Лабораторная работа № 3

**Разработка усложненного Python-проекта**

Цель работы: изучение свойств и методов использования дополнительного набора компонентов, применяемых для формирования графического пользовательского интерфейса в Python-приложениях.

*Введение*

В дополнение к тем виджетам, которые рассмотрены в лабораторных работах 1 и 2, модуль tkinter включает ряд визуальных компонентов, широко используемых для формирования пользовательских интерфейсов программных приложений.

**SpinBox**

Данный виджет используется для задания конкретного численного значения из заданного перечня. Для создания спинбокса используется класс Spinbox:

spin = Spinbox(root, from\_=0, to=100)

Перечень значений можно задавать двумя способами:

1. указанием граничных значений диапазона;
2. перечислением возможных значений.

Выше предстален первый вариант задания, где параметры from\_=0, to=100 задают границы диапазона.

Разрешается просто перечислить для Spinbox диапазон возможных значений следующим образом:

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

Тогда виджет будет отображать только эти 3 числа: 3, 8 и 11. Кроме того, можно указать ширину виджета с помощью параметра width.

Также можно задать значение по умолчанию для Spinbox. Для этого надо передать это значение параметру textvariable следующим образом:

var = IntVar()

var.set(36)

spin = Spinbox(window, from\_=0, to=100, width=5, textvariable=var)

Параметр textvariable определяет переменную (в данном случае var), через которую считывается или устанавливается текущее значение виджета.

**Progressbar**

Данный виджет используется для визуализации динамики протекания указанных процессов. Чтобы создать данный виджет, используется класс progressbar:

from tkinter.ttk import Progressbar

. . .

bar = Progressbar(root, length=200, orient = HORIZONTAL, mode = 'determinate')

Необязательный параметр orient имеет два варианта значения: HORIZONTAL (“horizontal”) и VERTICAL ("vertical"). По умолчанию установлен вариант HORIZONTAL.

Необязательный параметр mode имеет два варианта значения: “determinate” и "indeterminate". По умолчанию установлен вариант “determinate”. В этом случае работа виджета представляет собой разворачивающуюся ленту. Во втором случае динамика отображается движением ползунка по виджету.

Установить значение progressbar можно таким образом:

bar['value'] = 70

**всплывающие окна**

При работе с модулем tkinter можно использовать всплывающее окно messagebox, загружаемое в приложение следующим образом:

from tkinter import messagebox

Использование варианта окна messagebox.askyesno приводит к повлению окна-вопроса с двумя кнопками: **Да, Нет**. Нажатие "Да" в диалоговом окне возвращает в программу True, "Нет" вернет False (также как закрытие окна через крестик). Таким образом в коде можно обработать выбор пользователя:

answer = messagebox.askyesno(title="Вопрос", message="Перенести данные?")

т.е. на основе ответа можно управлять работой программы.

Опции title и message являются позиционными, так что можно указывать только значения, т.е. упрощенный вариант работы программы имеет вид:

messagebox.askyesno("Вопрос", "Перенести данные?")

Подобные окна генерируются также при использовании функции askokcancel() с надписями на кнопках "ОК" и "Отмена", askquestion() (возвращает не True или False, а строки 'yes' или 'no'), askretrycancel() ("Повторить", "Отмена"), askyesnocan-cel() ("Да", "Нет", "Отмена").

Другую группу составляют окна с одной кнопкой, которые служат для вывода сообщений различного характера. Это showerror(), showinfo() и showwarning().

**кнопка SpeedButton**

Во многих современных системах программирования средства создания GUI включают кнопку, которая может фиксироватться в двух позициях: нажата, отжата, - т.е. она может использоваться для отображения двоичных значений. При этом такие кнопки также могут образовывать радиогруппы.

В модуле tkinter подобные кнопки имитируются стандартной кнопкой Radiobatton в режиме indicatoron = 0.

r\_var = BooleanVar()

r\_var.set(0)

r1 = Radiobutton(text='Переключение', indicatoron = 0, variable=r\_var, value=0)

**контейнер Canvas**

Canvas является виджетом-контейнером, в который можно вставлять растровые рисунки. С помощью функций из пакетов Image и ImageTk из PIL формируется объект-изображение для загрузки в контейнер

from tkinter import \*

from PIL import ImageTk, Image

FILENAME = "pos.jpg" # файл с графическим изображением

root = Tk()

root.geometry('700x500')

c = Canvas(root, width=400, height=200, bg="white")

c.place(x = 200, y = 200)

img = ImageTk.PhotoImage(file = FILENAME)

c.create\_image(50, 10, image=img, anchor="nw")

root.mainloop()

В контейнере задаются координаты опорной точки (50, 10), относительно которой производится привязка самого рисунка на основе параметра anchor (рис. 3.1): N, NE, E, SE, SW, W, NW или CENTER.

NW

N

NE

E

SE

W

SW

CENTER

Рис. 3.1

У самого контейнера также задается положение и размер.

*Выполнение лабораторной работы*

1. В соответствии с вариантом выбрать задание на программную разработку:

В(1)-17. На форме располагаются: таблица размером 3х3, два радионабора для выбора текущей ячейки, компонент Scale, редактор Text. Все ячейки таблицы заполнены названиями различных предметов (птиц, рыб, животных и т.п.). Компонент Scale имеет размер ячейки и перемещается синхронно активной ячейке. При выборе ячейки на нем отображается изображение текущего предмета, а название предмета дублируется в редакторе Text. Кнопка Close заканчивает программу.

II. Результат разработки продемонстрировать преподавателю.

Содержание отчета:

1. Титульный лист установленной формы;
2. Задание на выполнение лабораторной работы;
3. Структура проекта;
4. Блок-схемы процедур, содержащих разветвления и/или циклы

5) Листинг программы.

Лабораторная работа № 4

**Управление временем в Python-проектах**

Цель работы: научить студентов использовать средства управления временем в программных Python-проектах.

*Введение*

При создании динамического изображения в программных проектах требуется механизм управления временем. Модуль tkinter позволяет реализовать это управление двумя способами:

1. Метод after() откладывает выполнение какого-либо кода на заданный промежуток времени. Его синтаксис имеет вид:

<окно>.after(<временная задержка в мс>[, <функция>]),

т.е. выполнение <функции> задерживается на время, указанное в (<временной задержке в мс>.

Периодическую задержку можно организовать:

а) через цикл с заранее известным числом повторений. Здесь также возможны два варианта.

Обобщенный алгоритм первого варианта (рис. 4.1):

Задание начальных значений

Цикл по конечному параметру

Тело цикла

<окно>.after(<время задержки>)

Рис. 4.1

Для того, чтобы изображение реконфигурировалось на каждом шаге цикла, необходимо после очередного перестроения активизировать метод <окно>.up-date\_idletasks().

Обобщенный алгоритм второго варианта представлен на рис. 4.2:

Задание начальных значений

Цикл по конечному параметру

<окно>.after(<время задержки>, <команда>)

Рис. 4.2

При этом предполагается, что аргумент <команда> метода after запускает некоторую функцию, находящуюся вне цикла.

б) цикл на основе рекурсивной структуры. Для его запуска и остановки требуется глобальная переменная-переключатель, которая устанавливается в соответствующие значения отдельными процедурами (рис. 4.3а и 4.б). А переключатель управляет работой процедуры таймера (рис. 4.3в).

а)

Перевод выключателя в состояние «Включено»

Запуск процедуры-таймера

к

Перевод выключателя в состояние «Выключено»

Выключение часов

к

Рис. 4.3

Включение часов

Процедура-таймер

Выполнение тела цикла

<окно>.after(<время задержки>, <запуск процедуры-таймера>)

Выключатель = «Включено»

Да

к

Нет

б)

в)

Этот вариант не требует использования метода update\_idletasks() для корректного отображения динамики

1. Метод sleep() также позволяет создавать временные задержки требуемой длительности. Его синтаксис имеет вид:

time.sleep(<задержка в сек.>)

Из записи видно, этот метод принадлежит модулю time, т.е. это модуль должен быть импортирован в начале скрипта.

Метод sleep() также требует для корректной работы поддержки методом update\_idletasks().

Изучение методов работы со временем эффективней всего проводить на примере динамической графики, поэтому ниже рассматриваются средства создания простейших изображений в рамках модуля tkinter.

Для создания подобных изображений в tkinter используется контейнер Canvas (канва), основы создания которого рассмотрены в предыдущей лабораторной работе.

На канву можно выводить текст командой, синтаксис которой имеет вид:

<имя канвы>.create\_text(<координаты опорной точки>, text="<отображаемый текст>", [<параметры шрифта>])

Ниже приведен пример размещения на канве трех строк шрифтом Verdana, размер которого равен 14 пунктам. Текст выравнивается по центру относительно точки привязки с координатами х=100, у=100.

c.create\_text(100, 100, text="Hello World,\nPython\nand Tk", justify=CENTER, font="Verdana 14 italic")

Основные параметры текста:

* fill – цвет текста;
* font – шрифт, включает название шрифта, его размер в пунктах и стиль;
* anchor – выравнивание текста относительно точки привязки (см. предыдущую лабораторную работу);
* activefill – цвет текста при наведении на него мыши.

Для того, чтобы в дальнейшем можно было менять параметры данного фрагмента текста, ему должно быть присвоено имя переменной.

Прямоугольник в простейшем случае формируется командой:

<имя канвы>. create\_rectangle(<координаты левого верхнего угла>, < координаты правого нижнего угла >, [<параметры прямоугольника>])

Основные параметры текста:

* outline - цвет контура прямоугольника;
* fill - цвет заливки;
* activefill - цвет заливки при наведении на прямоугольник мыши;
* activeoutline - цвет контура прямоугольника при заходе на него мыши;
* width - ширина контура фигуры в пикселях.

Графические средства tkinter позволяют создавать ломаную линию произвольной конфигурации. Одиночный отрезок создается командой c.create\_line(10, 10, 190, 50), т.е.по синтаксису:

<имя канвы>.create\_line(<координаты конца 1>, < координаты конца 2>)

Ломаная линия произвольно длины создается по синтаксису:

<имя канвы>.create\_line(<массив точек>)

Основные параметры линии:

* fill - цвет линии;
* activefill - цвет линии при наведении на прямоугольник мыши;
* width - ширина линии фигуры в пикселях;
* smooth – сглаживание линии.

Ниже приведен пример создания линии, когда выделяется отдельно каждая точка излома:

c.create\_line([300,80],[400,80],[450,75],[450,200],[300,180],[330,160])

Но такой же результат будет получен и при следующем задании линии:

c.create\_line(300,80,400,80,450,75,450,200,300,180,330,160)

Параметр smooth по умолчанию установлен в значение 0, что соответствует излому линии в каждой точке перегиба. При smooth=1 в точках перегиба линия сглаживается.

Метод create\_polygon контейнера Canvas близок по своему действию к предыдущему методу, но здесь первая и последняя точки линии автоматически соединяются, а замкнутые поверхности закрашиваются установленным цветом. Поэтому у данного метода параметр fill определяет цвет заливки замкнутого контура, а outline – цвет самого контура. Актуален также параметр smooth.

Окружности на канве создаются методом create\_oval, у которого в начале параметров указываются координаты левого верхнего и правого нижнего углов описывающего окружность прямоугольника:

c.create\_oval(50, 10, 150, 110, width=2)

Все параметры окружности аналогичны по своему назначению параметрам прямоугольника или полигона.

Рассмотренные фигуры можно перемещать по пространству канвы с помощью метода move:

<канва>, move(<имя фигуры>, <смещение по оси Х>, <смещение по оси У>)

Из синтаксиса метода следует, что для реализации такого преобразования у каждой фигуры необходим идентификатор, который должен присваиваться ей в момент создания.

*Выполнение лабораторной работы*

I. В соответствии с заданием выбрать вариант реализации программного проекта. Во всех вариантах задания графический объект создается на базе контейнера Canvas. На канве создается статическое изображение с динамическим компонентом-рамкой. Толщина рамки три пикселя.

В(1)-17. На пространстве формы изображен календарь за май текущего года с вертикальным расположением недель. Дни недели подписаны. Метка в виде шестиугольника циклически движется по датам с дискретом времени 0.6 сек. Запуск движения – кнопка «Пуск», остановка – команда главного меню, что приводит к установке рамки на первую дату.

Содержание отчета:

1. Титульный лист установленной формы;
2. Задание на выполнение лабораторной работы;
3. Структура проекта;
4. Блок-схемы процедур, содержащих разветвления и/или циклы

5) Листинг программы.

Лабораторная работа № 5

**Графика на основе канвы**

Цель работы: изучение стандартных средств канвы модуля tkinter по программному синтезу статических и динамических графических изображений.

*Введение*

Графический пользовательский интерфейс предполагает широкое использование динамики, что повышает информативность взаимодействия, расширяет изобразительные возможности интерфейса, снижает требования к квалификации пользователя. Строго говоря, формирование такого инструментария должно происходит с участием специалистов по дизайну и эргономике, но в подавляющем большинстве случаев весь процесс осуществляется только силами прикладных программистов, использующих графические средства языка реализации программной системы.

Ранее уже фрагментарно рассматривались возможности канвы модуля tkinter, представляющей графическое полотно заданного размера, состоящее из отдельных пикселей. Это полотно является объектом класса Canvas. У данного класса есть свои свойства и методы, с помощью которых можно создавать графические изображения. Для этого каждый пиксель может принимать один из 224 возможных цветовых оттенков.

При размещении геометрических примитивов и других объектов указываются их координаты на холсте в пикселях. Точкой отсчета является верхний левый угол канвы (рис. 5.1). Поскольку единицей размера является пиксель, то при изменении разрешения меняются и размеры рисунка.

Обычно начальными значениями координат являются (0, 0).

(0,0))

(210,180)

Х

У

Рис. 5.1

К рассмотренным ранее параметрам отрезка линии необходимо также добавить свойства:

* arrow – указание на формирование отрезка-стрелки. Значениями параметра arrow могут быть LAST, FIRST, BOTH. FIRST – стрелка направлена вперед, LAST – назад, BOTH – двунаправленная стрелка;
* arrowshape=” 1п 2п 3п” - параметры наконечника стрелки (рис. 5.2). Например, arrowshape="10 20 10";

⮝

1п

2п

3п

Рис. 5.2

* dash – параметры пунктира, например dash=(10,2) - тире длиной 10 пикселей и пробел 2 пикселя. Данная установка срабатывает только при толщине линии в 1 пиксель.

Ниже приведен пример создания пунктирного отрезка стрелки:

c.create\_line(100, 180, 100, 60, fill='green', width=5, arrow=LAST, dash=(10,2), activefill='lightgreen', arrowshape="10 20 10")

Новыми получаются фигуры при использовании метода create\_arc(). В зависимости от значения опции style можно получить сектор (по умолчанию), сегмент (CHORD) или дугу (ARC). Также как в случае create\_oval() координаты задают прямоугольник, в который вписана окружность (или эллипс), из которой "вырезают" сектор, сегмент или дугу. Опции start присваивается градус начала фигуры, extent определяет угол поворота.

c.create\_oval(10, 10, 190, 190, fill='lightgrey', outline='white')

c.create\_arc(10, 10, 190, 190, start=0, extent=45, fill='red')

c.create\_arc(10, 10, 190, 190, start=180, extent=25, fill='orange')

c.create\_arc(10, 10, 190, 190, start=240, extent=100, style=CHORD, fill='green')

c.create\_arc(10, 10, 190, 190, start=160, extent=-70, style=ARC, outline='darkblue', width=5)

В Tkinter существует два способа "пометить" фигуры, размещенные на холсте, – это идентификаторы, рассмотренные еще в предыдущей лабораторной работе, и теги. Первые всегда уникальны для каждого объекта. Два объекта не могут иметь одни и тот же идентификатор. Теги не уникальны. Группа объектов на холсте может иметь один и тот же тег. Это дает возможность менять свойства всей группы. Отдельно взятая фигура на Canvas может иметь как идентификатор, так и тег.

Кроме рассмотренного ранее метода move модифицировать графические объект также можно на основе методов:

* itemconfig изменяет указанные свойства объектов (<канва>.itemconfig(<имя фигуры>,{<парметр>=<новое значение>});
* coords изменяет координаты (ими можно менять и размер объекта) (<канва>.coords(<имя фигуры>, {<новая координата>}). Если указывается только идентификатор или тег, то coords() возвращает текущие координаты объекта.

Предположим, что происходит перемещение прямоугольника командой

t = c.create\_rectangle(x1, 20, x1+40, 60, outline='white', width=3)

Оператор m=c.coords(t) будет формировать на каждом шаге перемещения массив из четырех значений-координат этого прямоугольника. Доступ к каждой координате возможен по индексу в списке.

Удаление конкретной геометрической фигуры реализуется с использование метода delete(). Если же требуется очистить все пространство канвы, то включается ее метод ALL.

*Выполнение лабораторной работы*

В соответствии с вариантами разработать проект, реализующий динамическое изображение графическими средствами канвы и показать его преподавателю.

В(1)-17. Стилизованное изображение шашки проходит туда и обратно по главной диагонали доски с дискретом 0.7 сек; Запуск движения – команда контекстного меню.

Содержание отчета:

1. Титульный лист установленной формы;
2. Задание на выполнение лабораторной работы;
3. Структура проекта;
4. Блок-схемы процедур, содержащих разветвления и/или циклы

5) Листинг программы.