Графический интерфейс пользователя

Модуль tkinter

tkinter – графический модуль, который входит в стандартный комплект языка программирования Питон и позволяет программировать не только в объектном, но и в процедурном стиле.

```
>>> import tkinter
>>> tk = tkinter.Tk()
```

```
>>> from tkinter import *
>>> tk = Tk()
```

http://younglinux.info/tkinter.php - для начала

http://ru.wikiversity.org/wiki/Курс_по_библиотеке_Tkinter_языка_Python

Tcl/Tk

Скриптовый язык Tcl (от англ. Tool Command Language) разработан в 1988 году Джоном Оустерхаутом, который в то время работал в университете Бэркли.

«тикль-ток»

Области применения языка:

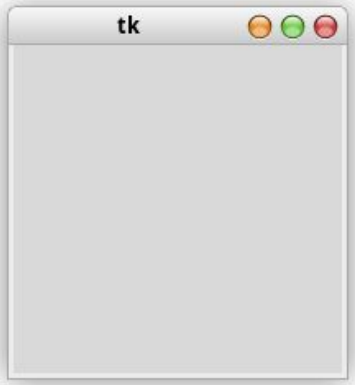
1) быстрое прототипирование,
2) создание графических интерфейсов для консольных программ,
3) встраивание в прикладные программы,
4) тестирование,
5) системная интеграция.

Язык состоит из команд и их параметров, разделенных пробелами. Все данные – текстовые. Ключевых слов нет.

Tk – графическая библиотека, написанная для Tcl.

Главное окно

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



Конструктор – функция, которая создает и возвращает объект.
Объект – то, что имеет методы.

tk – объект главного окна.

Вместе с созданием главного окна запускается цикл обработки сообщений. Сообщения поступают в программу от пользователя и от операционной системы.

Виджеты — «кирпичи» графической программы

Виджеты –
это базовые блоки для создания графического интерфейса программы.

Основные виджеты:

Toplevel — главное окно
Button — кнопка
Label — метка, текст
Entry — поле ввода строки текста
Text — поле ввода л.бого количества текста
Listbox - это список с выбором.
Frame — рамка
Checkbutton - отметить „галочкой“ пункт в окне

Алгоритм создания программы:

1. Создание окна приложения
2. Создание виджета
3. Изменение атрибутов виджета
4. «Сборка» виджета

…

2-4 повторить необходимое число раз

5. Сборка окна.

Toplevel - окно верхнего уровня.

Обычно используется для создания многооконных программ, а также для диалоговых окон.

Методы виджета. Эти же методы могут быть использованы для корневого (root) окна.

`title` - заголовок окна пример: **`root.title(`Заголовок окна`)`**

`resizable` - может ли пользователь изменять размер окна. Принимает два аргумента - возможность изменения размера по горизонтали и по вертикали. Без аргументов возвращает текущее значение.

`geometry` - устанавливает геометрию окна в формате ширина**x**высота+x+y (пример: geometry("600x400+40+80") - поместить окно в точку с координатам 40,80 и установить размер в 600x400).

Размер или координаты могут быть опущены:

(geometry("600x400") - только изменить размер,
geometry("+40+80") - только переместить окно).

Button — кнопка.

Label - это виджет, предназначенный для отображения какой-либо надписи без возможности редактирования пользователем.

Методы виджетов Button и Label

text - какой текст будет отображён на кнопке (в примере - ок)

width,height - соответственно, ширина и длина кнопки.

bg - цвет кнопки (сокращенно от background, в примере цвет - чёрный)

fg - цвет текста на кнопке (сокращённо от foreground, в примере цвет - красный)

font - шрифт и его размер (в примере - arial, размер - 14)

```
from tkinter import *
root=Tk()
bt1=Button(root,text='ok',width=25,height=5,
           bg='black',fg='red',)
bt1.pack()
root.mainloop()
```

Задание свойств виджета

Задавать свойства виджета можно тремя способами.

1. Во время создания объекта с помощью именованных параметров:

```
btn1 = Button(root, text='Медвед', bg='blue', fg='white')
```

2. После создания объекта, рассматривая объекты как словари, а название опции как ключ:

```
btn1["text"] = 'Медвед'
btn1["bg"] = 'blue'
btn1["fg"] = 'white'
```

3. После создания объекта при помощи его метода config():

```
btn1.config(text='Медвед', bg='blue', fg='white')
```

Окно с кнопкой

```
import tkinter
from tkinter import *
root = Tk()
root.geometry("300x200+40+80")

btn1 = Button(root)
btn1["text"] = 'Левая'
btn1["bg"] = 'blue'
btn1["fg"] = 'white'
btn1.place(x = 50, y = 50)

btn2 = Button(root, text='Правая', bg='green', fg='white')
btn2.place(x = 150, y = 50)

root.mainloop()
```

Связывание действий

```
import tkinter
from tkinter import *
root = Tk()
btn1 = Button(root, text='Привет', bg='blue', fg='white')
btn2 = Button(root, text='Медвед', bg='blue', fg='white')
btn1.place(x = 100, y = 50)
btn2.place(x = 200, y = 50)

def f1():
    root.title(btn1["text"])

def f2():
    root.title(btn2["text"])

btn1['command'] = f1  # здесь можно присвоить только
btn2['command'] = f2  # ф-цию без параметров

root.mainloop()
```

Правильное проектирование

Принцип проектирования "Разделение ответственности":
один модуль решает одну задачу.

Есть две задачи:

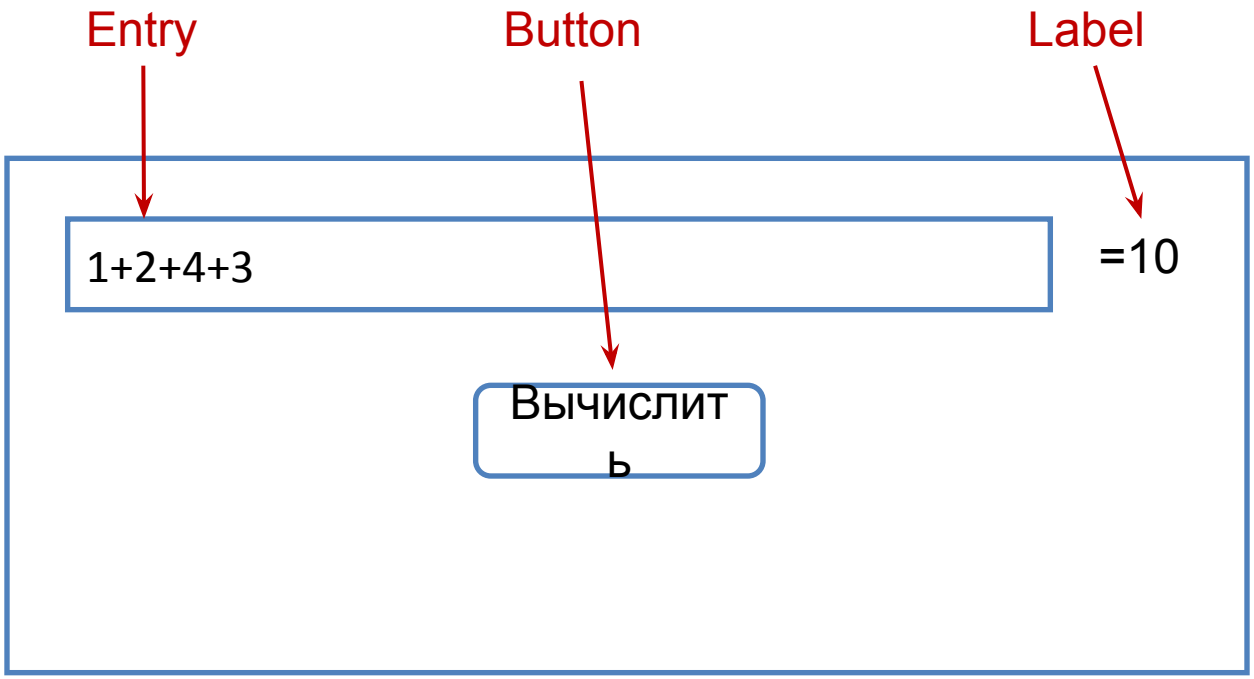
1) построение GUI,
2) выполнение расчетов.

Вывод: должно быть два модуля.

Ответственность разделена

```
def f(b, r):
    r.title(b["text"])
##################################################
import tkinter
from tkinter import *
root = Tk()
btn1 = Button(root, text='Привет', bg='blue', fg='white')
btn2 = Button(root, text='Медвед', bg='blue', fg='white')
btn1.place(x = 100, y = 50)
btn2.place(x = 200, y = 50)

btn1['command'] = lambda: f(btn1, root)
btn2['command'] = lambda: f(btn2, root)

root.mainloop()
```

Калькулятор



```
def calc(s):
    lst = s.split('+')
    summa = 0
    for x in lst:
        summa += int(x)
    return summa
#----------------------------------------
import tkinter
from tkinter import *
root = Tk()

root['width'] = 400
root['height'] = 200

entry = Entry(root, width = 25)
entry.place(x = 20, y = 20)

btn = Button(root, text='Вычислить')
btn.place(x = 20, y = 50)

lbl = Label(root)
lbl.place(x = 200, y = 20)

def btn_command():
    res = calc(entry.get())
    lbl['text'] = '= ' + str(res)

btn['command'] = btn_command
root.mainloop()
```

Код калькулятора

Самостоятельно

1. В первой программе (Превед-Медвед) сделать так, чтобы при нажатии на любую из двух кнопок надписи на них менялись местами. Разделение ответственности сохранить.
2. Спроектировать пользовательский интерфейс и написать программу для перевода чисел в р-ичную систему счисления. Число и основание системы вводит пользователь.

Виджеты и переменные

Метки

Виджеты (управляющие элементы) – видимые объекты, воспринимающие действия пользователя, и передающие информацию ему.

Метки (или надписи) — содержат строку (или несколько строк) текста и служат для информирования пользователя.

```
label1 = Label(root, text = "Это метка!\nИз двух строк.",
         font = "Arial 18")

label1.place(x = 0, y = 0)
```

Текстовое поле

В него пользователь может ввести только одну строку текста.

```
entry = Entry(root, width=20, bd=3)
```

width – ширина поля в символах
bd – ширина границы

Многострочное текстовое поле

Позволяет ввести несколько строк текста.

```
text = Text(root, width=30, height=5,
        font="Verdana 12",
        wrap=WORD)
```

Свойство *wrap* в зависимости от своего значения позволяет переносить текст, вводимый пользователем, либо по символам, либо по словам, либо вообще не переносить, пока пользователь не нажмет Enter.

"Переменные" библиотеки tkinter

Переменные – это объекты, которые дают дополнительные возможности для управления виджетами.
Переменные бывают четырех типов:

StringVar() – строковые;
IntVar() – целые;
DoubleVar() – вещественные;
BooleanVar() – логические.

Переменные имеют методы set() и get() для установки и получения их значений.
Связываются переменные с виджетами через свойство виджета variable или textvariable.

```
v = StringVar()
v.set("Привет!")
ent = Entry(root, width=20, bd=3, textvariable = v)
```

Одну переменную можно связать с несколькими виджетами.

Радиокнопки (переключатели)

Радиокнопка никогда не используется в одиночку. Их используют группами, при этом в одной группе может быть «включена» лишь одна кнопка.

```
var1 = IntVar()
var1.set(1)

var2 = IntVar()
var2.set(0)

# эти кнопоки объединяет в группу переменная var1
r1 = Radiobutton(root,text="Первая", variable=var1,value=0)
r2 = Radiobutton(root,text="Вторая", variable=var1,value=1)
r3 = Radiobutton(root,text="Третья",   variable=var1,value=2)
r4 = Radiobutton(root,text="Четвертая",variable=var1,value=3)
for i in r1,r2,r3,r4: i.pack()
```

Сколько переменных, столько и групп. Внутри группы кнопки различаются по значению переменной.

Радиокнопки (переключатели)

```
def change():
    label['text'] = var1.get()
#-----------------------------------------
from tkinter import *
root = Tk()

var1 = IntVar()
var1.set(1)

# эти кнопоки объединяет в группу переменная var1
r1 = Radiobutton(root,text="Первая", variable=var1,value=0)
r2 = Radiobutton(root,text="Вторая", variable=var1,value=1)
r3 = Radiobutton(root,text="Третья",  variable=var1,value=2)
r4 = Radiobutton(root,text="
Четвертая",variable=var1,value=3)

button = Button(text="Изменить", command=change)
label = Label(width=20, height=5, font = 18)

for i in r1,r2,r3,r4: i.pack()

button.pack()
label.pack()
root.mainloop()
```

Флажки

Объект checkbutton предназначен для выбора одного и более не взаимоисключающих пунктов.
В отличие от радиокнопок, каждый флажок привязывается к отдельной переменной, значение которой определяется свойствами onvalue (включено) и offvalue (выключено).

```
v1 = IntVar()
v2 = IntVar()

check1 = Checkbutton(root, text="Первый флажок",
         variable=v1, onvalue=1, offvalue=0)

check2 = Checkbutton(root, text="Второй флажок",
         variable=v2, onvalue=5, offvalue=0)
```

Списки

Listbox – это объект, в котором пользователь может выбрать один или несколько пунктов в зависимости от значения опции selectmode. Значение selectmode = SINGLE позволяет выбирать лишь один пункт из списка.

```
words = ['Linux','Python','Tk','Tkinter']

listbox = Listbox(root, selectmode=SINGLE, height=4)
# заполнение списка пунктами
for i in words:
 listbox.insert(END,i)
```

Получить выбранную в списке строку можно методом listbox.selection_get()

Менеджеры расположения

При изменении размеров главного окна все его дочерние виджеты должны подстроиться под новые размеры родителя.

Объект, который меняет расстановку виджетов, называется менеджером расположения.

В библиотеке tkinter их три: *grid, pack* и *place*.

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

entry = Entry(root, width = 25)
entry.place(x = 20, y = 20)

lbl = Label(root, text= '= 1000')
lbl.place(x = 200, y = 20)

btn = Button(root, text='Вычислить')
btn.place(x = 20, y = 50)

root.mainloop()
```

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

entry = Entry(root, width = 25)
entry.pack()

lbl = Label(root, text= '= 1000')
lbl.pack()

btn = Button(root, text='Вычислить')
btn.pack()

root.mainloop()
```

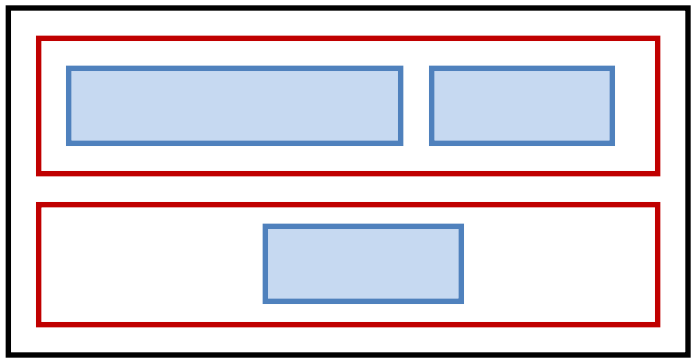
Менеджер **pack**

Аргументы:

side ("left" / "right" / "top" / "bottom") - к какой стороне должен примыкать размещаемый виджет. "top" – по умолчанию.

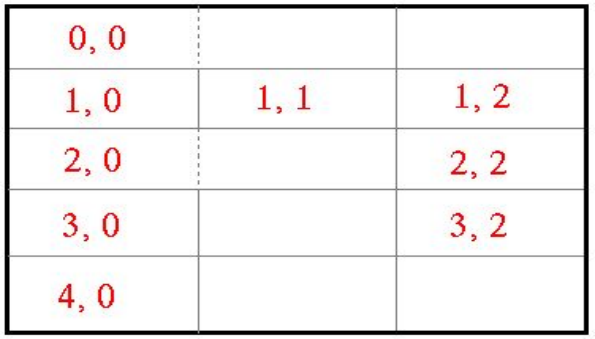
```
import tkinter
from tkinter import *
root = Tk()
entry = Entry(root, width = 25)
S = TOP
entry.pack(side=S)

lbl = Label(root, text='= 1000', bg='gray')
lbl.pack(side=S)

btn = Button(root, text='Вычислить')
btn.pack(side=S)
root.mainloop()
```

pack + Frame



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

f1=Frame(root,bg='gray', bd=5)
f2=Frame(root,bg='gray', bd=5)
f1.pack()
f2.pack()

entry = Entry(f1, width = 25)
entry.pack(side=LEFT)

lbl = Label(f1, text='= 1000',
bg='gray')
lbl.pack(side=LEFT)

btn = Button(f2, text='Вычислить')
btn.pack()

root.mainloop()
```

Менеджер **grid**



row - номер строки, в который помещаем виджет.

rowspan - сколько строк занимает виджет

column - номер столбца, в который помещаем виджет.

columnspan - сколько столбцов занимает виджет.

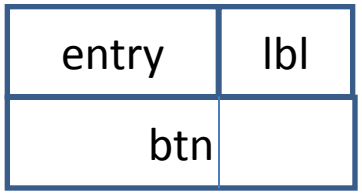
padx / **pady** - размер внешней границы (бордюра) по горизонтали и вертикали.

ipadx / **ipady** - размер внутренней границы (бордюра) по горизонтали и вертикали. Разница между pad и ipad в том, что при указании pad расширяется свободное пространство, а при ipad расширяется помещаемый виджет.

sticky ("n", "s", "e", "w" или их комбинация) - указывает к какой границе "приклеивать" виджет. Позволяет расширять виджет в указанном направлении.

Применение менеджера **grid**

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

entry = Entry(root, width = 25)
entry.grid(row = 0, column = 0, padx=5, pady=5)

lbl = Label(root, text='= 1000', bg='gray')
lbl.grid(row = 0, column = 1, padx=5, pady=5)

btn = Button(root, text='Вычислить')
btn.grid(row = 1, column = 0, columnspan = 2)

root.mainloop()
```



Менеджер **place**

Аргументы:

anchor ("n", "s", "e", "w", "ne", "nw", "se", "sw" или "center") – какой угол или сторона размещаемого виджета будет указана в аргументах x/y/relx/rely. По умолчанию "nw" - левый верхний угол.

bordermode ("inside", "outside", "ignore") – определяет, каким образом будут учитываться границы при размещении виджета.

x и **y** – абсолютные координаты виджета в пикселях.

width и **height** – абсолютные ширина и высота виджета.

relx и **rely** – относительные координаты (от 0.0 до 1.0) размещения виджета.

relwidth и **relheight** – относительные ширина и высота виджета.

place - резиновые координаты

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

entry = Entry(root, width = 25)
entry.place(relx = 0.1, rely = 0.1)

lbl = Label(root, text='= 1000',
bg='gray')
lbl.place(relx = 0.7, rely = 0.1)

btn = Button(root, text='Вычислить')
btn.place(relx = 0.4, rely = 0.5)

root.mainloop()
```

Самостоятельно

1. Расположить в окне три кнопки с надписями "Красный", "Зеленый", "Синий". При нажатии на кнопку окно должно окрашиваться в соответствующий цвет.

2. Сделать игру "крестики-нолики" на поле 3x3. Использовать 9 кнопок. Два игрока ходят по очереди, программа объявляет победителя или ничью.

3. Второй вариант того же – игра одного игрока против компьютера.

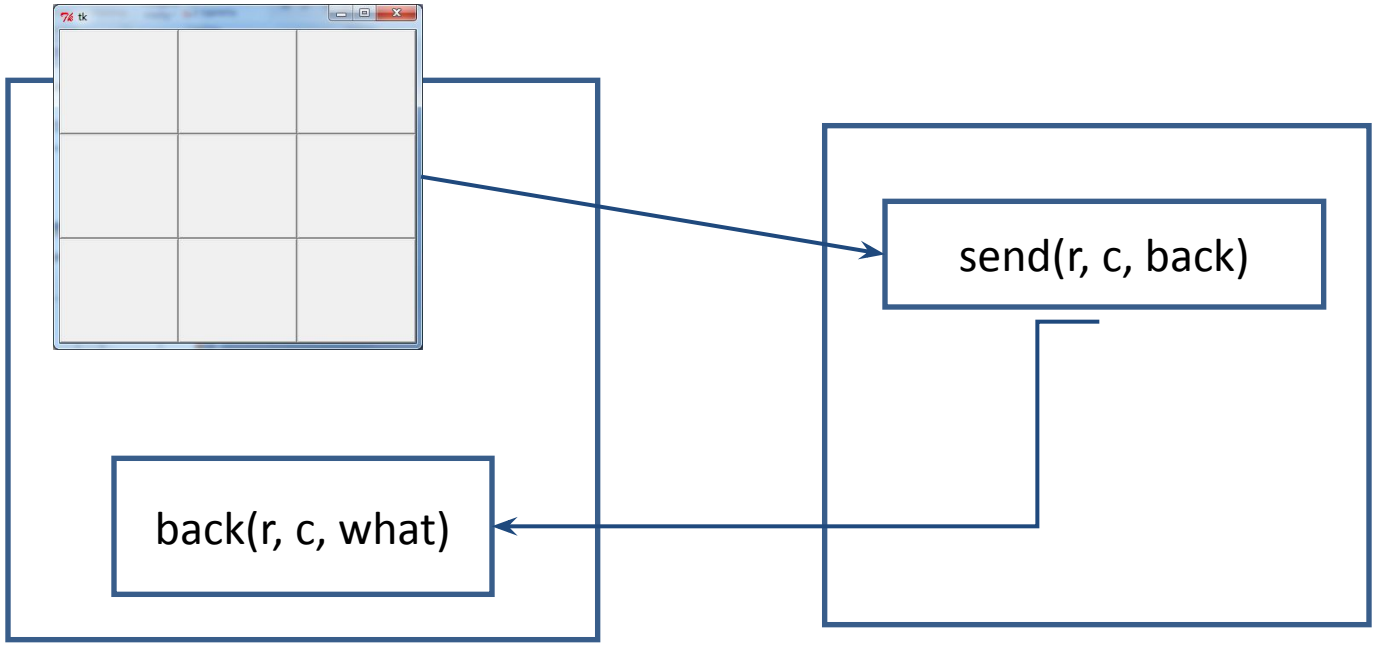
Три кнопки – вариант 1

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

def paintRed():   tk.config(bg="red")
def paintGreen(): tk.config(bg="green")
def paintBlue():  tk.config(bg="blue")

r = Button(tk, text="Red", command=paintRed)
r.pack(side=LEFT)

g = Button(tk, text="Green", command=paintGreen)
g.pack(side=LEFT)

b = Button(tk, text="Blue", command=paintBlue)
b.pack(side=LEFT)

tk.mainloop()
```

Три кнопки – вариант 2

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

def paint(color): tk.config(bg=color)

buttons = [
  Button(tk, text="Red",   command=lambda: paint("red")),
  Button(tk, text="Green", command=lambda: paint("green")),
  Button(tk, text="Blue",  command=lambda: paint("blue")),]
for b in buttons:
    b.pack(side=LEFT)

tk.mainloop()
```

Крестики-нолики



Главный модуль – GUI

Модель

Крестики-нолики

```
from model import *
from tkinter import *
tk = Tk()
# квадратный массив кнопок с индивидуальными командами
b = [[ Button(tk, command=lambda: send(0, 0, back)),
       Button(tk, command=lambda: send(0, 1, back)),
       Button(tk, command=lambda: send(0, 2, back))],
     [ Button(tk, command=lambda: send(1, 0, back)),
       Button(tk, command=lambda: send(1, 1, back)),
       Button(tk, command=lambda: send(1, 2, back))],
     [ Button(tk, command=lambda: send(2, 0, back)),
       Button(tk, command=lambda: send(2, 1, back)),
       Button(tk, command=lambda: send(2, 2, back))],]

# настройка внешнего вида кнопок
for r in range(3):
    for c in range(3):
        b[r][c].config(font="Consolas 44", width=4, text=" ")
        b[r][c].grid(row=r, column=c)

# функция обратного вызова - ставит значки на кнопках,
сообщает о победе
def back(r, c, what, winner):
    global b
    if winner != 2:
        tk.title("Победил " + ("нолик", "крестик")[winner])
        disableButtons()
    b[r][c]["text"] = "0X"[what]

def disableButtons():
    for r in range(3):
        for c in range(3):
            b[r][c]["command"] = ''

tk.mainloop()
```

```
# Модель в виде числовой таблицы с числами: 0-нолик, 1-
крестик, 2-пусто
S = 2
X = 1
field = [ [S,S,S], [S,S,S], [S,S,S], ]

# Что ставить на очередном ходе - крестик или нолик
# Начинаем с крестика
current = X

# Команда, которую модуль main подает модели
def send(r, c, back):
    global current, field
    if field[r][c] == S:
        field[r][c] = current
        current = (current + 1) % 2
        back(r, c, field[r][c], getWinner(field))

def getWinner(m):
    # по гризонтали
    if m[0][0] == m[0][1] == m[0][2] != S: return m[0][0]
    if m[1][0] == m[1][1] == m[1][2] != S: return m[1][0]
    if m[2][0] == m[2][1] == m[2][2] != S: return m[2][0]
    # по вертикали
    if m[0][0] == m[1][0] == m[2][0] != S: return m[0][0]
    if m[0][1] == m[1][1] == m[2][1] != S: return m[0][1]
    if m[0][2] == m[1][2] == m[2][2] != S: return m[0][2]
    # по диагонали
    if m[0][0] == m[1][1] == m[2][2] != S: return m[0][0]
    if m[2][0] == m[1][1] == m[0][2] != S: return m[2][0]
    return 2
```

События

Что такое событие

Типовой сценарий события

1. Пользователь совершил действие (кликнул мышкой по кнопке).
2. Операционная система получила информацию об этом.
3. ОС вычислила программу и кнопку, по которой кликнул пользователь.
4. ОС отправила сообщение программе с информацией о действии пользователя (т.е. вызвала какую-то закрытую функцию программы).
5. Внутри этой закрытой функции находится проверка, есть ли функция, которую программист связал с этим событием.
6. Если функция есть, она вызывается и ей передается параметр – информация о событии.

Метод bind()

Метод bind() привязывает функцию к какому-либо действию пользователя (нажатие кнопки мыши, нажатие клавиши на клавиатуре и т.д.).

Метод bind() принимает три аргумента:

1) название события
2) функцию, которая будет вызвана при наступлении события
3) третий аргумент необязательный – строка "+" - означает, что эта привязка добавляется к уже существующим.

```
from tkinter import *
root = Tk()
frm = Frame(root, width=400, height=400, bg="pink")
frm.pack()

def my(e):
    b = Button(frm, text="XXX")
    b.place(x = e.x, y = e.y)

frm.bind("<Button-1>", my)
root.mainloop()
```

Виды событий

MouseWheel - прокрутка колесом мыши

KeyPress, KeyRelease - нажатие и отпускание клавиши на клавиатуре

ButtonPress, ButtonRelease, Motion - нажатие, отпускание клавиши мыши, движение мышью

Configure - изменение положения или размера окна

Map, Unmap - показывание или сокрытие окна (например, в случае сворачивания/разворачивания окна пользователем)

Expose - событие генерируется, когда необходимо всё окно или его часть перерисовать

FocusIn, FocusOut - получение или лишение фокуса ввода виджетом

Enter, Leave - курсор мыши "входит" в виджет, или "уходит" из виджета

Пример: визуализация координат курсора

```
def show(ev):
    """выводит координаты мыши в заголовок окна"""
    str = "x = {0} y = {1}".format(ev.x, ev.y)
    root = ev.widget.master
    root.title(str)

from tkinter import *
root = Tk()

f1 = Frame(root, bg="yellow", width = 400, height = 400)
f1.pack()
f1.bind("<Motion>", show)

root.mainloop()
```

Рисование на холсте

Сначала нужно создать холст (род виджета, как кнопка или окно)

```
canvas = Canvas(root, width = 500, height = 500, bg =
"lightblue", cursor = "pencil")
```

На холсте можно рисовать линии

```
canvas.create_line(200,50,300,50,width=3,fill="blue")
canvas.create_line(0,0,100,100,width=2,arrow=LAST)
```

прямоугольники

```
x = 75
y = 110
canvas.create_rectangle(x,y,x+80,y+50,fill="white",
     outline="blue")
```

многоугольники

```
canvas.create_polygon([250,100],[200,150],[300,150],
     fill="yellow")
canvas.create_polygon([300,80],[400,80],[450,75],[450,200],
     [300,180],[330,160],outline="white",smooth=1)
```

эллипсы

```
canvas.create_oval([20,200],[150,300],fill="gray50")
```

Самостоятельно

1. Решить задачу с тремя кнопками, используя обработку событий.

2. Переписать игру в крестики нолики, используя обработку событий.

Три кнопки – вариант 3

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

def click(e):
    tk.config(bg = e.widget.color)

for color in ["Red", "Green", "Blue"]:
    b = Button(tk, text = color)
    b.bind("<Button-1>", click)
    b.color = color
    b.pack(side = LEFT)

tk.mainloop()
```