федеральное государственное бюджетное образовательное учреждение высшего образования

**«Вологодский государственный университет»**

Институт математики, естественных и компьютерных наук

Кафедра автоматики и вычислительной техники

Отчет по лабораторной работе №**7**

Дисциплина: «Кроссплатформенное программирование»

Наименование темы: «Работа с графикой. Модуль tkinter»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 09.03.03  код направления  подготовки/  специальности | 43.10  код выпускающей  кафедры | 6  регистрационный номер по журналу | 01  код формы  обучения | 2025  год |

|  |  |
| --- | --- |
| Руководитель | ст. преподаватель Ковырзина Т.Ф. |
| Выполнил (а) студент | Леньков Р. А. |
| Группа, курс | 4Б09 ПИБ - 21 |
| Дата сдачи |  |
| Дата защиты |  |
| Оценка по защите |  |

Вологда

2025 г.

**Лабораторная работа №7.**

**Работа с графикой. Модуль tkinter**

**Цель работы:** Рассмотрение способов работы с графикой с помощью модуля tkinter

**Краткий теоретический материал**

Графический пользовательский интерфейс GUI (Graphical User Interface) – это разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

Библиотека tkinter (от англ. Tk interface) – это встроенная графическая библиотека на основе средств Tk, широко распространенная в мире GNU/Linux и других UNIX-подобных систем, которая портирована (переведена), в частности, на Microsoft Windows и Apple Mac OS).

**Общие свойства и методы виджетов**

Все виджеты в Tkinter обладают некоторыми общими свойствами. Виджеты создаются вызовом конструктора соответствующего класса. Первый аргумент (как правило неименованный, но можно использовать имя master) – это родительский виджет, в который будет помещен (упакован) текущий виджет. Родительский виджет можно не указывать, в таком случае будет использовано главное окно приложения.

**Упаковщики**

Упаковщик (менеджер геометрии, менеджер расположения) – это специальный механизм, который размещает (упаковывает) виджеты на окне. В tkinter есть три упаковщика: pack(), place() и grid(). Обратите внимание, что в одном виджете можно использовать только один тип упаковки, при смешивании разных типов упаковки программа, скорее всего, не будет работать.

Для каждого виджета указываем, в какой он находится строке, и в каком столбце. Если нужно, указываем, сколько ячеек он занимает (если, например, нам нужно разместить три виджета под одним, необходимо "растянуть" верхний на три ячейки).

**Привязка событий**

Для большинства виджетов, реагирующих на действие пользователя, активацию виджета (например, нажатие кнопки) можно привязать к функции обработчику этого события, используя опцию command. К таким виджетам относятся: Button, Checkbutton, Radiobutton, Spinbox, Scrollbar, Scale. Например: button = Button(command=callback). Такой способ является предпочтительным и наиболее удобным способом привязки. Другим способом привязки является использование метода bind().

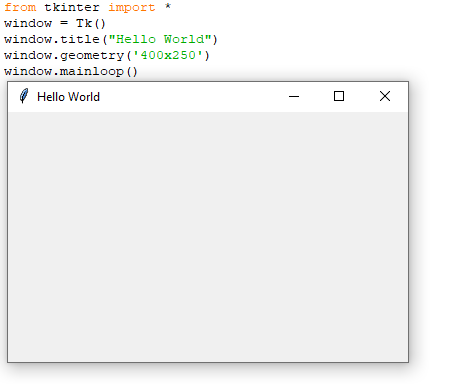
**Задание 1.**

1. Для начала, следует импортировать Tkinter и создать окно, в котором мы зададим его название. функция mainloop вызывает бесконечный цикл окна, поэтому окно будет ждать любого взаимодействия с пользователем, пока не будет закрыто.

В отсутствии функции mainloop , для пользователя ничего не отобразится.

2. Установите размер окна по умолчанию, используя функцию geometry следующим образом:

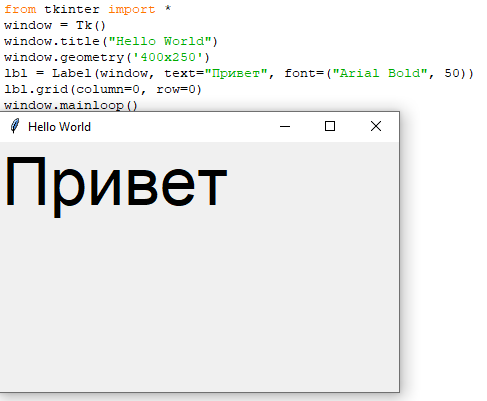
window.geometry('400x250')



**Задание 2. Создание Виджета Label**

Также можно изменить стиль шрифта. Для этого передайте параметр font таким образом:

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



**Задание 3. Создание кнопки**

1. Кнопка создается и добавляется в окно так же, как и метка:

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

btn.grid(column=1, row=0)

1. Поменяйте 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")

1. запишем функцию, которую нужно выполнить при нажатии кнопки:

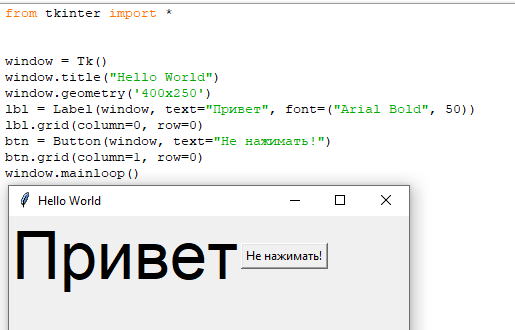
def clicked():

lbl.configure(text="кнопку нажали")

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.



**Задание4**

1. создайте текстовое поле с помощью класса Tkinter Entry следующим образом:

txt = Entry(window, width=10)

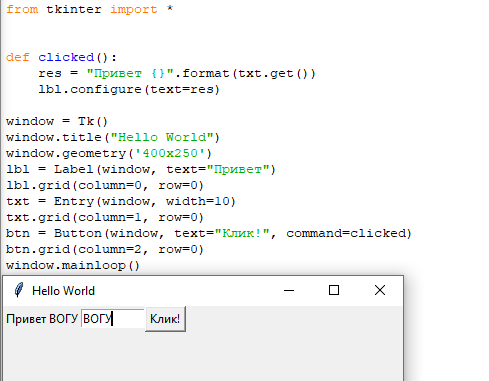
Затем вы можете добавить его в окно, используя функцию grid.

получить текст ввода, используя функцию get.

def clicked():

res = "Привет {}".format(txt.get())

lbl.configure(text=res)



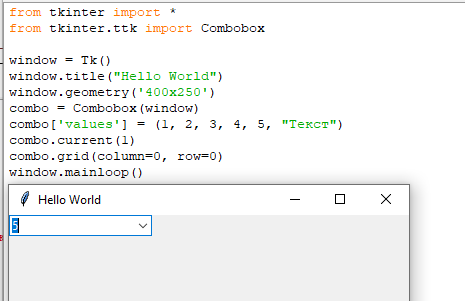
**Задание 5**

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

from tkinter.ttk import Combobox

combo = Combobox(window)

Затем добавьте свои значения в поле со списком.



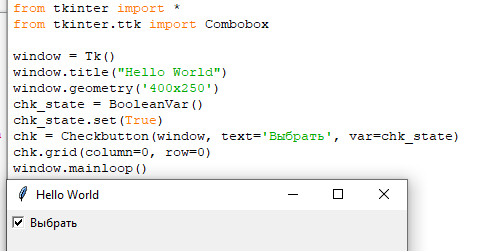
**Задание 6**

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:



**Задание 7**

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.

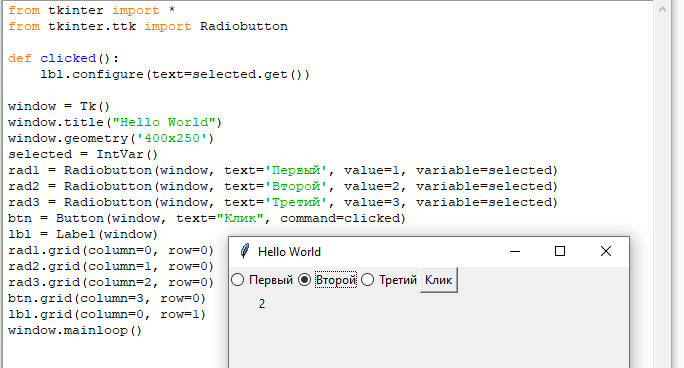
Кроме того, вы можете задать command любой из этих кнопок для определенной функции. Если пользователь нажимает на такую кнопку, она запустит код функции.  
Вот пример:

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

def clicked():

# Делайте, что нужно

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



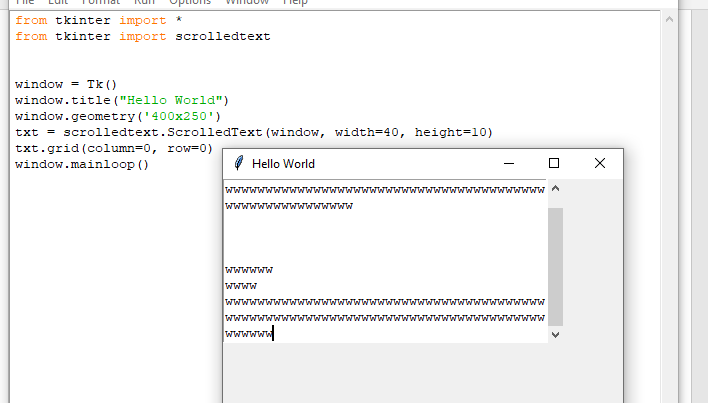
**Задание 8**

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.

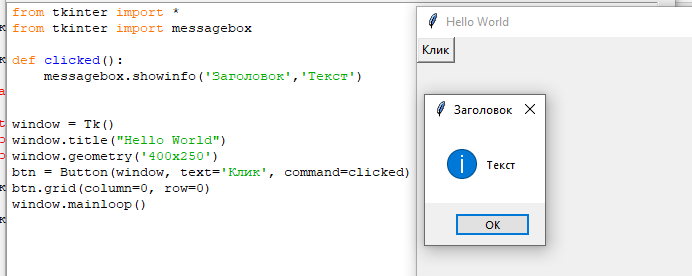


**Задание9**

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

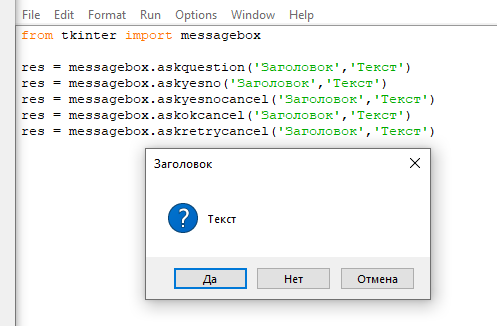
from tkinter import messagebox

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



**Задание 10**

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



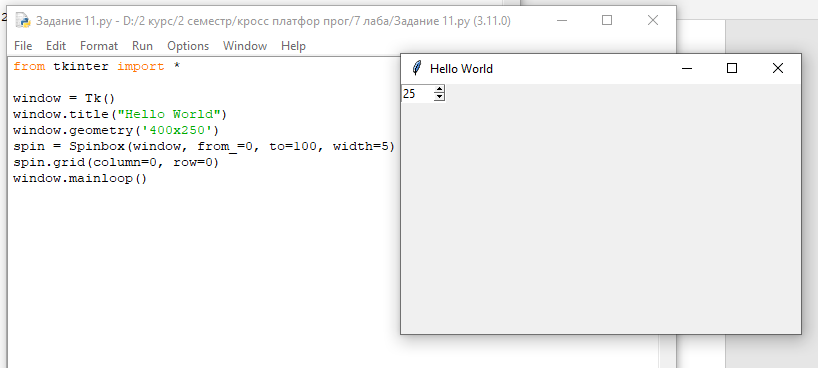
**Задание 11**

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)



**Задание 12**

Чтобы создать данный виджет, используйте класс progressbar :

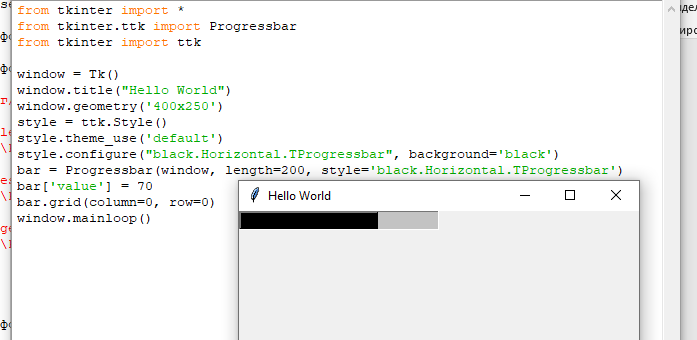
from tkinter.ttk import Progressbar

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.



**Задание 13**

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\_\_))

Задание 13

Для добавления панели меню, используйте класс menu:

from tkinter import Menu

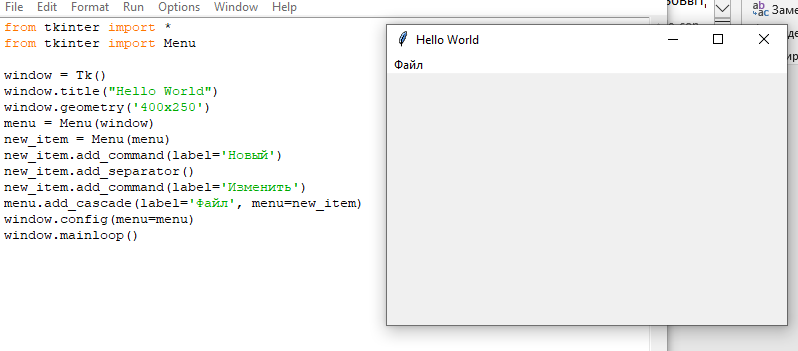
menu = Menu(window)

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:

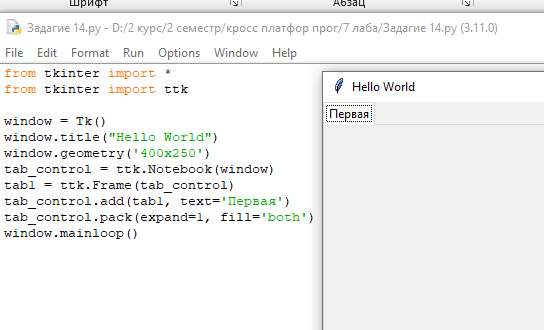
menu.add\_cascade(label='Автор', menu=new\_item)



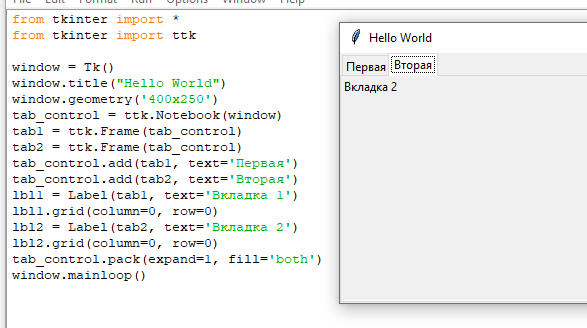
**Задание 14**

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.



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



**Индивидуальные задания**

Разработать программу на языке Python, в которой на базе модуля *tkinter* разработан графический интерфейс, имеющий несколько окон или вкладок:

* окно с кнопкой (объект Button), надпись на которой меняется по каждому щелчку и содержит следующую информацию – номер щелчка и число, которое определяется согласно колонке "Число" табл. №1:
  + 1 – очередное простое число (начиная с 1);
  + 2 – очередное число Фибоначчи (начиная с 1);
  + 3 – очередное простое число вида 2 n - 1 (начиная с n = 1);
* окно, содержащее:
  + текстовое поле (объект Text), куда выводится начало некоторого небольшого сообщения (рассказа и т.п.);
  + поле ввода пароля (объект Entry);
  + кнопку для продолжения вывода сообщения (объект Button);
  + второе текстовое поле (объект Text), куда выводится неожиданное (смешное) завершение сообщения (при правильном вводе пароля) или сообщение о неправильном вводе пароле;
  + метки (объект Label) – при необходимости;
* окно, содержащее:
  + пять флажков (объекты Checkbutton), задающие согласно колонке "Флажки" табл. №1:  
    1 – числа 1, 2, 4, 8 и 16;  
    2 – 5 букв кириллицы (на усмотрение студента);  
    2 – 5 английских букв (на усмотрение студента);
  + текстовое поле, куда помещается результат:  
    для чисел – сумма заданных чисел;  
    для букв – слово, состоящее из выбранных букв;
* окно, содержащее:
  + переключатели (объекты Radiobutton), задающие согласно колонке "Переключатели" табл. №1:  
    1 – изменяемые типы;  
    2 – неизменяемые типы;  
    3 – контейнеры;  
    4 – последовательности;
  + текстовое поле, куда помещается результат – пример литерала для выбранного класса.

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1 – Перечень индивидуальных заданий | | | |
| **Номер п/п** | **Число** | **Флажкий** | **Переключатели** |
| 6 | 3 | 2 | 2 |

import **tkinter** as **tk**

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

class **PrimeButtonTab**(**ttk**.**Frame**):

    def **\_\_init\_\_**(self, master):

**super**().**\_\_init\_\_**(master)

        self.click\_count = 0

        self.n = 1

        self.primes = []

        self.button = **tk**.**Button**(

            self,

            text="Click me (0)",

            command=self.**update\_button\_text**

        )

        self.button.pack(pady=20)

    def **is\_prime**(self, num):

        if num < 2:

            return False

        for i in **range**(2, **int**(num\*\*0.5) + 1):

            if num % i == 0:

                return False

        return True

    def **update\_button\_text**(self):

        self.click\_count += 1

        while True:

            candidate = 2\*\*self.n - 1

            if self.**is\_prime**(candidate):

                self.primes.**append**(candidate)

                self.n += 1

                break

            self.n += 1

        self.button.config(

            text=f"Щелчок: {self.click\_count}, Число: {self.primes[-1]}"

        )

class **StoryTab**(**ttk**.**Frame**):

    def **\_\_init\_\_**(self, master):

**super**().**\_\_init\_\_**(master)

        self.story\_part1 = **tk**.**Text**(self, height=5, width=50)

        self.story\_part1.**insert**(

**tk**.END,

            "Я ничего умного не придумал\n"

            "На этом все:\n"

            "Введи пароль.'\n"

        )

        self.story\_part1.pack(pady=10)

        self.password\_label = **tk**.**Label**(self)

        self.password\_label.pack()

        self.password\_entry = **tk**.**Entry**(self, show="\*")

        self.password\_entry.pack(pady=5)

        self.continue\_button = **tk**.**Button**(

            self,

            text="Продолжить рассказ",

            command=self.**continue\_story**

        )

        self.continue\_button.pack(pady=10)

        self.story\_part2 = **tk**.**Text**(self, height=5, width=50)

        self.story\_part2.pack(pady=10)

    def **continue\_story**(self):

        password = self.password\_entry.**get**()

        if password == "0705":

            self.story\_part2.**delete**(1.0, **tk**.END)

            self.story\_part2.**insert**(

**tk**.END,

                "Продолжения не будет\n"

                "Я сказал что все конец\n"

            )

        else:

            self.story\_part2.**delete**(1.0, **tk**.END)

            self.story\_part2.**insert**(

**tk**.END,

                "Даже пароль не знаете\n"

            )

class **CheckboxTab**(**ttk**.**Frame**):

    def **\_\_init\_\_**(self, master):

**super**().**\_\_init\_\_**(master)

        self.letters = ['П', 'Р', 'И', 'В', 'ЕТ']

        self.check\_vars = []

        for letter in self.letters:

            var = **tk**.**IntVar**()

            cb = **tk**.**Checkbutton**(

                self,

                text=f"Буква '{letter}'",

                variable=var

            )

            cb.pack(anchor=**tk**.W)

            self.check\_vars.**append**(var)

        self.compose\_button = **tk**.**Button**(

            self,

            text="Составить слово",

            command=self.**compose\_word**

        )

        self.compose\_button.pack(pady=10)

        self.result\_text = **tk**.**Text**(self, height=2, width=30)

        self.result\_text.pack(pady=10)

    def **compose\_word**(self):

        selected\_letters = [

            self.letters[i]

            for i, var in **enumerate**(self.check\_vars)

            if var.get() == 1

        ]

        word = ''.**join**(selected\_letters)

        self.result\_text.**delete**(1.0, **tk**.END)

        if word:

            self.result\_text.**insert**(**tk**.END, f"Полученное слово: {word}")

        else:

            self.result\_text.**insert**(**tk**.END, "Выберите хотя бы одну букву")

class **RadioTab**(**ttk**.**Frame**):

    def **\_\_init\_\_**(self, master):

**super**().**\_\_init\_\_**(master)

        self.types = [

            ("Кортеж (tuple)", "(1, 2, 3)"),

            ("Строка (str)", "'Неизменяемая строка'"),

            ("Число (int)", "42"),

            ("Число (float)", "3.14"),

            ("Фрозенсет (frozenset)", "frozenset({1, 2, 3})")

        ]

        self.selected\_type = **tk**.**StringVar**()

        self.selected\_type.**set**(self.types[0][1])

        for text, value in self.types:

            rb = **tk**.**Radiobutton**(

                self,

                text=text,

                variable=self.selected\_type,

                value=value

            )

            rb.pack(anchor=**tk**.W)

        self.show\_button = **tk**.**Button**(

            self,

            text="Показать пример",

            command=self.**show\_example**

        )

        self.show\_button.pack(pady=10)

        self.result\_text = **tk**.**Text**(self, height=2, width=30)

        self.result\_text.pack(pady=10)

    def **show\_example**(self):

        example = self.selected\_type.**get**()

        self.result\_text.**delete**(1.0, **tk**.END)

        self.result\_text.**insert**(**tk**.END, f"Пример неизменяемого типа: {example}")

def **main**():

    root = **tk**.**Tk**()

    root.title("Программа с вкладками")

    root.geometry("500x400")

    notebook = **ttk**.**Notebook**(root)

    notebook.pack(fill='both', expand=True)

*# Создаем вкладки*

    tab1 = **PrimeButtonTab**(notebook)

    notebook.**add**(tab1, text="Простые числа")

    tab2 = **StoryTab**(notebook)

    notebook.**add**(tab2, text="Рассказ")

    tab3 = **CheckboxTab**(notebook)

    notebook.**add**(tab3, text="Буквы")

    tab4 = **RadioTab**(notebook)

    notebook.**add**(tab4, text="Типы данных")

    root.**mainloop**()

if \_\_name\_\_ == "\_\_main\_\_":

**main**()

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, мультимедиа, число

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Индивидуальное задание:

*# окно авторизации и регистрации, после успешной рег открывается кредитный кальултор, регистрация в отдельной файл*

import **tkinter** as **tk**

from **tkinter** import **ttk**, **messagebox**

import **hashlib**

import **os**

import **sys**

import **subprocess**

class **Registration**():

def **\_\_init\_\_**(self, appWindow):

self.appWindow = appWindow

self.appWindow.title("Sign up")

self.appWindow.geometry("400x450")

self.appWindow.resizable(False, False)

self.main\_frame = **ttk**.**Frame**(appWindow)

self.main\_frame.pack(padx=20, pady=20, fill=**tk**.BOTH, expand=True)

self.**widgets**()

script\_dir = **os**.path.**dirname**(**os**.path.**abspath**(\_\_file\_\_))

self.users\_file = **os**.path.**join**(script\_dir, "users\_file.txt")

self.appWindow.mainloop()

def **redirect\_to\_login**(self):

self.appWindow.destroy()

current\_dir = **os**.path.**dirname**(**os**.path.**abspath**(\_\_file\_\_))

login\_script = **os**.path.**join**(current\_dir, "login.py")

if **sys**.platform == "win32":

**subprocess**.Popen([**sys**.executable, login\_script])

else:

**subprocess**.**Popen**(["python3", login\_script])

def **widgets**(self):

title\_label = **ttk**.**Label**(

self.main\_frame,

text='Registration',

font=('Arial', 32)

)

title\_label.pack(pady=10)

**ttk**.**Label**(self.main\_frame, text='Username:').pack(anchor='center', pady=10, )

self.in\_username = **ttk**.**Entry**(self.main\_frame, width=30)

self.in\_username.pack(pady=5)

**ttk**.**Label**(self.main\_frame, text='Password:').pack(anchor='center', pady=10)

self.in\_password = **ttk**.**Entry**(self.main\_frame, width=30, show='\*')

self.in\_password.pack(pady=5)

**ttk**.**Label**(self.main\_frame, text='Repeat password:').pack(anchor='center', pady=10)

self.in\_rpassword = **ttk**.**Entry**(self.main\_frame, width=30, show='\*')

self.in\_rpassword.pack(pady=5)

btn = **ttk**.**Button**(

self.main\_frame,

text='Register',

command=self.**register\_user**

)

btn.pack(pady=20)

btn2 = **ttk**.**Button**(

self.main\_frame,

text='Sign in',

command=self.**redirect\_to\_login**

)

btn2.pack()

def **register\_user**(self):

username = self.in\_username.**get**().**strip**()

password = self.in\_password.**get**()

rpassword = self.in\_rpassword.**get**()

if not username:

**messagebox**.**showerror**('Error', 'Field can`t be empty')

return

if not password:

**messagebox**.**showerror**('Error', 'Field can`t be empty')

return

if password != rpassword:

**messagebox**.**showerror**('Error', 'Invalid password')

return

if self.**user\_exists**(username):

**messagebox**.**showerror**('Error', 'User already exist')

return

hashed\_password = self.**hash\_password**(password)

with **open**(self.users\_file, 'a') as f:

f.**write**(f'{username}:{hashed\_password} \n')

**messagebox**.**showinfo**('Already', 'You`re registered')

self.**redirect\_to\_login**()

def **hash\_password**(self, password):

return **hashlib**.**sha256**(password.encode()).**hexdigest**()

def **user\_exists**(self, username):

if not **os**.path.**exists**(self.users\_file):

return False

with **open**(self.users\_file, 'r') as f:

for line **in** f:

if line.**split**(':')[0] == username:

return True

return False

if \_\_name\_\_ == "\_\_main\_\_":

appWindow = **tk**.**Tk**()

app = **Registration**(appWindow)

import **tkinter** as **tk**

from **tkinter** import **ttk**, **messagebox**

import **hashlib**

import **os**

import **sys**

import **subprocess**

class **Login**:

def **\_\_init\_\_**(self, appWindow):

self.appWindow = appWindow

self.appWindow.title("Sign in")

self.appWindow.geometry("400x350")

self.appWindow.resizable(False, False)

self.main\_frame = **ttk**.**Frame**(appWindow)

self.main\_frame.pack(padx=20, pady=20, fill=**tk**.BOTH, expand=True)

script\_dir = **os**.path.**dirname**(**os**.path.**abspath**(\_\_file\_\_))

self.users\_file = **os**.path.**join**(script\_dir, "users\_file.txt")

self.**widgets**()

def **widgets**(self):

title\_label = **ttk**.**Label**(

self.main\_frame,

text="Sign in",

font=("Arial", 14)

)

title\_label.pack(pady=10)

**ttk**.**Label**(self.main\_frame, text="Username:").pack(anchor=**tk**.W, pady=10)

self.in\_username = **ttk**.**Entry**(self.main\_frame, width=30)

self.in\_username.pack(fill=**tk**.X, pady=5)

**ttk**.**Label**(self.main\_frame, text="Password:").pack(anchor=**tk**.W, pady=10)

self.in\_password = **ttk**.**Entry**(self.main\_frame, width=30, show="\*")

self.in\_password.pack(fill=**tk**.X, pady=5)

login\_button = **ttk**.**Button**(

self.main\_frame,

text="Sign in",

command=self.**login\_user**

)

login\_button.pack(pady=20)

back\_button = **ttk**.**Button**(

self.main\_frame,

text="Sign up",

command=self.**redirect\_to\_signup**

)

back\_button.pack()

def **hash\_password**(self, password):

return **hashlib**.**sha256**(password.encode()).**hexdigest**()

def **login\_user**(self):

username = self.in\_username.**get**().**strip**()

password = self.in\_password.**get**()

if not username:

**messagebox**.**showerror**("Error", "Input username")

return

if not password:

**messagebox**.**showerror**("Error", "Input password")

return

users\_file\_path = **os**.path.**abspath**(self.users\_file)

hashed\_password = self.**hash\_password**(password)

user\_found = False

with **open**(users\_file\_path, "r") as f:

for line **in** f:

parts = line.**strip**().**split**(':')

if **len**(parts) == 2:

if parts[0] == username and parts[1] == hashed\_password:

user\_found = True

break

if user\_found:

**messagebox**.**showinfo**("Already", "You`re signed!")

self.**redirect\_to\_calculator**()

else:

**messagebox**.**showerror**("Error", "Invalid data")

def **redirect\_to\_calculator**(self):

self.appWindow.destroy()

script\_dir = **os**.path.**dirname**(**os**.path.**abspath**(\_\_file\_\_))

calculator\_script = **os**.path.**join**(script\_dir, "credit.py")

if **sys**.platform == "win32":

**subprocess**.Popen([**sys**.executable, calculator\_script])

else:

**subprocess**.**Popen**(["python3", calculator\_script])

def **redirect\_to\_signup**(self):

self.appWindow.destroy()

script\_dir = **os**.path.**dirname**(**os**.path.**abspath**(\_\_file\_\_))

calculator\_script = **os**.path.**join**(script\_dir, "registration.py")

if **sys**.platform == "win32":

**subprocess**.Popen([**sys**.executable, calculator\_script])

else:

**subprocess**.**Popen**(["python3", calculator\_script])

if \_\_name\_\_ == "\_\_main\_\_":

appWindow = **tk**.**Tk**()

app = **Login**(appWindow)

appWindow.**mainloop**()

import **tkinter** as **tk**

from **tkinter** import **ttk**, **messagebox**

class **CreditCalculator**:

def **\_\_init\_\_**(self, appWindow):

self.appWindow = appWindow

self.appWindow.title("Credit Calculator")

self.appWindow.geometry("500x600")

main\_frame = **ttk**.**Frame**(appWindow)

main\_frame.pack(fill=**tk**.BOTH, expand=True, padx=10, pady=10)

self.**widgets**(main\_frame)

def **widgets**(self, parent):

title\_label = **ttk**.**Label**(

parent,

text="Credit Calculator",

font=("Arial", 14)

)

title\_label.pack(pady=10)

input\_frame = **ttk**.LabelFrame(parent, text="Credit settings")

input\_frame.pack(fill=**tk**.X, pady=8)

**ttk**.**Label**(input\_frame, text="Cash (rub):").pack(anchor=**tk**.W, pady=5)

self.amount\_entry = **ttk**.**Entry**(input\_frame)

self.amount\_entry.pack(fill=**tk**.X, pady=5)

**ttk**.**Label**(input\_frame, text="Interest rate (% years):").pack(anchor=**tk**.W, pady=5)

self.rate\_entry = **ttk**.**Entry**(input\_frame)

self.rate\_entry.pack(fill=**tk**.X, pady=5)

**ttk**.**Label**(input\_frame, text="Period (month):").pack(anchor=**tk**.W, pady=5)

self.term\_entry = **ttk**.**Entry**(input\_frame)

self.term\_entry.pack(fill=**tk**.X, pady=5)

**ttk**.**Label**(input\_frame, text="Type:").pack(anchor=**tk**.W, pady=5)

self.payment\_type = **tk**.**StringVar**(value="annuity")

payment\_frame = **ttk**.**Frame**(input\_frame)

payment\_frame.pack(anchor=**tk**.W, pady=5)

calculate\_button = **ttk**.**Button**(

parent,

text="Calculate",

command=self.**calculate\_payment**

)

calculate\_button.pack(expand=True, pady=10)

result\_frame = **ttk**.LabelFrame(parent, text='Result')

result\_frame.pack(fill=**tk**.BOTH, expand=True)

self.result\_text = **tk**.**Text**(

result\_frame,

height=15,

wrap=**tk**.WORD,

font=("Arial", 10)

)

self.result\_text.pack(fill=**tk**.BOTH, expand=True, padx=5, pady=5)

def **calculate\_payment**(self):

try:

amount = **float**(self.amount\_entry.**get**())

annual\_rate = **float**(self.rate\_entry.**get**())

monthly\_rate = annual\_rate / 100 / 12

term = **int**(self.term\_entry.**get**())

if amount <= 0 or annual\_rate <= 0 or term <= 0:

raise **ValueError**("Values must be positive")

self.result\_text.**delete**(1.0, **tk**.END)

if monthly\_rate == 0:

monthly\_payment = amount / term

else:

annuity\_ratio = (monthly\_rate \* (1 + monthly\_rate) \*\* term) / ((1 + monthly\_rate) \*\* term - 1)

monthly\_payment = amount \* annuity\_ratio

total\_payment = monthly\_payment \* term

overpayment = total\_payment - amount

result = (

f"Amount: {amount:,.2f} \n"

f"Interest rate: {annual\_rate:.2f}% per year\n"

f"Term: {term} months\n\n"

f"Monthly payment: {monthly\_payment:,.2f} \n"

f"Total payment: {total\_payment:,.2f} \n"

f"Overpayment: {overpayment:,.2f} \n\n"

)

self.result\_text.**insert**(**tk**.END, result)

balance = amount

for month in **range**(1, term + 1):

interest = balance \* monthly\_rate

principal = monthly\_payment - interest

balance -= principal

if balance < 0.01:

balance = 0

except **ValueError** as e:

**messagebox**.**showerror**("Error", f"Invalid data: {**str**(e)}")

except **ZeroDivisionError**:

**messagebox**.**showerror**("Error", "Term can`t be zero")

except **Exception** as e:

**messagebox**.**showerror**("Error", f"An error occurred: {**str**(e)}")

if \_\_name\_\_ == "\_\_main\_\_":

appWindow = **tk**.**Tk**()

app = **CreditCalculator**(appWindow)

appWindow.**mainloop**()

Вывод: Я рассмотрел способы работы с графикой с помощью модуля tkinter.