**4-я Встреча**

**Cодержание**

1. Знакомство с понятием оконных приложений
2. Знакомство с инструментами для их создания
3. Создание пустого окна
4. Знакомство с инструментами расположения виджетов
5. Использования виджетов в оконном приложение
6. Создание панели курсов валют
7. Компиляция py проекта

**Введение**

Сегодня мы с вами отойдём от консольных приложений и напишем оконное приложение-табло для курса валют. Познакомимся с элементами оконных приложения и ознакомимся с представлением цвета в компьютере.

**1-й урок**

Оконные приложения, вы все с ними работаете — это браузер, игра, наш редактор кода. На протяжение последний уроков мы с вами писали исключительно консольные приложения, поскольку весь их функционал был заключён в консоли, сегодня мы создадим своё первое оконное приложения. Многие программы на сегодняшний день используют графический интерфейс, который более интуитивен и удобен для пользователя, чем консоль. И с помощью языка программирования python также можно создавать графические программы.

Для этого в python есть множество уже написанных решений, чтобы создать своё оконное приложение. Нам не нужно будет читать документацию как пользоваться windows api или учить низкоуровневые языки программирования, для python существует множество библиотек которые позволяют делать оконные приложения. Давайте узнаем, что это за библиотеки:

1. PyQT — графический фреймворк Qt, написанный на языке С++
2. WxPython — python-обёртка для API разных ОС. Одно из самых популярных решений для создания оконных приложений на python. На нём написаны Google диск, десктопная версия, dropbox, BitTorrent
3. Tkinter, самый популярный конкурент WxPython

Мы будем пользоваться tkinter. Это довольно мощный инструмент, но по нему очень сложно найти обучающие материалы в отличие от других инструментов. Tkinter идёт в коробке с python, его не требуется устанавливать, нам нужно подключить его в наш скрипт командой import.

Tkinter (от англ. Tk interface) — кросс-платформенная графическая библиотека на основе средств Tk (широко распространённая в мире GNU/Linux и других UNIXподобных систем, портирована также и на Microsoft Windows), написанная Стином Лумхольтом (Steen Lumholt) и Гвидо ван Россумом. Входит в стандартную библиотеку Python.

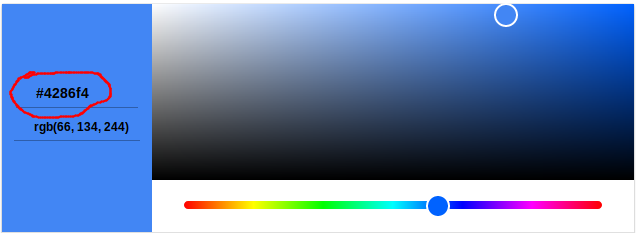
**Из чего состоит оконное приложение? (Кнопки, текстовые поля, выпадающие поля, текст, картинки)**

Все эти элементы называются виджетами, виджеты это маленькие кирпичики из которых состоит любое оконное приложение. Но приложения уже давно не выглядят как просто набор кнопок и текста, сейчас к процессу проектирования и создания приложений привлекают дизайнеров которые думают как и где лучше расположить виджеты и подбирают цветовую гамму для приложения. Вот это интересный момент, дело в том, что цвета в оконном приложение можно указывать просто используя их названия на английском, но такой подход не универсален. Дело в том наш код в любом случае преобразуется в набор нулей и единиц, а для каждого оттенка надо зарезервировать двоичный код на цветовой палитре, в зависимости от её типа там может содержаться от 4млн до 16млн цветов.

Двоичный код для красного цвета выглядит следующим образом 111111110000000000000000, теперь представьте что программист будет сидеть и вбивать 16 млн таких последовательностей. Но выход есть, для обозначения цвета можно использовать шестнадцатиричный код, который занимает меньше места, он используется в низкоуровневом программировании и компьютерной документации, поскольку в современных компьютерах минимальной адресуемой единицей памяти является 8-битный байт, значения которого удобно записывать двумя шестнадцатеричными цифрами. Такое использование началось с системы IBM/360, где вся документация использовала шестнадцатеричную систему, в то время как в документации других компьютерных систем того времени (даже с 8-битными символами, как, например, PDP-11 или БЭСМ-6) использовали восьмеричную систему.

Не пугайтесь, нам не нужно будет запоминать код каждого цвета или оттенка. Мы делаем всё проще, вбиваем в Google запрос color picker, нам всплывает подсказка, мы выбираем интересующий нас цвет и копируем его шестнадцатиричный код.

Этот код мы будем использовать для указания цветовой гаммы для элементов нашего оконного приложения. Давайте приступим к его созданию.



**Введение**

Сейчас мы приступим к созданию оконного приложения при помощи средств библиотеки tkinter. Познакомимся с позиционированием элементов в приложение и позиционированием приложений на экране.

**2-й урок**

Tkinter доступен в виде отдельного встроенного модуля, который содержит все необходимые графические компоненты - кнопки, текстовые поля и т.д.

Основным моментом в построении оконных приложений является создание окна. Затем мы добавляем туда виджеты или любые другие графические элементы. Поэтому создадим вначале простейшее окно. Для этого в папке fourth\_lesson создадим файл myTestWindow.py и реализуем в нём следующий скрипт:

Задание 1

|  |
| --- |
| from tkinter import \*    root = Tk()  root.title("Моё приложение")  root.geometry("500x500")    root.mainloop() |

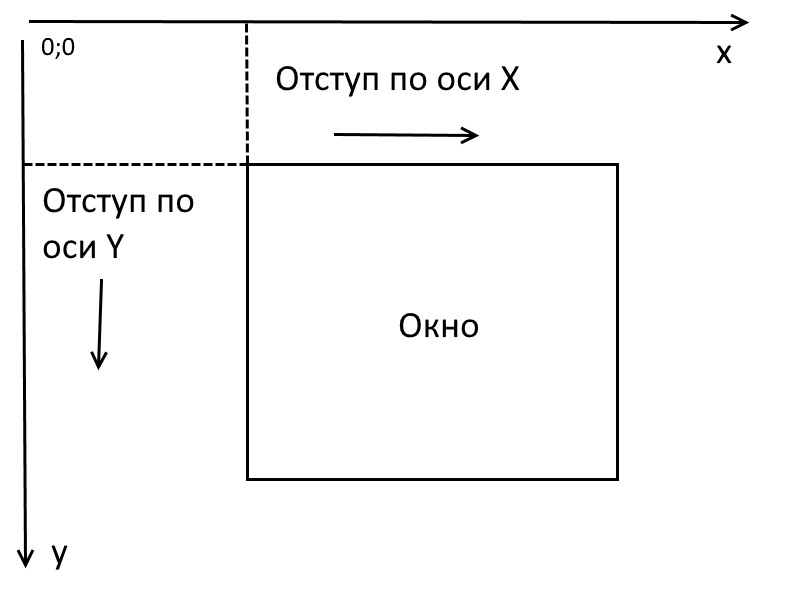
Первым делом импортируем tkinter и сразу указываем, что мы собираемся работать со всеми инструментами с этой библиотеки, чтобы не писать каждый раз tkinter перед вызовом его методов. Дальше мы объявляем переменную root и записываем в неё экземпляр класса Tk(), в дальнейшем мы обязательно познакомимся с этим подробнее, но пока не стоит углубляться. Просто считайте, что мы записали в переменную root основу нашего приложения(окно) и дальше при помощи различных команд будем туда добавлять элементы.

Первая команда — это функция title, в которую мы в кавычках передаём название главного окна строкой, затем мы указываем его размеры при помощи geometry, обратите внимание, что размер мы тоже указываем строкой. Ну а root.mainloop() запускает наше окно со всеми настройками, поэтому он всегда должен быть в конце.

По умолчанию наше окно появляется в левом верхнем углу, но если немного поменять строку которую мы передаём в метод geometry

Задание 2

|  |
| --- |
|  |

Теперь строка в методе geometry имеет следующий формат: "width x height + X + Y." X и Y, это самые обычные координаты на системе координат. Верхний левый угол экрана это точка 0:0 и когда мы указываем координаты, мы указываем отступ между левым верхним углом экрана и левым верхним углом нашего окна. То есть при запуске окно будет находиться на 500 пикселей вправо и на 500 пикселей вниз от верхнего левого угла экрана.

На окнах мы размещаем виджеты, виджеты — текст, кнопки, поля ввода, галочки. Давайте попробуем разместить текст на нашем окне.

Задание 3

|  |
| --- |
| lab = Label(root, text="Текст", bg="red", fg="gold", font="16")  lab.place(x=100, y=100) |

Мы создали переменную lab, в неё поместили экземпляр класса Label и передали аргументы, первый аргумент это переменная в которой хранится наше окно, второй аргумент это параметр текст, содержимое, того что мы хотим вывести. Дальше мы размещаем наш элемент при помощи метода place, происходит это точно так-же как и с окном, только теперь отступ идёт от левого верхнего края окна.

Бывает, что нам нужно изменить какой-то параметр у нашего шаблона, допустим текст или цвет, делается это следующим образом:

Задание 4

|  |
| --- |
| lab['text'] = "Поменяли"  lab['bg'] = "black" |

Так-же мы можем добавить кнопку, т. к. это тоже виджет

**//Запишите вызов кнопки одной строкой, а потом разложите на несколько строк, для читаемости**

Задание 5

|  |
| --- |
| btn = Button(text="Кнопка",  background="#555",  foreground="#ccc",  font="16"  ) |

У кнопки есть множество атрибутов, но нас интересует ещё один, очень важный атрибут, атрибут command. Это именно тот атрибут, который говорит кнопке что ей нужно что-то сделать. Но, что конкретно надо уточнить, давайте создадим функцию, которую как раз и будет выполнять кнопка

Задание 5

|  |
| --- |
| def changeLabel():  lab['text'] = "Вы нажали"  lab = Label(root, text="Текст", bg="#FF6666", fg="gold", font="16")  lab.place(x=100, y=100)  btn = Button(text="Hello",  background="#555",  foreground="#ccc",  font="16",  command=changeLabel  )    btn.place(x=200, y=200) |

Мы передали атрибуту commad значением, название той функции которую мы хотим вызвать при нажатии.

Но просто изменения текста это слишком просто, давайте сделаем так чтоб все наши клики на кнопку считались, давайте попробуем сделать это самостоятельно, для этого надо переписать нашу функцию.

Подсказываю что надо сделать:

1. На для начала заменить слово текст, значением 0. Кавычки мы никуда не убираем
2. В функции мы объявляем переменную счётчик (count)
3. Как мы переопределяли значение атрибута text, так-же мы можем записать его в переменную count=lab["text"], только учтите, значение хранится в текстовом виде и нам нужно преобразовать его в число
4. затем мы увеличиваем переменную count на 1
5. Затем, просто изменяем атрибут текст присвоив ему значение count

Ответ

|  |
| --- |
| def changeLabel():  count = int(lab['text'])  count = count + 1  lab['text'] = count |

Посмотрите на эту запись count = count + 1, выглядит не очень удобно и такие записи часто встречаются в коде, они длинные, поэтому существует укороченная запись, она называет инкремент, поэтому запись count = count + 1 можно заменить записью count+=1, если нам нужно уменьшать на одну, запись count = count - 1, то мы используем декремент, count-=1.

Последний штрих, давайте добавим фотографии на наше приложение

Задание 6

|  |
| --- |
| img = PhotoImage(file='pic.png')  pic = Label(root, image=img)  pic.place(x=0, y=0) |

**3-й урок**

Давайте теперь закрепим наши знаний о tkinter и сделаем графический интерфейс-табло для курса валют, поместим логотип банка, и выведем курс биткоина, рубля и евро на сегодняшний день. Для этого создадим новый файл, tablo.py.

Первым делом подключаем нужные нам библиотеки

Задание 7

|  |
| --- |
| from tkinter import \*  import requests  from bs4 import BeautifulSoup  from datetime import datetime |

Затем следует создать главное окно и указать для него настройки

Задание 8

|  |
| --- |
| window = Tk()  window.geometry("400x350+300+300")  window.title("Курс валют") |

Поместим логотип, чтоб наше приложение выглядело красивее.

Задание 9

|  |
| --- |
| img = PhotoImage(file='logo.png')  logo = Label(window, image=img)  logo.place(x=0, y=0) |

Дальше давайте разместим название нашего табло

Задание 10

|  |
| --- |
| lab = Label(window, text="Курс валют \n MAXIMUM банк", font="Arial 22")  lab.place(y=25, x=150) |

Картинка занимает у нас 150px, поэтому по x мы указываем точку 150px, делаем это для того, чтобы текст не наезжал на картинку.

Вы ещё обратили внимание на символ \n, это спецсимвол, он означает переход на следующую строку.

Теперь нам нужно использовать скрипт для получение курса с предыдущего урока

Задание 11

|  |
| --- |
| url = "http://www.cbr.ru/scripts/XML\_daily.asp?"  today = datetime.today()  today = today.strftime("%d/%m/%Y")  payload = {"date\_req" : today}  responce = requests.get(url, params=payload)  xml = BeautifulSoup(responce.content, "lxml")  def getCourse (id):  return xml.find("valute", {'id': id}).value.text |

После этого надо указать дату, на которую актуален наш курс.

Задание 12

|  |
| --- |
| course\_info = Label(window, text="Курс на: " + today.replace('/', '.'), fg="black", font="Arial 18")  course\_info.place(y=150, x=90) |

Мы используем переменную today, в которую мы записывали текущую дату для запросу к api центробанка. Но для корректного запроса, мы указывали, что поля даты у нас между собой разделяют слэши, это американский формат записи и на нашем табло он будет смотреться непривычно. Поэтому мы использовали метод replace и в строке даты заменили все слеши на точки.

Добавим курс доллара и евро на табло.

Задание 13

|  |
| --- |
| usd\_course = Label(window, text="$ " + getCourse("R01235"), font="Arial 16")  usd\_course.place(y=190, x=100)  eur\_course = Label(window, text="€ " + getCourse("R01239"), font="Arial 16")  eur\_course.place(y=230, x=100) |

После этого всего, можем запустить и посмотреть что у нас вышло.

На прошлом уроке, мы научились воспроизводить звук при помощи python. Давайте реализуем озвучку в нашем табло для курсов валют, начнём с подключения необходимых библиотек.

Задание 14

|  |
| --- |
| import os  from gtts import gTTS  from pygame import mixer  import time |

После того как библиотеки подключили, давайте создадим функцию которая собственно и будет озвучивать. Инициализируем в ней mixer, который позволяет запускать звук внутри программы и укажем название файла в который будем сохранять наши аудиодорожки, чтобы потом их проигрывать.

Задание 15

|  |
| --- |
| def voice\_course():  mixer.init()  mp3\_name = "audio.mp3" |

Дальше нам нужно будет сформировать строчки которые будет озвучивать наша программа, давайте запишем все эти строчки в единый массив и в цикле будем вызывать и озвучивать каждую.

Задание 16

|  |
| --- |
| def voice\_course():  mixer.init()  mp3\_name = "audio.mp3"  **valutes = [**  **"за 1 доллар дают {} рублей".format(getCourse("R01235")[:-2]),**  **"за 1 евро дают {} рублей".format(getCourse("R01239")[:-2])**  **]** |

В итоге у нас в массиве будут 2 строчки следующего содержания:

1. за 1 доллар дают 63,58 рублей
2. за 1 евро дают 71,65 рублей

Мы с вами обрезали 2 последних символа, чтобы озвучка была лаконичнее и красивее, теперь давайте запустим цикл, передадим эти строчки в библиотеку для озвучки, сохраним итог в аудиофайл и вызовем аудиофайл при помощи mixer.

Задание 17

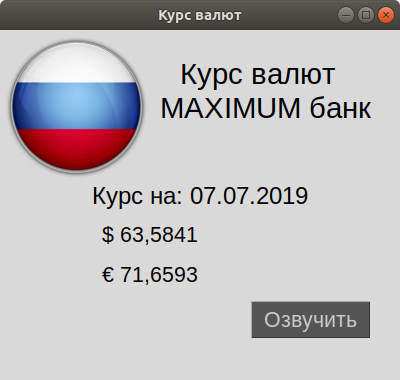
|  |
| --- |
| def voice\_course():  mixer.init()  mp3\_name = "audio.mp3"  valutes = [  "за 1 доллар дают {} рублей".format(getCourse("R01235")[:-2]),  "за 1 евро дают {} рублей".format(getCourse("R01239")[:-2])  ]  **for valute in valutes:**  **tts=gTTS(text=valute, lang='ru')**  **tts.save(mp3\_name)**  **mixer.music.load(mp3\_name)**  **mixer.music.play()**  **time.sleep(3)** |

В конце цикла мы добавим небольшую задержку, чтобы у нас не переплеталась озвучка разных курсов, использовали для этого библиотеку time из которой вызвали функцию sleep() и внутри неё указали сколько секунд наша программа ожидает.

Теперь давайте добавим кнопку при нажатию на которую, будет вызываться наша функция

Задание 18

|  |
| --- |
| voice\_btn = Button(text="Озвучить",  background="#555",  foreground="#ccc",  font="Arial 16",  command=voice\_course  )  voice\_btn.place(y=270, x=250) |

****

**Заключение**

По итогу мы с вами получим именно такое окно, где при нажатии у нас начинают озвучиваться актуальные курсы валют.

Сегодня мы с вами научились создавать графические приложения на python. На следующем уроке мы с вами познакомимся с одним их основных подходов к программированию, который использую в IT-компания промышленные программисты для создания реальных продуктов.