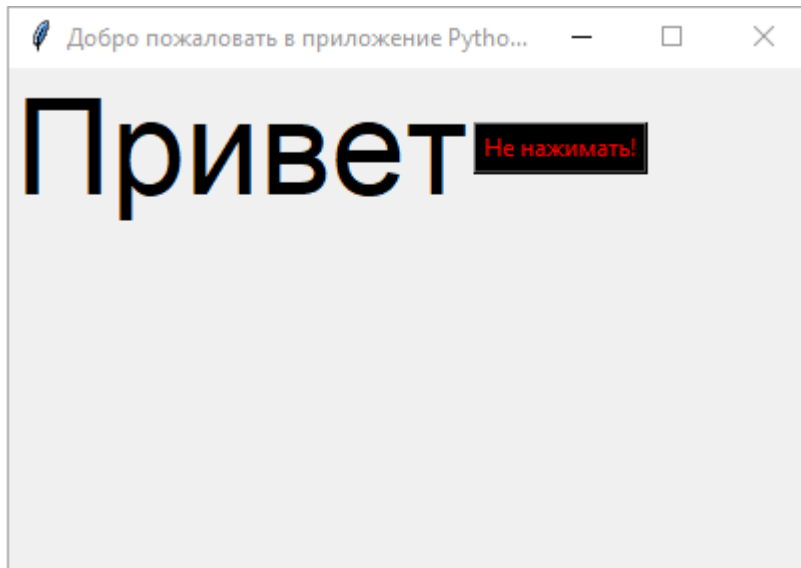


Обратите внимание, что мы помещаем кнопку во второй столбец окна, что равно 1. Если вы забудете и поместите кнопку в том же столбце, который равен 0, он покажет только кнопку.

Изменение цвета текста и фона у Button

Вы можете поменять цвет текста кнопки или любого другого виджета, используя свойство `fg`. Кроме того, вы можете поменять цвет фона любого виджета, используя свойство `bg`.

```
btn = Button(window, text="Не нажимать!", bg="black", fg="red")
```



Теперь, если вы попытаетесь щелкнуть по кнопке, ничего не произойдет, потому что событие нажатия кнопки еще не написано.

Кнопка Click

Для начала, мы запишем функцию, которую нужно выполнить при нажатии кнопки:

```
def clicked():  
    lbl.configure(text="Я же просил...")
```

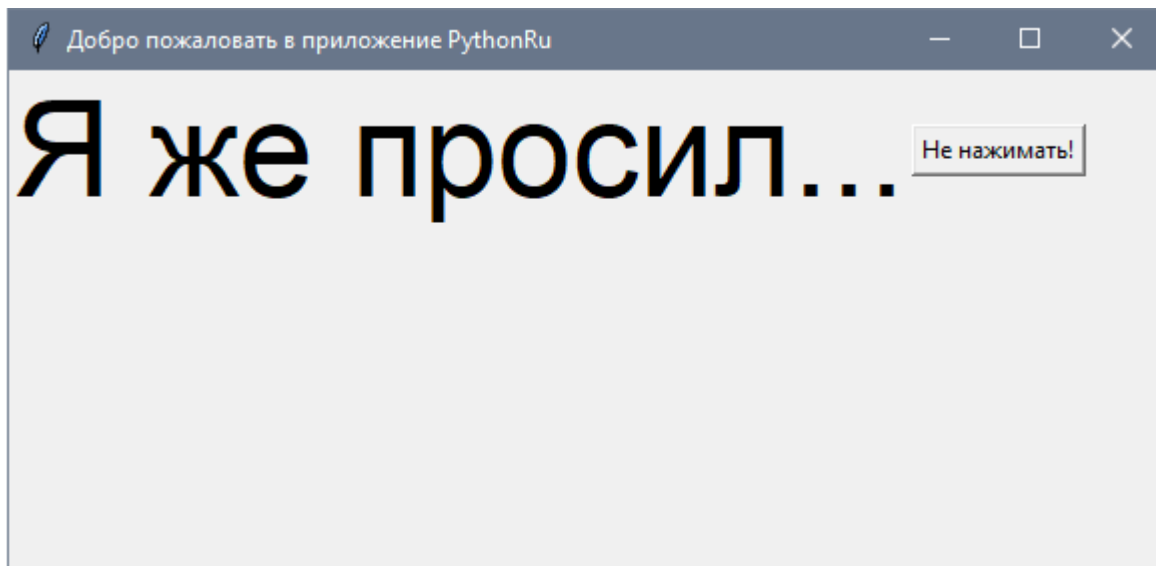
Затем мы подключим ее с помощью кнопки, указав следующую функцию:

```
btn = Button(window, text="Не нажимать!", command=clicked)
```

Обратите внимание: мы пишем `clicked`, а не `clicked()` с круглыми скобками. Теперь полный код будет выглядеть так:

```
from tkinter import *  
def clicked():  
    lbl.configure(text="Я же просил...")  
window = Tk()  
window.title("Добро пожаловать в приложение PythonRu")  
window.geometry('400x250')  
lbl = Label(window, text="Привет", font=("Arial Bold", 50))  
lbl.grid(column=0, row=0)  
btn = Button(window, text="Не нажимать!", command=clicked)  
btn.grid(column=1, row=0)  
window.mainloop()
```

При нажатии на кнопку, результат, как и ожидалось, будет выглядеть следующим образом:



Получение ввода с использованием класса Entry (текстовое поле Tkinter)

В предыдущих примерах GUI Python мы ознакомились со способами добавления простых виджетов, а теперь попробуем получить пользовательский ввод, используя класс Tkinter Entry (текстовое поле Tkinter).

Вы можете создать текстовое поле с помощью класса Tkinter Entry следующим образом:

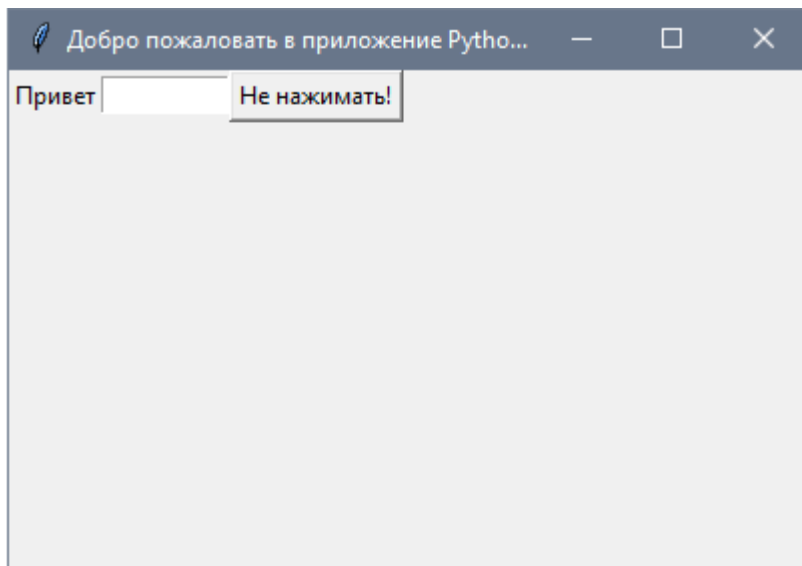
```
txt = Entry(window, width=10)
```

Затем вы можете добавить его в окно, используя функцию grid.

Наше окно будет выглядеть так:

```
from tkinter import *
def clicked():
    lbl.configure(text="Я же просил...")
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
lbl = Label(window, text="Привет")
lbl.grid(column=0, row=0)
txt = Entry(window, width=10)
txt.grid(column=1, row=0)
btn = Button(window, text="Не нажимать!", command=clicked)
btn.grid(column=2, row=0)
window.mainloop()
```

Полученный результат будет выглядеть так:



Теперь, если вы нажмете кнопку, она покажет то же самое старое сообщение, но что же будет с отображением введенного текста в виджет `Entry`?

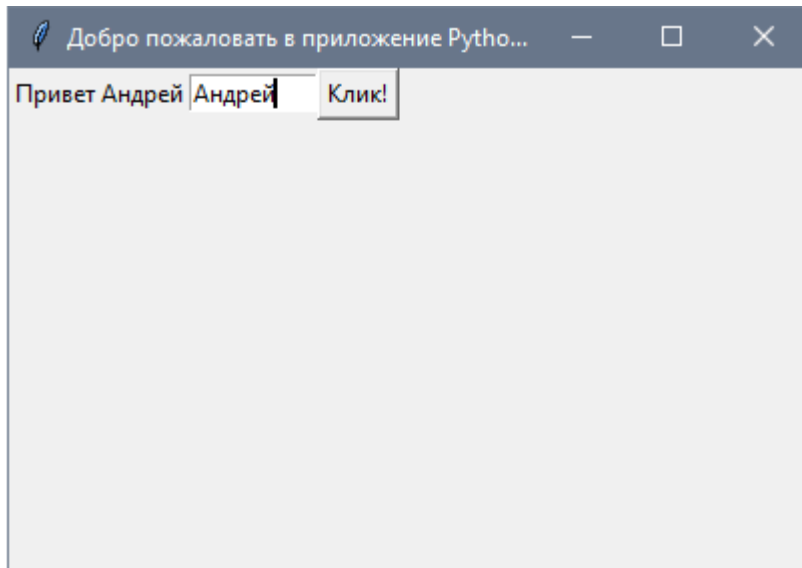
Во-первых, вы можете получить текст ввода, используя функцию `get`. Мы можем записать код для выбранной функции таким образом:

```
def clicked():  
    res = "Привет {}".format(txt.get())  
  
    lbl.configure(text=res)
```

Если вы нажмете на кнопку — появится текст «Привет » вместе с введенным текстом в виджете записи. Вот полный код:

```
from tkinter import *  
def clicked():  
    res = "Привет {}".format(txt.get())  
    lbl.configure(text=res)  
window = Tk()  
window.title("Добро пожаловать в приложение PythonRu")  
window.geometry('400x250')  
lbl = Label(window, text="Привет")  
lbl.grid(column=0, row=0)  
txt = Entry(window,width=10)  
txt.grid(column=1, row=0)  
btn = Button(window, text="Клик!", command=clicked)  
btn.grid(column=2, row=0)  
window.mainloop()
```

Запустите вышеуказанный код и проверьте результат:



Каждый раз, когда мы запускаем код, нам нужно нажать на виджет ввода, чтобы настроить фокус на ввод текста, но как насчет автоматической настройки фокуса?

Установка фокуса виджета ввода

Здесь все очень просто, ведь все, что нам нужно сделать, — это вызвать функцию `focus`:

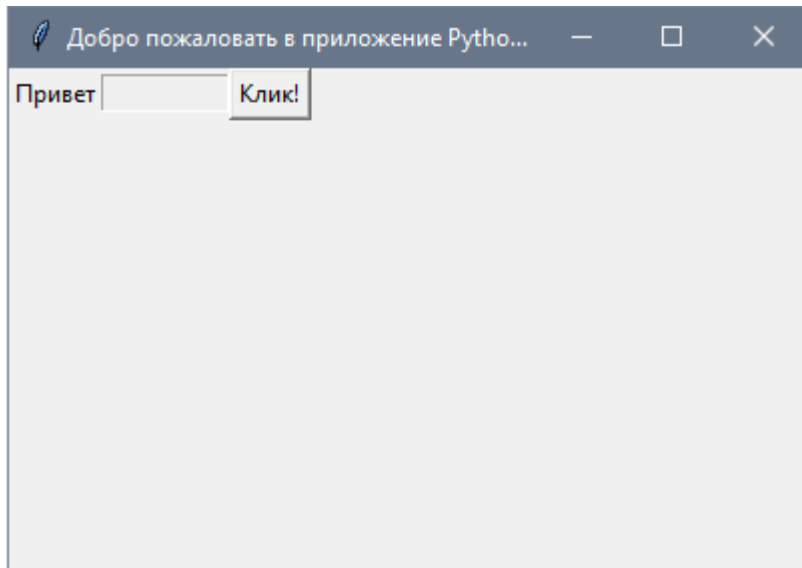
```
txt.focus()
```

Когда вы запустите свой код, вы заметите, что виджет ввода в фокусе, который дает возможность сразу написать текст.

Отключить виджет ввода

Чтобы отключить виджет ввода, отключите свойство состояния:

```
txt = Entry(window,width=10, state='disabled')
```



Теперь вы не сможете ввести какой-либо текст.

Добавление виджета Combobox

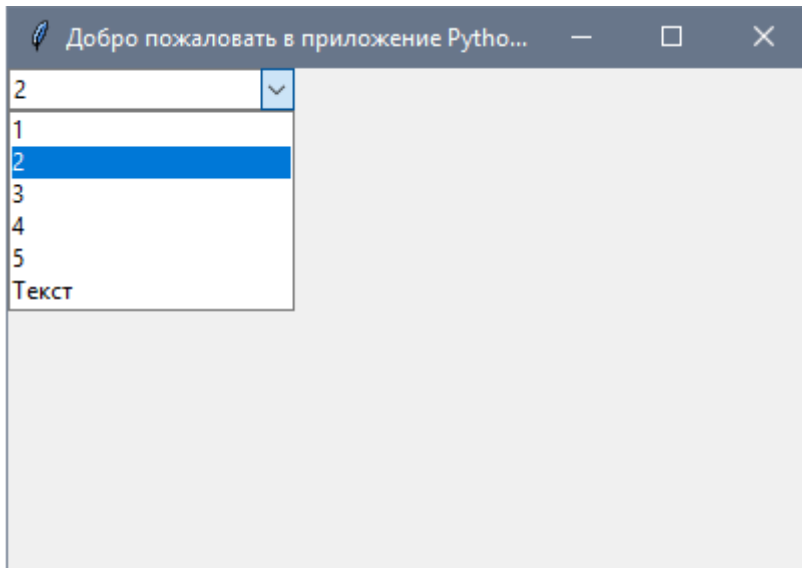
Чтобы добавить виджет поля с выпадающим списком, используйте класс `Combobox` из `ttk` следующим образом:

```
from tkinter.ttk import Combobox
```

```
combo = Combobox(window)
```

Затем добавьте свои значения в поле со списком.

```
from tkinter import *
from tkinter.ttk import Combobox
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
combo = Combobox(window)
combo['values'] = (1, 2, 3, 4, 5, "Текст")
combo.current(1) # установите вариант по умолчанию
combo.grid(column=0, row=0)
window.mainloop()
```



Как видите с примера, мы добавляем элементы `combobox`, используя значения `tuple`.
Чтобы установить выбранный элемент, вы можете передать индекс нужного элемента текущей функции.
Чтобы получить элемент `select`, вы можете использовать функцию `get` вот таким образом:

```
combo.get()
```

Добавление виджета `Checkbutton` (чекбокса)

С целью создания виджета `checkbutton`, используйте класс `Checkbutton`:

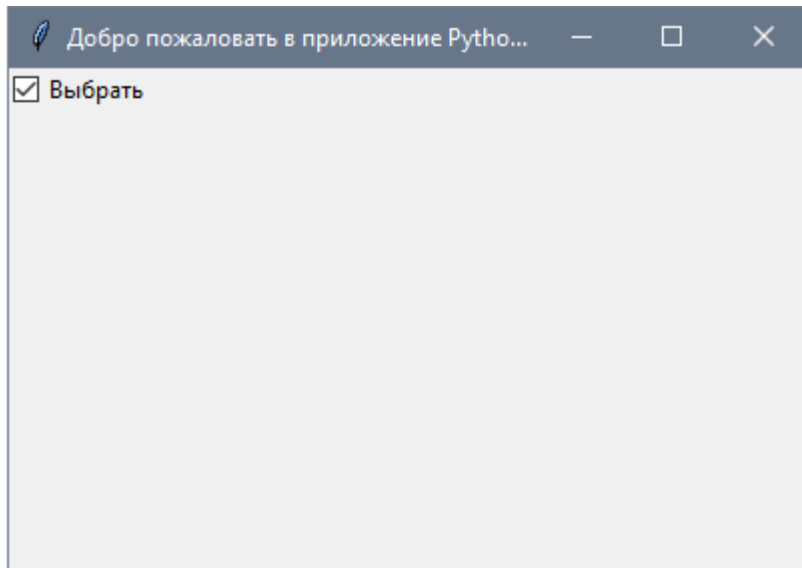
```
from tkinter.ttk import Checkbutton
```

```
chk = Checkbutton(window, text='Выбрать')
```

Кроме того, вы можете задать значение по умолчанию, передав его в параметр `var` в `Checkbutton`:

```
from tkinter import *
from tkinter.ttk import Checkbutton
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
chk_state = BooleanVar()
chk_state.set(True) # задайте проверку состояния чекбокса
chk = Checkbutton(window, text='Выбрать', var=chk_state)
chk.grid(column=0, row=0)
window.mainloop()
```

Посмотрите на результат:



Установка состояния Checkbutton

Здесь мы создаем переменную типа `BooleanVar`, которая не является стандартной переменной Python, это переменная Tkinter, затем передаем ее классу `Checkbutton`, чтобы установить состояние чекбокса как `True` в приведенном выше примере.

Вы можете установить для `BooleanVar` значение `false`, что бы чекбокс не был отмечен. Так же, используйте `IntVar` вместо `BooleanVar` и установите значения 0 и 1.

```
chk_state = IntVar()
chk_state.set(0) # False
chk_state.set(1) # True
```

Эти примеры дают тот же результат, что и `BooleanVar`.

Добавление виджетов Radio Button

Чтобы добавить radio кнопки, используйте класс `RadioButton`:

```
rad1 = Radiobutton(window, text='Первый', value=1)
```

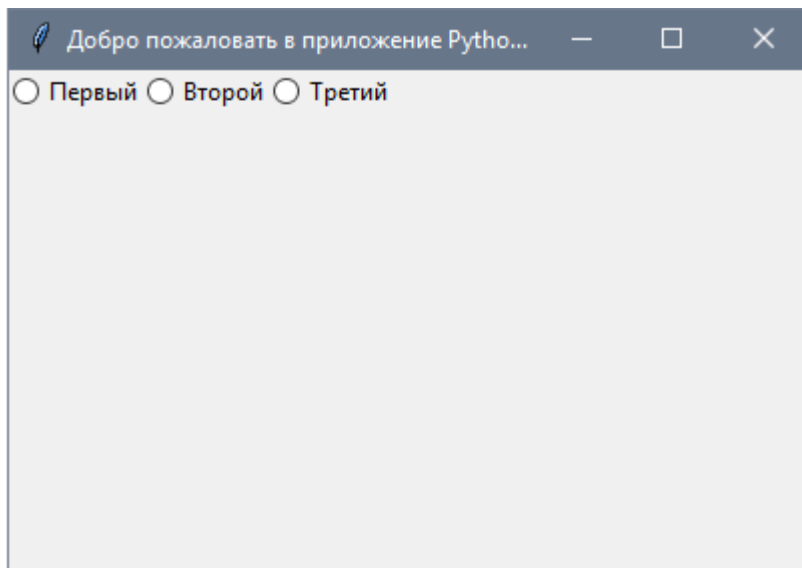
Обратите внимание, что вы должны установить `value` для каждой radio кнопки с уникальным значением, иначе они не будут работать.

```
from tkinter import *
from tkinter.ttk import Radiobutton
```

```
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
```

```
rad1 = Radiobutton(window, text='Первый', value=1)
rad2 = Radiobutton(window, text='Второй', value=2)
rad3 = Radiobutton(window, text='Третий', value=3)
rad1.grid(column=0, row=0)
rad2.grid(column=1, row=0)
rad3.grid(column=2, row=0)
window.mainloop()
```

Результатом вышеприведенного кода будет следующий:



Кроме того, вы можете задать `command` любой из этих кнопок для определенной функции. Если пользователь нажимает на такую кнопку, она запустит код функции.

Вот пример:

```
rad1 = Radiobutton(window, text='Первая', value=1, command=clicked)
```

```
def clicked():
```

```
    # Делайте, что нужно
```

Достаточно легко!

Получение значения Radio Button (Избранная Radio Button)

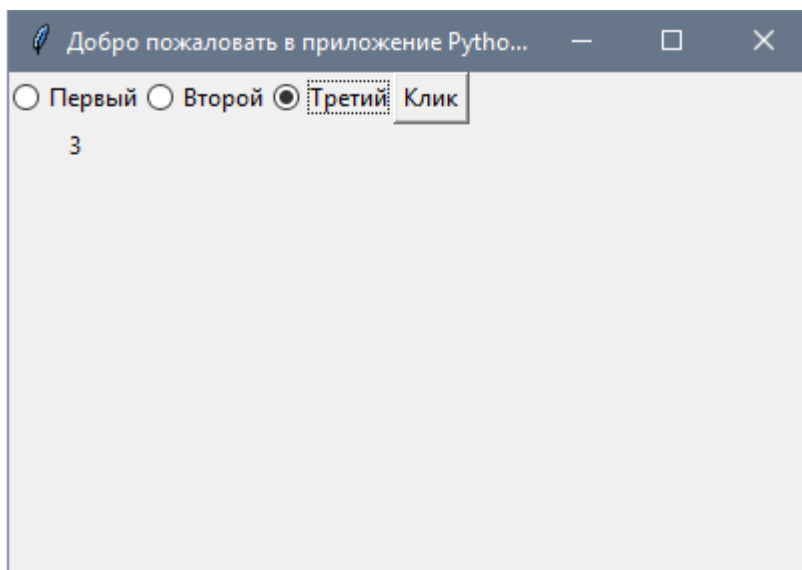
Чтобы получить текущую выбранную radio кнопку или ее значение, вы можете передать параметр переменной и получить его значение.

```
from tkinter import *
from tkinter.ttk import Radiobutton
def clicked():
    lbl.configure(text=selected.get())
```

```

window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
selected = IntVar()
rad1 = Radiobutton(window, text='Первый', value=1, variable=selected)
rad2 = Radiobutton(window, text='Второй', value=2, variable=selected)
rad3 = Radiobutton(window, text='Третий', value=3, variable=selected)
btn = Button(window, text="Клик", command=clicked)
lbl = Label(window)
rad1.grid(column=0, row=0)
rad2.grid(column=1, row=0)
rad3.grid(column=2, row=0)
btn.grid(column=3, row=0)
lbl.grid(column=0, row=1)
window.mainloop()

```



Каждый раз, когда вы выбираете radio button, значение переменной будет изменено на значение кнопки.

Добавление виджета ScrolledText (текстовая область Tkinter)

Чтобы добавить виджет `ScrolledText`, используйте класс `ScrolledText`:

```
from tkinter import scrolledtext
```

```
txt = scrolledtext.ScrolledText(window, width=40, height=10)
```

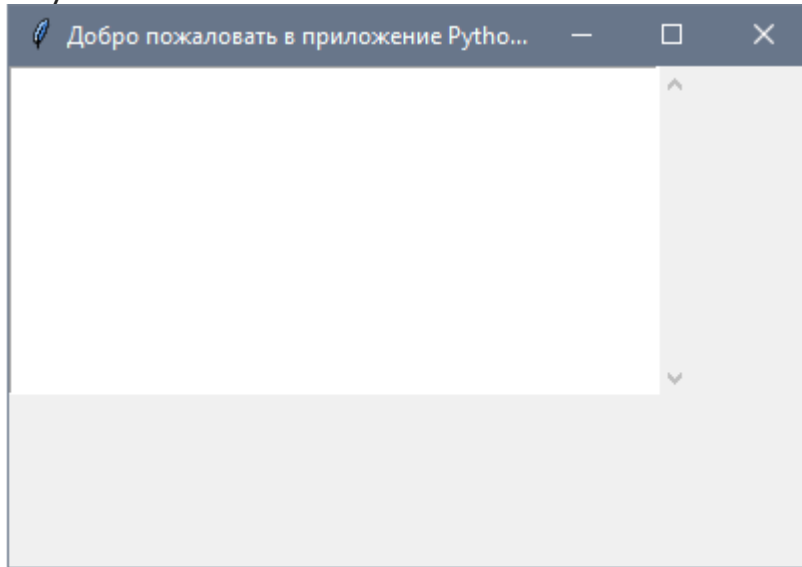
Здесь нужно указать ширину и высоту `ScrolledText`, иначе он заполнит все окно.

```

from tkinter import *
from tkinter import scrolledtext
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
txt = scrolledtext.ScrolledText(window, width=40, height=10)
txt.grid(column=0, row=0)
window.mainloop()

```

Результат:



Настройка содержимого Scrolledtext

Используйте метод `insert`, чтобы настроить содержимое `Scrolledtext`:

```
txt.insert(INSERT, 'Текстовое поле')
```

Удаление/Очистка содержимого Scrolledtext

Чтобы очистить содержимое данного виджета, используйте метод `delete`:

```
txt.delete(1.0, END) # мы передали координаты очистки
```

Создание всплывающего окна с сообщением

Чтобы показать всплывающее окно с помощью Tkinter, используйте `messagebox` следующим образом:

```
from tkinter import messagebox
```

```
messagebox.showinfo('Заголовок', 'Текст')
```

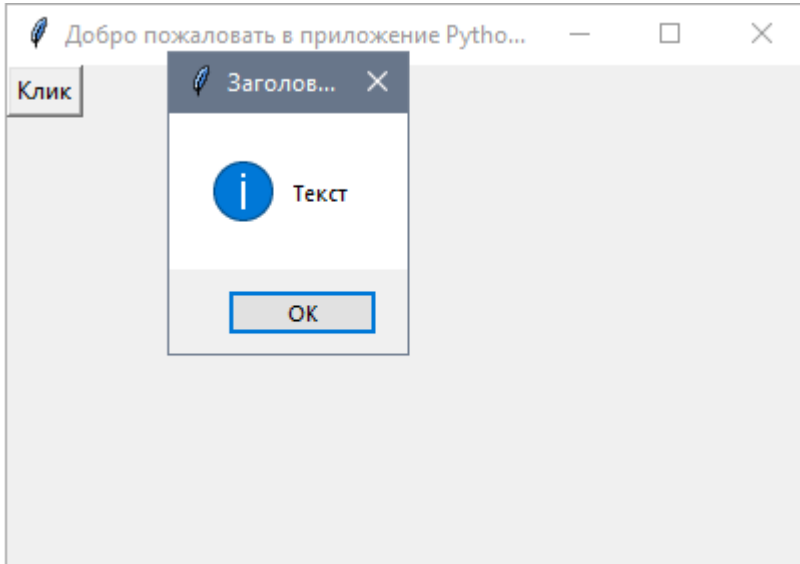
Довольно легко! Давайте покажем окно сообщений при нажатии на кнопку пользователем.

```
from tkinter import *
from tkinter import messagebox
def clicked():
```



```
messagebox.showinfo('Заголовок', 'Текст')
```

```
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
btn = Button(window, text='Клик', command=clicked)
btn.grid(column=0, row=0)
window.mainloop()
```



Когда вы нажмете на кнопку, появится информационное окно.

Показ сообщений о предупреждениях и ошибках

Вы можете показать предупреждающее сообщение или сообщение об ошибке таким же образом. Единственное, что нужно изменить—это функция сообщения.

```
messagebox.showwarning('Заголовок', 'Текст') # показывает предупреждающее сообщение
```

```
messagebox.showerror('Заголовок', 'Текст') # показывает сообщение об ошибке
```

Показ диалоговых окон с выбором варианта

Чтобы показать пользователю сообщение “да/нет”, вы можете использовать одну из следующих функций `messagebox`:

```
from tkinter import messagebox
res = messagebox.askquestion('Заголовок', 'Текст')
res = messagebox.asksyesno('Заголовок', 'Текст')
res = messagebox.asksyesnocancel('Заголовок', 'Текст')
res = messagebox.askokcancel('Заголовок', 'Текст')
res = messagebox.askretrycancel('Заголовок', 'Текст')
```

Вы можете выбрать соответствующий стиль сообщения согласно вашим потребностям. Просто замените строку функции `showinfo` на одну из предыдущих и запустите скрипт. Кроме того, можно проверить, какая кнопка нажата, используя переменную результата.

Если вы кликнете **OK**, **yes** или **retry**, значение станет **True**, а если выберете **no** или **cancel**, значение будет **False**.

Единственной функцией, которая возвращает одно из трех значений, является функция `askyesnocancel`; она возвращает **True/False/None**.

Добавление SpinBox (Виджет спинбокс)

Для создания виджета спинбокса, используйте класс `Spinbox`:

```
spin = Spinbox(window, from_=0, to=100)
```

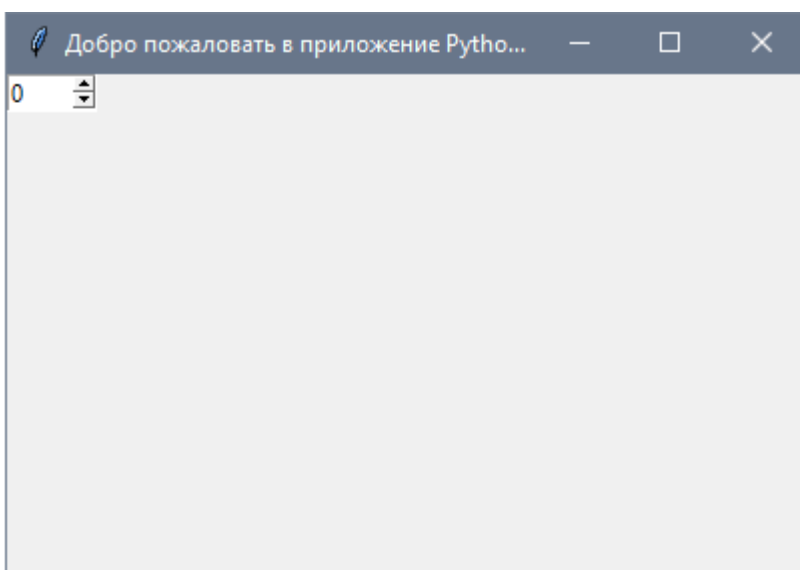
Таким образом, мы создаем виджет `Spinbox`, и передаем параметры `from` и `to`, чтобы указать диапазон номеров.

Кроме того, вы можете указать ширину виджета с помощью параметра `width`:

```
spin = Spinbox(window, from_=0, to=100, width=5)
```

Проверим пример полностью:

```
from tkinter import *
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
spin = Spinbox(window, from_=0, to=100, width=5)
spin.grid(column=0, row=0)
window.mainloop()
```



Вы можете указать числа для `Spinbox`, вместо использования всего диапазона следующим образом:

```
spin = Spinbox(window, values=(3, 8, 11), width=5)
```

Виджет покажет только эти 3 числа: 3, 8 и 11.

Задать значение по умолчанию для Spinbox

В случае, если вам нужно задать значение по умолчанию для Spinbox, вы можете передать значение параметру `textvariable` следующим образом:

```
var = IntVar()
```

```
var.set(36)
```

```
spin = Spinbox(window, from_=0, to=100, width=5, textvariable=var)
```

Теперь, если вы запустите программу, она покажет 36 как значение по умолчанию для Spinbox.

Добавление виджета Progressbar

Чтобы создать данный виджет, используйте класс `progressbar` :

```
from tkinter.ttk import Progressbar
```

```
bar = Progressbar(window, length=200)
```

Установите значение progressbar таким образом:

```
bar['value'] = 70
```

Вы можете установить это значение на основе любого процесса или при выполнении задачи.

Изменение цвета Progressbar

Изменение цвета Progressbar немного сложно. Сначала нужно создать стиль и задать цвет фона, а затем настроить созданный стиль на Progressbar. Посмотрите следующий пример:

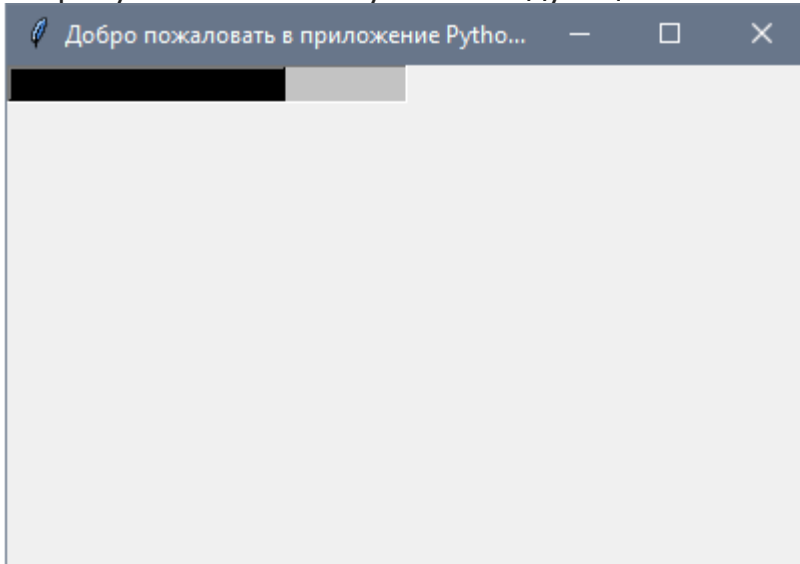
```
from tkinter import *
from tkinter.ttk import Progressbar
from tkinter import ttk
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
```

```

style = ttk.Style()
style.theme_use('default')
style.configure("black.Horizontal.TProgressbar", background='black')
bar = Progressbar(window, length=200, style='black.Horizontal.TProgressbar')
bar['value'] = 70
bar.grid(column=0, row=0)
window.mainloop()

```

И в результате вы получите следующее:



Добавление поля загрузки файла

Для добавления поля с файлом, используйте класс `filedialog`:

```
from tkinter import filedialog
```

```
file = filedialog.askopenfilename()
```

После того, как вы выберете файл, нажмите “Открыть”; переменная файла будет содержать этот путь к файлу. Кроме того, вы можете запросить несколько файлов:

```
files = filedialog.askopenfilenames()
```

Указание типа файлов (расширение фильтра файлов)

Возможность указания типа файлов доступна при использовании параметра `filetypes`, однако при этом важно указать расширение в tuples.

```
file = filedialog.askopenfilename(filetypes = (("Text files", "*.txt"), ("all files", "*.*")))
```

Вы можете запросить каталог, используя метод `askdirectory` :

```
dir = filedialog.askdirectory()
```

Вы можете указать начальную директорию для диалогового окна файла, указав `initialdir` следующим образом:

```
from os import path
```

```
file = filedialog.askopenfilename(initialdir= path.dirname(__file__))
```

Добавление панели меню

Для добавления панели меню, используйте класс `menu`:

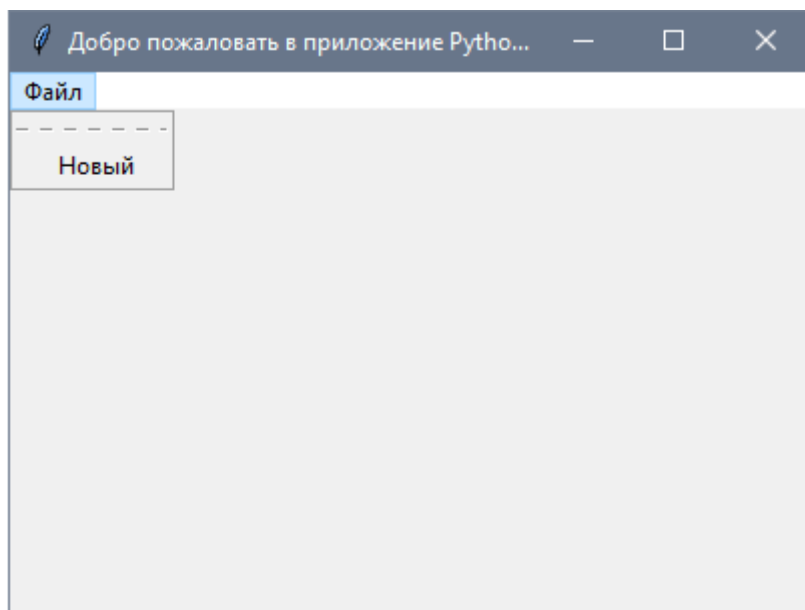
```
from tkinter import Menu
menu = Menu(window)
menu.add_command(label='Файл')
window.config(menu=menu)
```

Сначала мы создаем меню, затем добавляем наш первый пункт подменю. Вы можете добавлять пункты меню в любое меню с помощью функции `add_cascade()` таким образом:

```
menu.add_cascade(label='Автор', menu=new_item)
```

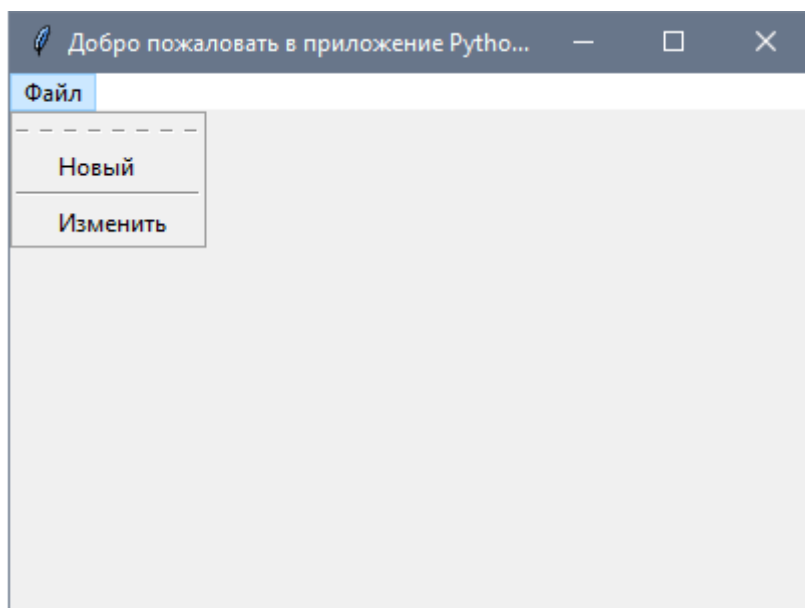
Наш код будет выглядеть так:

```
from tkinter import *
from tkinter import Menu
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
menu = Menu(window)
new_item = Menu(menu)
new_item.add_command(label='Новый')
menu.add_cascade(label='Файл', menu=new_item)
window.config(menu=menu)
window.mainloop()
```



Таким образом, вы можете добавить столько пунктов меню, сколько захотите.

```
from tkinter import *
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
menu = Menu(window)
new_item = Menu(menu)
new_item.add_command(label='Новый')
new_item.add_separator()
new_item.add_command(label='Изменить')
menu.add_cascade(label='Файл', menu=new_item)
window.config(menu=menu)
window.mainloop()
```



Теперь мы добавляем еще один пункт меню “Изменить” с разделителем меню. Вы можете заметить пунктирную линию в начале, если вы нажмете на эту строку, она отобразит пункты меню в небольшом отдельном окне.

Можно отключить эту функцию, с помощью `tearoff` подобным образом:

```
new_item = Menu(menu, tearoff=0)
```

Просто отредактируйте `new_item`, как в приведенном выше примере и он больше не будет отображать пунктирную линию.

Вы так же можете ввести любой код, который работает, при нажатии пользователем на любой элемент меню, задавая свойство команды.

```
new_item.add_command(label='Новый', command=clicked)
```

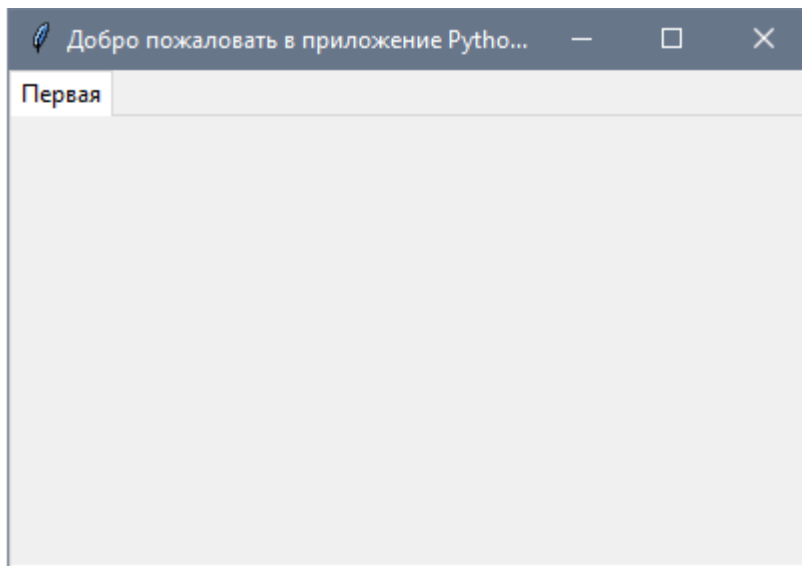
Добавление виджета Notebook (Управление вкладкой)

Для удобного управления вкладками реализуйте следующее:

- Для начала, создается элемент управления вкладкой, с помощью класса `Notebook`.
- Создайте вкладку, используя класс `Frame`.
- Добавьте эту вкладку в элемент управления вкладками.
- Запакуйте элемент управления вкладкой, чтобы он стал видимым в окне.

```
from tkinter import *  
from tkinter import ttk
```

```
window = Tk()  
window.title("Добро пожаловать в приложение PythonRu")  
window.geometry('400x250')  
tab_control = ttk.Notebook(window)  
tab1 = ttk.Frame(tab_control)  
tab_control.add(tab1, text='Первая')  
tab_control.pack(expand=1, fill='both')  
window.mainloop()
```

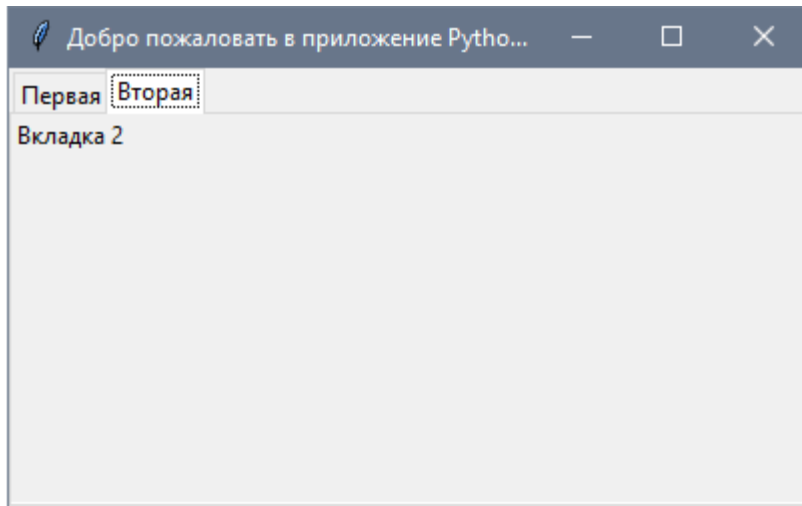


Таким образом, вы можете добавлять столько вкладок, сколько нужно.

Добавление виджетов на вкладку

После создания вкладок вы можете поместить виджеты внутри этих вкладок, назначив родительское свойство нужной вкладке.

```
from tkinter import *
from tkinter import ttk
window = Tk()
window.title("Добро пожаловать в приложение PythonRu")
window.geometry('400x250')
tab_control = ttk.Notebook(window)
tab1 = ttk.Frame(tab_control)
tab2 = ttk.Frame(tab_control)
tab_control.add(tab1, text='Первая')
tab_control.add(tab2, text='Вторая')
lbl1 = Label(tab1, text='Вкладка 1')
lbl1.grid(column=0, row=0)
lbl2 = Label(tab2, text='Вкладка 2')
lbl2.grid(column=0, row=0)
tab_control.pack(expand=1, fill='both')
window.mainloop()
```

Добавление интервала для виджетов (Заполнение)

Вы можете добавить отступы для элементов управления, чтобы они выглядели хорошо организованными с использованием свойств `padx` и `pady`.

Передайте `padx` и `pady` любому виджету и задайте значение.

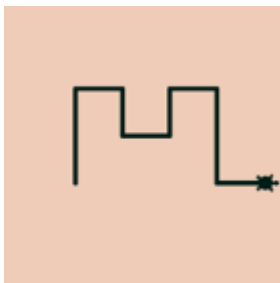
```
lbl1 = Label(tab1, text='label1', padx=5, pady=5)
```

Задания для выполнения:

Все решенные задачи должны быть доступны в графическом окне как вкладки: «Задача 1», «Задача 2» и т.д. Все виджеты на вкладке должны быть подписаны. Например «Выберите цвет линии» и далее текстовое поле или всплывающее окно и т.д.

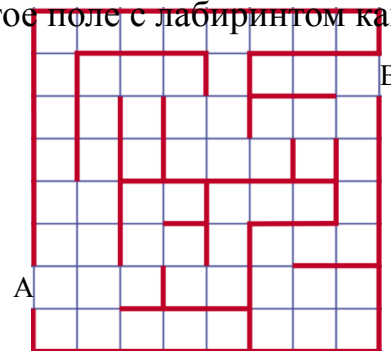
Вариант 1.

1. Вкладка задача 1 должна содержать выпадающий список, с выбором цвета линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок



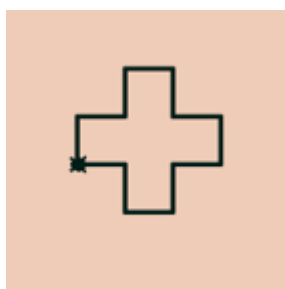
2. Вкладка должна содержать два текстовых поля, для выбора высоты и ширины прямоугольника и кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется прямоугольник с желтыми контурами и оранжевой заливкой на зеленом фоне.

3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.
Необходимо нарисовать три графика функций. Выражение для построения графика вводится пользователем в текстовом поле. Если одно из полей не заполнено – выводится предупреждение. Организовать выбор цвета линий, график должен содержать легенду.
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».
5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.



Вариант 2.

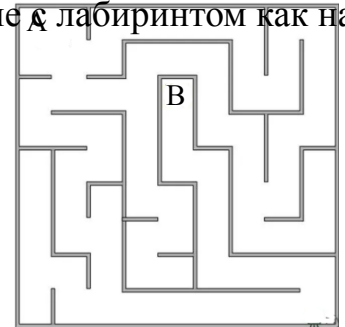
1. Вкладка задача 1 должна содержать выпадающий список, с выбором цвета линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок



2. Вкладка должна содержать текстовое поле для выбора длины стороны квадрата и выпадающий список для выбора цвета контура, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется квадрат с заданными параметрами контура и зеленой заливкой на синем фоне.
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.
Организовать генерацию массива случайных чисел. Размер – выбирается пользователем. Организовать выбор распределения случайных величин в списке для генерации. Построить гистограмму распределения полученных случайных чисел.

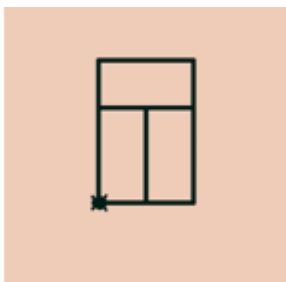
Пользователь должен иметь возможность выбора ColorMap при отображении гистограммы (не менее 5 разных вариантов)

4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».
5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.



Вариант 3.

1. Вкладка задача 1 должна содержать выпадающий список, с выбором цвета линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок

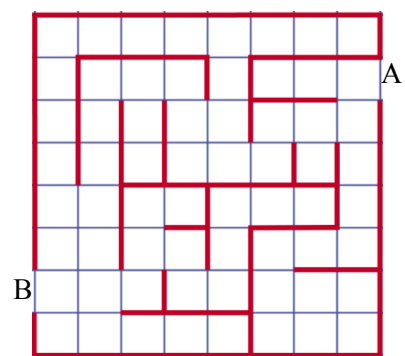


2. Вкладка должна содержать текстовое поле для выбора радиуса круга и выпадающий список для выбора цвета заливки, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется окружность с красным контуром на черном фоне и заливкой заданного цвета.

3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.

Реализовать программу для генерации пароля. С помощью различных виджетов пользователь выбирает требования к паролю – минимальная/максимальная длина, необходимость и количество цифр, возможность расположения их не подряд, букв с разным регистром, специальных символов.

4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».

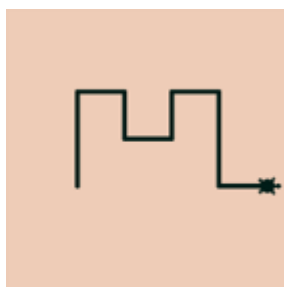


5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на

рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.

Вариант 4.

1. Вкладка задача 1 должна содержать выпадающий список, с выбором цвета фона и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок



2. Вкладка должна содержать четыре Radio Button для выбора цвета контура прямоугольника и два текстовых поля для определения его сторон, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется прямоугольник с заданными параметрами на желтом поле и желтой заливкой.
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.

Программа должна решать систему из 3 линейных уравнений методом Гаусса. Ввод уравнений организуется с использованием текстовых полей и должен иметь примерно следующий вид:

$$\square x + \square y + \square z = \square$$

$$\square x + \square y + \square z = \square$$

$$\square x + \square y + \square z = \square$$

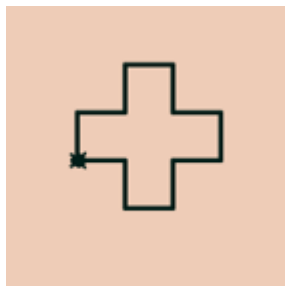
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».
5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По



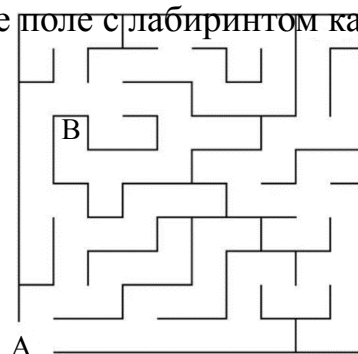
достижении точки В – вывести количество шагов.

Вариант 5.

1. Вкладка задача 1 должна содержать всплывающее окно, с выбором цвета фона и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок

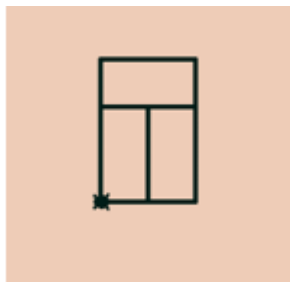


2. Вкладка должна содержать два текстовых поля, для выбора высоты и ширины прямоугольника и кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется прямоугольник с желтыми контурами и оранжевой заливкой на зеленом фоне
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне
Напишите программу для построения круговой диаграммы по заданным параметрам. Пользователь должен иметь возможность ввести название диаграммы, размер массива с данными, после чего должно появиться окно с массивом который заполняет пользователь. После ввода всех значений – строится диаграмма.
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».
5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.



Вариант 6.

1. Вкладка задача 1 должна содержать выпадающий список, с выбором цвета фона и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок



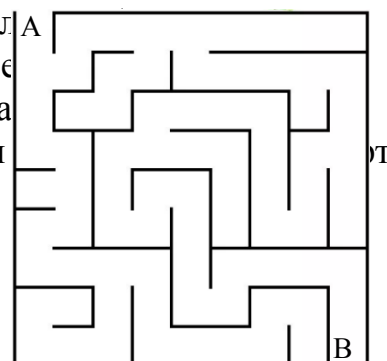
2. Вкладка должна содержать текстовое поле для выбора радиуса круга и выпадающий список для выбора цвета заливки, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется окружность с красным контуром на черном фоне и заливкой заданного цвета

3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.

Написать программу для работы с текстовым файлом. Пользователь вводит путь к файлу и выбирает одно из действий: подсчитать количество вхождений указанной буквы, подсчитать количество символов, найти самую часто встречаемую букву, найти самую редко встречаемую букву, подсчитать количество слов, количество строк, количество знаков препинания, количество гласных и согласных букв.

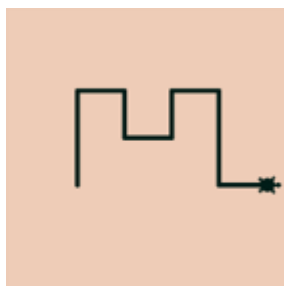
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».

5. Написать игру «Лабиринт». На экран выводится клетчатое поле рисунке. У пользователя имеется 4 кнопки обозначающие стрелки движения. При нажатии стрелки соответствующая клеточка за движение в выбранном направлении невозможно – выводится – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.

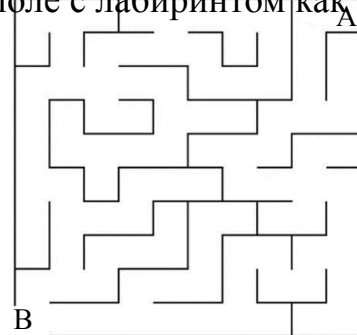


Вариант 7.

1. Вкладка задача 1 должна содержать текстовое поле для ввода значения толщины линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок

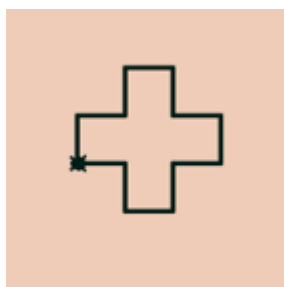


2. Вкладка должна содержать текстовое поле для выбора длины стороны квадрата и выпадающий список для выбора цвета контура, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется квадрат с заданными параметрами контура и зеленой заливкой на синем фоне
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.
Необходимо нарисовать три графика функций. Выражение для построения графика вводится пользователем в текстовом поле. Если одно из полей не заполнено – выводится предупреждение. Организовать выбор цвета линий, график должен содержать легенду
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».
5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.

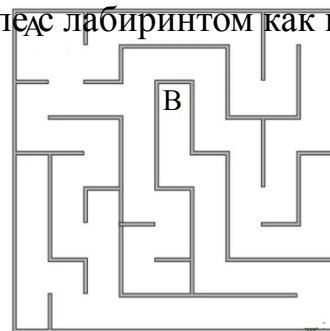


Вариант 8.

1. Вкладка задача 1 должна содержать текстовое поля для ввода значения толщины линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок

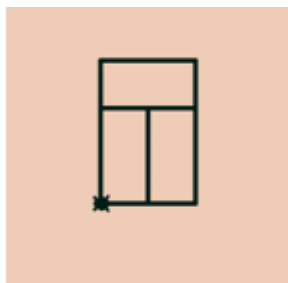


2. Вкладка должна содержать четыре Radio Button для выбора цвета контура прямоугольника и два текстовых поля для определения его сторон, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется прямоугольник с заданными параметрами на желтом поле и желтой заливкой
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.
Организовать генерацию массива случайных чисел. Размер – выбирается пользователем. Организовать выбор распределения случайных величин в списке для генерации. Построить гистограмму распределения полученных случайных чисел. Пользователь должен иметь возможность выбора ColorMap при отображении гистограммы (не менее 5 разных вариантов)
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».
5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.



Вариант 9.

1. Вкладка задача 1 должна содержать текстовое поля для ввода значения толщины линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок

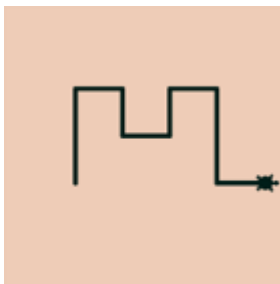


2. Вкладка должна содержать два текстовых поля, для выбора высоты и ширины прямоугольника и кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется прямоугольник с желтыми контурами и оранжевой заливкой на зеленом фоне
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных

Реализовать программу для генерации пароля. С помощью различных виджетов пользователь выбирает требования к паролю – минимальная/максимальная длина, необходимость и количество цифр, возможность расположения их не подряд, букв с разным регистром, специальных символов.

- поле с лабиринтом как
стрелки – направления
а закрашивается. Если
тся предупреждение. Ст
ести количество шагов.
-
- A

1. Вкладка задача 1 должна содержать выпадающий список, с выбором цвета линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок



- Программа должна решать систему из 3 линейных уравнений методом Гаусса. Ввод уравнений организуется с использованием текстовых полей и должен иметь примерно

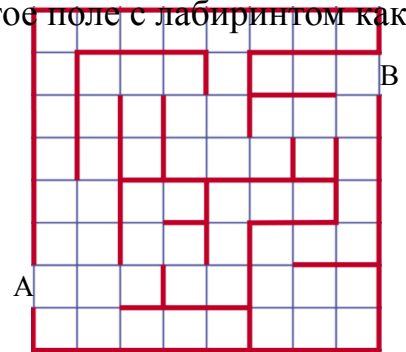
следующий вид:

$$\square x + \square y + \square z = \square$$

$$\square x + \square y + \square z = \square$$

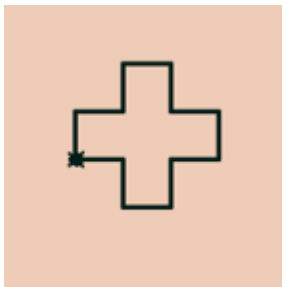
$$\square x + \square y + \square z = \square$$

4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».
5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.



Вариант 11.

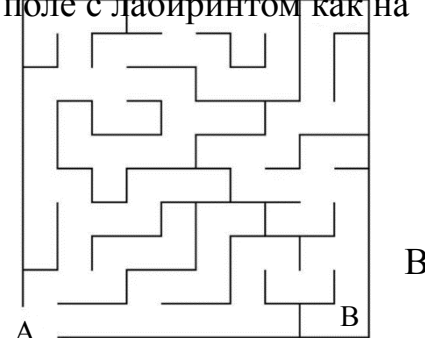
1. Вкладка задача 1 должна содержать выпадающий список, с выбором цвета линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок



2. Вкладка должна содержать текстовое поле для выбора радиуса круга и выпадающий список для выбора цвета заливки, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется окружность с красным контуром на черном фоне и заливкой заданного цвета
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне
- Напишите программу для построения круговой диаграммы по заданным параметрам. Пользователь должен иметь возможность ввести название диаграммы, размер массива с данными, после чего должно появиться окно с массивом который заполняет пользователь. После ввода всех значений – строится диаграмма.
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и

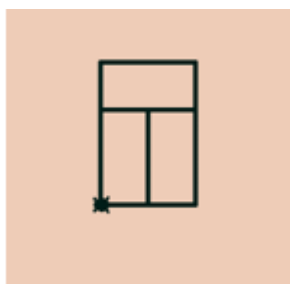
точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».

- Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки – вывести количество шагов.



Вариант 12.

- Вкладка задача 1 должна содержать всплывающее окно, с выбором цвета линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок

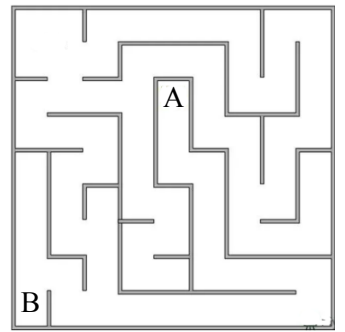


- Вкладка должна содержать два текстовых поля, для выбора высоты и ширины прямоугольника и кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется прямоугольник с желтыми контурами и оранжевой заливкой на зеленом фоне

- Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.

Написать программу для работы с текстовым файлом. Пользователь вводит путь к файлу и выбирает одно из действий: подсчитать количество вхождений указанной буквы, подсчитать количество символов, найти самую часто встречаемую букву, найти самую редко встречаемую букву, подсчитать количество слов, количество строк, количество знаков препинания, количество гласных и согласных букв

- Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».

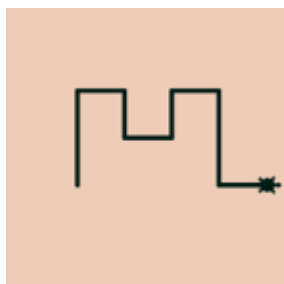


5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления

движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.

Вариант 13.

1. Вкладка задача 1 должна содержать выпадающий список, с выбором цвета фона и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок



2. Вкладка должна содержать текстовое поле для выбора радиуса круга и выпадающий список для выбора цвета заливки, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется окружность с красным контуром на черном фоне и заливкой заданного цвета

3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.

Необходимо нарисовать три графика функций. Выражение для построения графика вводится пользователем в текстовом поле. Если одно из полей не заполнено – выводится предупреждение. Организовать выбор цвета линий, график должен содержать легенду

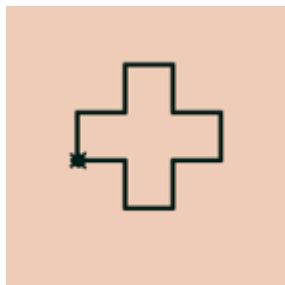
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».

5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.



Вариант 14.

1. Вкладка задача 1 должна содержать выпадающий список, с выбором цвета фона и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок

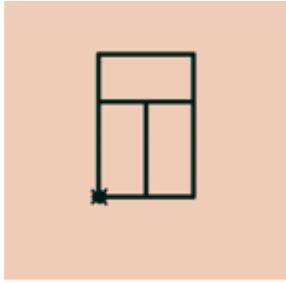


2. Вкладка должна содержать четыре Radio Button для выбора цвета контура прямоугольника и два текстовых поля для определения его сторон, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется прямоугольник с заданными параметрами на желтом поле и желтой заливкой
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.
Организовать генерацию массива случайных чисел. Размер – выбирается пользователем. Организовать выбор распределения случайных величин в списке для генерации. Построить гистограмму распределения полученных случайных чисел. Пользователь должен иметь возможность выбора ColorMap при отображении гистограммы (не менее 5 разных вариантов)
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».
5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки – вывести количество шагов.

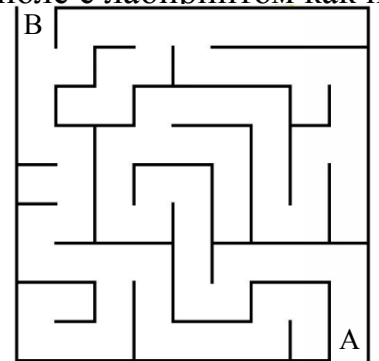


Вариант 15.

1. Вкладка задача 1 должна содержать выпадающий список, с выбором цвета фона и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок

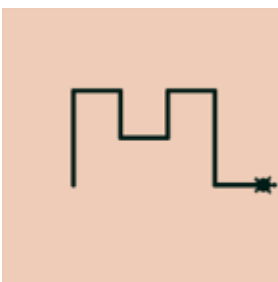


2. Вкладка должна содержать два текстовых поля, для выбора высоты и ширины прямоугольника и кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется прямоугольник с желтыми контурами и оранжевой заливкой на зеленом фоне
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.
Реализовать программу для генерации пароля. С помощью различных виджетов пользователь выбирает требования к паролю – минимальная/максимальная длина, необходимость и количество цифр, возможность расположения их не подряд, букв с разным регистром, специальных символов
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».
5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.

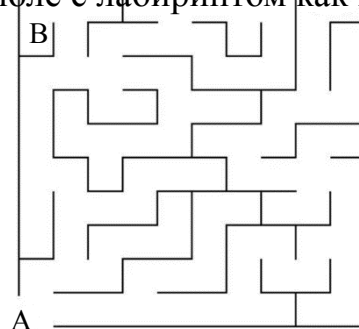


Вариант 16.

1. Вкладка задача 1 должна содержать выпадающий список, с выбором цвета линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок

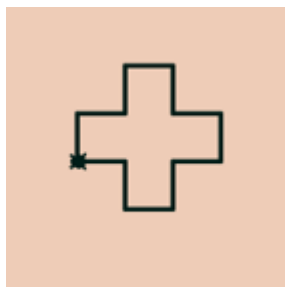


2. Вкладка должна содержать текстовое поле для выбора длины стороны квадрата и выпадающий список для выбора цвета контура, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется квадрат с заданными параметрами контура и зеленой заливкой на синем фоне
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне
 Напишите программу для построения круговой диаграммы по заданным параметрам. Пользователь должен иметь возможность ввести название диаграммы, размер массива с данными, после чего должно появиться окно с массивом который заполняет пользователь. После ввода всех значений – строится диаграмма\
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».
5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.



Вариант 17.

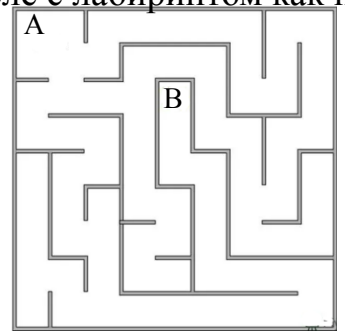
1. Вкладка задача 1 должна содержать выпадающий список, с выбором цвета линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок



2. Вкладка должна содержать текстовое поле для выбора радиуса круга и выпадающий список для выбора цвета заливки, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется окружность с красным контуром на черном фоне и заливкой заданного цвета
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.

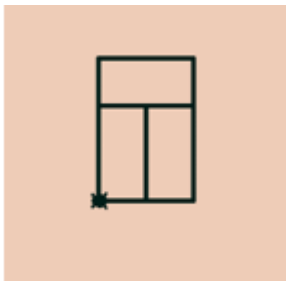
Написать программу для работы с текстовым файлом. Пользователь вводит путь к файлу и выбирает одно из действий: подсчитать количество вхождений указанной буквы, подсчитать количество символов, найти самую часто встречаемую букву, найти самую редко встречаемую букву, подсчитать количество слов, количество строк, количество знаков препинания, количество гласных и согласных букв

4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».
5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.



Вариант 18.

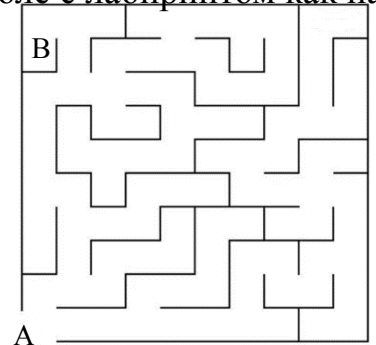
1. Вкладка задача 1 должна содержать текстовое поле для ввода значения толщины линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок



2. Вкладка должна содержать четыре Radio Button для выбора цвета заливки прямоугольника и два текстовых поля для определения его сторон, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется прямоугольник с заданными параметрами на желтом поле с зеленым контуром
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.
Необходимо нарисовать три графика функций. Выражение для построения графика вводится пользователем в текстовом поле. Если одно из полей не заполнено – выводится предупреждение. Организовать выбор цвета линий, график должен содержать легенду
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и

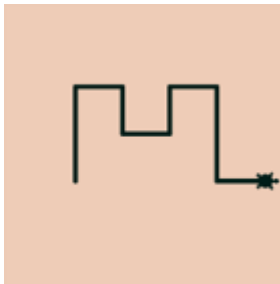
точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».

5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.

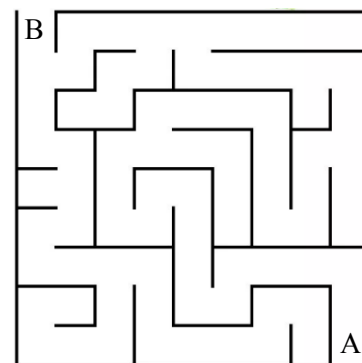


Вариант 19.

1. Вкладка задача 1 должна содержать текстовое поля для ввода значения толщины линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок



2. Вкладка должна содержать два текстовых поля, для выбора высоты и ширины прямоугольника и кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется прямоугольник с красными контурами и зеленой заливкой на оранжевом фоне
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.
Организовать генерацию массива случайных чисел. Размер – выбирается пользователем. Организовать выбор распределения случайных величин в списке для генерации. Построить гистограмму распределения полученных случайных чисел. Пользователь должен иметь возможность выбора ColorMap при отображении гистограммы (не менее 5 разных вариантов)
4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».

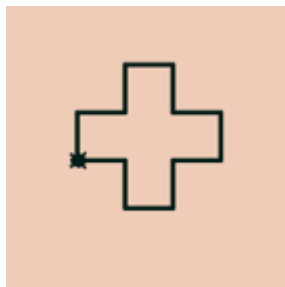


5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если

движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.

Вариант 20.

1. Вкладка задача 1 должна содержать текстовое поле для ввода значения толщины линии и одну кнопку «Выполнить задание 1». При нажатии на кнопку с использованием команд черепашки нарисовать рисунок



2. Вкладка должна содержать текстовое поле для выбора длины стороны квадрата и выпадающий список для выбора цвета заливки, кнопку выполнить задание. При нажатии на кнопку с использованием команд черепашки рисуется квадрат с заданными параметрами заливки и синим контуром на черном фоне
3. Вкладка должна содержать виджеты для ввода или выбора всех требуемых входных данных для решения задачи. Решение должно отображаться в новом окне.

Реализовать программу для генерации пароля. С помощью различных виджетов пользователь выбирает требования к паролю – минимальная/максимальная длина, необходимость и количество цифр, возможность расположения их не подряд, букв с разным регистром, специальных символов

4. Написать программу «Калькулятор», которая будет содержать кнопки с цифрами (и точку), и основными математическими операциями (сложение, вычитание, умножение, деление, возведение в квадрат, скобки). Отображать результат ввода при каждом нажатии кнопки. Вывод результата вычисления – после нажатия «=».
5. Написать игру «Лабиринт». На экран выводится клетчатое поле с лабиринтом как на рисунке. У пользователя имеется 4 кнопки обозначающие стрелки – направления движения. При нажатии стрелки соответствующая клеточка закрашивается. Если движение в выбранном направлении невозможно – выводится предупреждение. Старт – точка А, финиш – точка В. По достижении точки В – вывести количество шагов.

