KORISNIČKA SUČELJA

Aleksandar Maksimović
IRB

Tkinter

Tkinter - standardno sučelje Pythona za Tk GUI toolkit

- Unix, Windows i Macintosh sustavi od verzije 8.0
- veliki broj modula
- Tk sučelje (dll, so ili statički library) _tkinter binarni file

Importiranje

Import Tkinter

ili

from Tkinter import *

Tkinter root

Na "*NIX" operacijskim sustavima komponente grafičkog korisničkog sučelja nazivaju se widget. widget je kovanica rijeci window i gadget.

Tkinter mora imati Tk root widget. To je običan prozor s naslovom (title bar) i ukrasima od OS.

- root widget
  - samo jedan za aplikaciju
  - mora biti kreiran prije svih ostalih widgeta

Hello world GUI

Prvo napravimo root widget

- root=Tk()

Napravimo mjesto za poruku Label koja je "dijete" root prozoru

- w=Label(root,text="Hello, world!")
  - Label - tekst, ikona ili slika

Naredbom pack postižemo da je veličina Label widgeta jednaka tekstu

Na kraju petlja koja ovisi o događaju

- root.mainloop()

skripta

```
# File: hello1.py

from Tkinter import *

root = Tk()

w = Label(root, text="Hello, world!")
w.pack()

root.mainloop()
```

rezultat

GUI gumbi

```
# File: hello2.py

from Tkinter import *

class App:

    def __init__(self, master):

        frame = Frame(master)
        frame.pack()

        self.button = Button(frame, text="QUIT", fg="red", command=frame.quit)
        self.button.pack(side=LEFT)

        self.hi_there = Button(frame, text="Hello", command=self.say_hi)
        self.hi_there.pack(side=LEFT)

    def say_hi(self):
        print "hi there, everyone!"

root = Tk()

app = App(root)

root.mainloop()
```

konstruktor

funkcija

pozivanje klase

gumbi

Konstruktor __init__ dodaje widget frame i dva gumba glavnom root prozoru.

frame služi kao spremnik (container), u ovom slučaju sadrži dva gumba.

lokalnoj varijabli frame je pridružena klasa Frame, pack naredbom dobivamo vidljivi frame.

Pridružujemo "djecu" widgetu frame. Lokalne varijable button i hi_there prikazuju gumbe kojima su pridružene funkcije printanja i izlazak iz aplikacije

```
self.button = Button(frame, text="QUIT", fg="red", command=frame.quit)
self.button.pack(side=LEFT)

self.hi_there = Button(frame, text="Hello", command=self.say_hi)
self.hi_there.pack(side=LEFT)
```

klase Tkinter

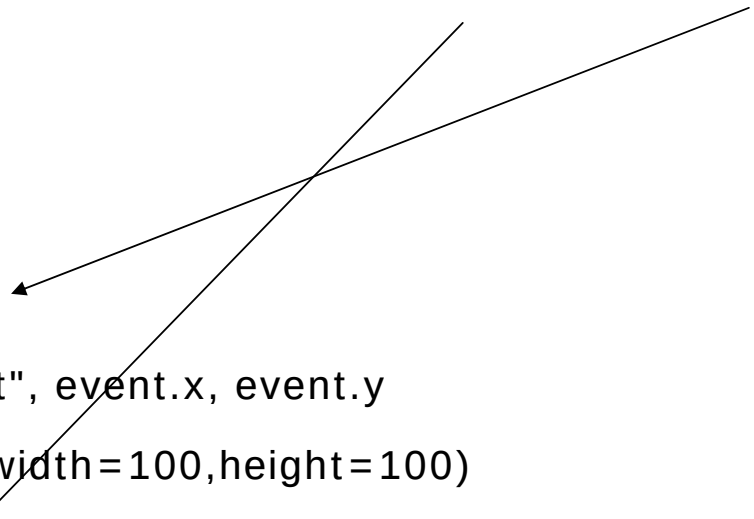
Widget	Description
Button	A simple button, used to execute a command or other operation.
Canvas	Structured graphics. This widget can be used to draw graphs and plots, create graphics editors, and to implement custom widgets.
Checkbutton	Represents a variable that can have two distinct values. Clicking the button toggles between the values.
Entry	A text entry field.
Frame	A container widget. The frame can have a border and a background, and is used to group other widgets when creating an application or dialog layout.
Label	Displays a text or an image.
Listbox	Displays a list of alternatives. The listbox can be configured to get radiobutton or checklist behavior.
Menu	A menu pane. Used to implement pulldown and popup menus.
Menubutton	A menubutton. Used to implement pulldown menus.
Message	Display a text. Similar to the label widget, but can automatically wrap text to a given width or aspect ratio.

klase ...

Radiobutton	Represents one value of a variable that can have one of many values. Clicking the button sets the variable to that value, and clears all other radiobuttons associated with the same variable.
Scale	Allows you to set a numerical value by dragging a "slider".
Scrollbar	Standard scrollbars for use with canvas, entry, listbox, and text widgets.
Text	Formatted text display. Allows you to display and edit text with various styles and attributes. Also supports embedded images and windows.
Toplevel	A container widget displayed as a separate, top-level window.

bind

widget.bind(event, handler) - veza događaja i funkcije



```
# File: bind1.py
from Tkinter import *
root = Tk()
def callback(event):
        print "clicked at", event.x, event.y
frame = Frame(root, width=100,height=100)
frame.bind("<Button-1>", callback)
frame.pack()
root.mainloop()
```

događaji

<modifier-type-detail> -sintaksa događaja koje opisujemo stringom

Event	Description
<Button-1>	A mouse button is pressed over the widget. Button 1 is the leftmost button, button 2 is the middle button (where available), and button 3 the rightmost button. When you press down a mouse button over a widget, Tkinter will automatically "grab" the mouse pointer, and mouse events will then be sent to the current widget as long as the mouse button is held down. The current position of the mouse pointer (relative to the widget) is provided in the x and y members of the event object passed to the callback. You can use ButtonPress instead of Button, or even leave it out completely: <Button-1>, <ButtonPress-1>, and <1> are all synonyms. For clarity, I prefer the <Button-1> syntax.
<B1-Motion>	The mouse is moved, with mouse button 1 being held down (use B2 for the middle button, B3 for the right button). The current position of the mouse pointer is provided in the x and y members of the event object passed to the callback.
<Button-Release-1>	Button 1 was released. The current position of the mouse pointer is provided in the x and y members of the event object passed to the callback.

događaji

<Double-Button-1>	Button 1 was double clicked. You can use Double or Triple as prefixes. Note that if you bind to both a single click (<Button-1>) and a double click, both bindings will be called.
<Enter>	The mouse pointer entered the widget (this event doesn't mean that the user pressed the **Enter** key!).
<Leave>	The mouse pointer left the widget.
<Return>	The user pressed the Enter key. You can bind to virtually all keys on the keyboard. For an ordinary 102-key PC-style keyboard, the special keys are **Cancel** (the Break key), **BackSpace**, **Tab**, **Return**(the Enter key), **Shift_L** (any Shift key), **Control_L** (any Control key), **Alt_L** (any Alt key), **Pause**, **Caps_Lock**, **Escape**, **Prior** (Page Up), **Next** (Page Down), **End**, **Home**, **Left**, **Up**, **Right**, **Down**, **Print**, **Insert**, **Delete**, **F1**, **F2**, **F3**, **F4**, **F5**, **F6**, **F7**, **F8**, **F9**, **F10**, **F11**, **F12**, **Num_Lock**, and **Scroll_Lock**.
<Key>	The user pressed any key. The key is provided in the char member of the event object passed to the callback (this is an empty string for special keys).

objekt

događaj je objekt u Pythonu, prenosimo ga u callback i koristimo atribute

Attribute	Description
widget	The widget which generated this event. This is a valid Tkinter widget instance, not a name. This attribute is set for all events.
x, y	The current mouse position, in pixels.
x_root, y_root	The current mouse position relative to the upper left corner of the screen, in pixels.
char	The character code (keyboard events only), as a string.
keysym	The key symbol (keyboard events only).
keycode	The key code (keyboard events only)
num	The button number (mouse button events only)
width, height	The new size of the widget, in pixels (Configure events only).
type	The event type.

izbornik

```
# File: menu1.py
from Tkinter import *
def callback():
        print "called the callback!"
root = Tk()
# create a menu
menu = Menu(root)
root.config(menu=menu)
filemenu = Menu(menu)
menu.add_cascade(label="File", menu=filemenu)
filemenu.add_command(label="New", command=callback)
filemenu.add_command(label="Open...", command=callback)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=root.quit)
helpmenu = Menu(menu)
menu.add_cascade(label="Help", menu=helpmenu)
helpmenu.add_command(label="About...",command=callback)
mainloop()
```

Ne moramo koristiti pack() kod izbornika.

alatna traka

```
from Tkinter import *
root = Tk()
def callback():
        print "called the callback!"
# create a toolbar
toolbar = Frame(root)
b = Button(toolbar, text="new", width=6, command=callback)
b.pack(side=LEFT, padx=2, pady=2)
b = Button(toolbar, text="open", width=6, command=callback)
b.pack(side=LEFT, padx=2, pady=2)
toolbar.pack(side=TOP, fill=X)
mainloop()
```

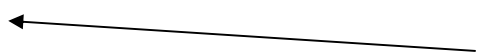
Ona je u stvari frame, tj. nema alatne trake kao poseban widget.

statusna traka

```
# File: tkSimpleStatusBar.py
class StatusBar(Frame):
def __init__(self, master):
        Frame.__init__(self, master)
        self.label = Label(self, bd=1, relief=SUNKEN, anchor=W)
        self.label.pack(fill=X)
def set(self, format, *args):
        self.label.config(text=format % args)
        self.label.update_idletasks()
def clear(self):
        self.label.config(text="")
        self.label.update_idletasks()
root=Tk(); status = StatusBar(root)
status.pack(side=BOTTOM, fill=X)
```

status bar

trenutno stanje



brisanje stanje

dijalog

```
# File: dialog1.py
from Tkinter import *
class MyDialog:
        def __init__(self, parent):
                top = self.top = Toplevel(parent)
                Label(top, text="Value").pack()
                self.e = Entry(top)
                self.e.pack(padx=5)
                b = Button(top, text="OK", command=self.ok)
                b.pack(pady=5)
        def ok(self):
                print "value is", self.e.get()
                self.top.destroy()
```

```
root = Tk()
Button(root, text="Hello!").pack()
root.update()
d = MyDialog(root)
root.wait_window(d.top)
```

Ne koristimo mainloop,

- root prozor aktivan
- <Enter> ne radi moramo kliknuti OK
- nemamo Cancel za dijalog

klasa dijalog

```
from Tkinter import *
import os
class Dialog(Toplevel):
def __init__(self, parent, title = None):
        Toplevel.__init__(self, parent)
        self.transient(parent)
        if title:
                self.title(title)
        self.parent = parent
        self.result = None
        body = Frame(self)
        self.initial_focus = self.body(body)
        body.pack(padx=5, pady=5)
        self.buttonbox()
        self.grab_set()
        if not self.initial_focus:
        self.initial_focus = self
        self.protocol("WM_DELETE_WINDOW", self.cancel)
        self.geometry("+%d+%d" % (parent.winfo_rootx()+50,
        parent.winfo_rooty()+50))
        self.initial_focus.focus_set()
        self.wait_window(self)
```

klasa

```
# construction hooks
def body(self, master):
        # create dialog body. return widget that should have
        # initial focus. this method should be overridden
        pass
def buttonbox(self):
        # add standard button box. override if you don't want   # standard buttons
        box = Frame(self)
        w = Button(box, text="OK", width=10, command=self.ok, default=ACTIVE)
        w.pack(side=LEFT, padx=5, pady=5)
        w = Button(box, text="Cancel", width=10, command=self.cancel)
        w.pack(side=LEFT, padx=5, pady=5)
        self.bind("&lt;Return>", self.ok)
        self.bind("&lt;Escape>", self.cancel)
        box.pack()
```

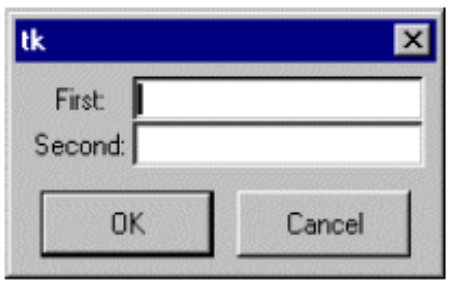
primjena klase

```
# File: dialog2.py
import tkSimpleDialog
class MyDialog(tkSimpleDialog.Dialog):
        def body(self, master):
                Label(master, text="First:").grid(row=0)
                Label(master, text="Second:").grid(row=1)
                self.e1 = Entry(master)
                self.e2 = Entry(master)
                self.e1.grid(row=0, column=1)
                self.e2.grid(row=1, column=1)
                return self.e1 # initial focus
```

```
  def apply(self):
        first = string.atoi(self.e1.get())
        second = string.atoi(self.e2.get())
        print first, second # or something

from Tkinter import *
root = Tk()
Button(root, text="Hello!").pack()
root.update()
d = MyDialog(root)
root.wait_window(d.top)
```

primjena klase

body funkcija - vraća fokus na prvi tekst *First*:, može vratiti trenutno aktivni widget, ako to nije važno može se napraviti return None.



Procesiranje (obrada) nalazi se u funkciji *apply* , umjesto obrade ili korištenja podataka podatke možemo spremiti u varijablu (*result)* koju možemo kasnije koristiti

```
def apply(self):
   first = int(self.e1.get())
   second = int(self.e2.get())
   self.result = first, second

d = MyDialog(root)
print d.result
```

Ako napravimo Cancel, metoda *apply* nikada se ne izvrši i *result* nije inicijaliziran. Konstruktor postavlja ovu varijablu kao *None*, zbog toga se može testirati prije procesiranja. Ostale varijable inicijaliziramo u funkciji *body* .

Grid

pack menađer - Frame s oznakom (label) "First" + ulazno polje (entry field) + pack(side=LEFT)

- Frame s oznakom (label) "Second" + ulazno polje (entry field) + pack(side=LEFT)

- spakiraju se oba Frame widgeta u novi Frame + side=TOP

može doći do različitih problema, ako su ulazna polja različite dužine ne dolazi do željenog poravnavanja

Grid menađer - jednostavnije

First:	<entry field>
Second:	<entry field>
<checkbutton>	

.grid(row=N, column=M)

- broji se od 0

- widgeti su centrirani u ćeliji

opcija sticky služi za rastezanje ili poravnavanje ćelija

validate

```
def apply(self):
   try:
      first = int(self.e1.get())
      second = int(self.e2.get())
      dosomething((first, second))
   except ValueError:
      tkMessageBox.showwarning(
         "Bad input",
         "Illegal values, please try again"
      )
```

ok funkcija uništila je dijalog kada poziva metodu *apply*. Dizajn namjerno napravljen, jer bi dugačak račun zbunio korisnika prisutnim dijalogom. Metoda *validate* poziva se prije nego se dijalog uništi.

U primjeru dio koda iz funkcije *apply* preseljen je u funkciju validate, gdje se rezultat spremi za korištenje u *apply* metodi.

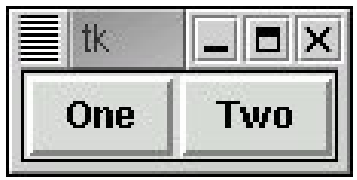
```
def validate(self):
   try:
      first= int(self.e1.get())
      second = int(self.e2.get())
      self.result = first, second
      return 1
   except ValueError:
      tkMessageBox.showwarning(
         "Bad input",
         "Illegal values, please try again"
      )
      return 0

def apply(self):
   dosomething(self.result)
```

GUI Tkinter- gumbi

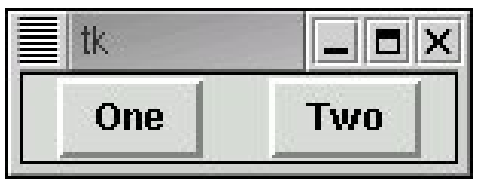
```
>>> from Tkinter import *
>>> win=Tk()
>>> b1 = Button(win,text="One")
>>> b2 = Button(win,text="Two")
```

```
>>> b1.pack()
>>> b2.pack()
```


```
>>> b2.pack(side=LEFT)
>>> b1.pack(side=LEFT)
```




```
>>> b1.pack(side=LEFT,padx=10)
>>> b2.pack(side=LEFT,padx=10)
```



GUI Tkinter- grid

```
>>> win = Tk()
>>> b1 = Button(win,text="One")
>>> b2 = Button(win,text="Two")
>>> b1.grid(row=0, column=0)
>>> b2.grid(row=1, column=1)
```


```
>>> l = Label(win, text="This is a label")
>>> l.grid(row=1,column=0)
```

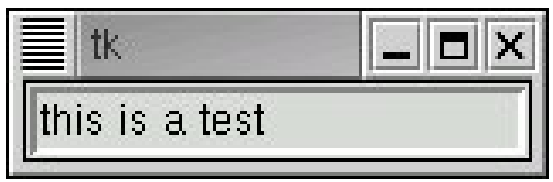
GUI Tkinter- grid


```
>>> win = Tk()
>>> f = Frame(win)
>>> b1 = Button(f, "One")
>>> b2 = Button(f, "Two")
>>> b3 = Button(f, "Three")
>>> b1.pack(side=LEFT)
>>> b2.pack(side=LEFT)
>>> b3.pack(side=LEFT)
>>> l = Label(win,"This label is over all buttons")
>>> l.pack()
>>> f.pack()
```

```
>>> b1.configure(text="Uno")
 >>> def but1() : print "Button one was pushed"
...
>>> b1.configure(command=but1)
```

GUI Tkinter- Entry widgets

```
>>> win = Tk()
>>> v = StringVar()
>>> e = Entry(win,textvariable=v)
>>> e.pack()
```



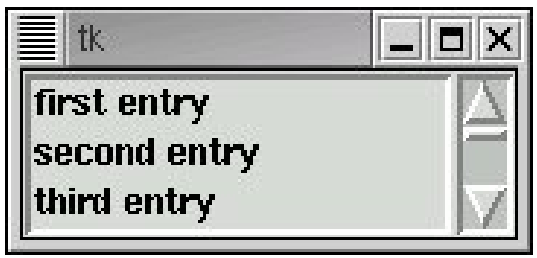

```
>>> v.get()
 "this is a test"
```

```
>>> v.set("this is set from the program")
```

GUI Tkinter-Listbox widget

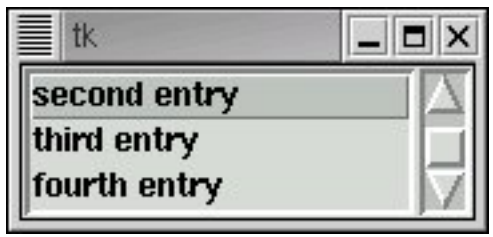
```
>>> win = Tk()
>>> lb = Listbox(win, height=3)
>>> lb.pack()
>>> lb.insert(END,"first entry")
>>> lb.insert(END,"second entry")
>>> lb.insert(END,"third entry")
>>> lb.insert(END,"fourth entry")
```

```
>>> sb = Scrollbar(win,orient=VERTICAL)
>>> sb.pack(side=LEFT,fill=Y)
```



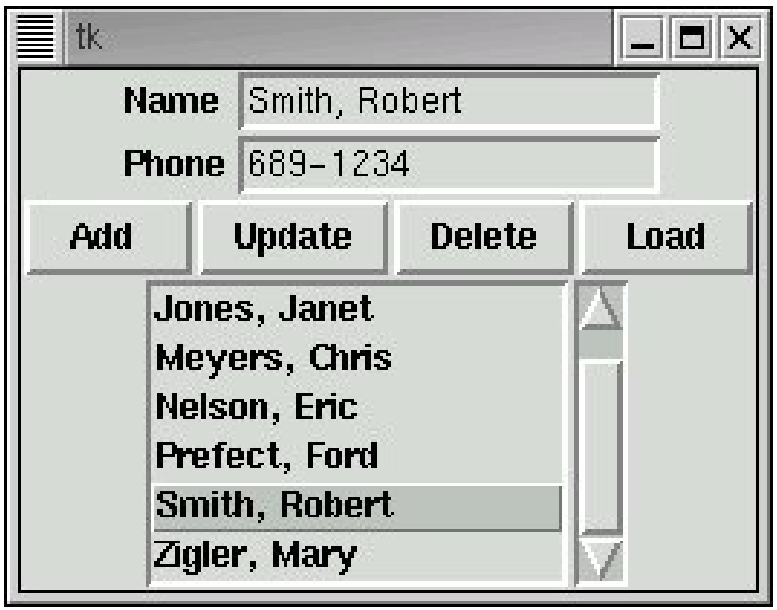
4 fali

```
>>> sb.configure(command=lb.yview)
>>> lb.configure(yscrollcommand=sb.set)
```




```
>>> lb.curselection()
('2',)
```

GUI Tkinter



Početna lista: phones.py

```
phonelist = [ ['Meyers, Chris', '343-4349'],
 ['Smith, Robert', '689-1234'],
 ['Jones, Janet', '483-5432'],
 ['Barnhart, Ralph','683-2341'],
['Nelson, Eric', '485-2689'],
['Prefect, Ford', '987-6543'],
['Zigler, Mary', '567-8901'],
['Smith, Bob', '689-1234'] ]
```

zadatak

Napravi GUI, koristi funkcije setSelect i select.curselection()

```
from Tkinter import *
from phones import *
def makeWindow () :
        global nameVar, phoneVar, select
        win = Tk()
        frame1 = Frame(win);  frame1.pack()
        Label(frame1, text="Name").grid(row=0, column=0, sticky=W)
        nameVar = StringVar()
        name = Entry(frame1, textvariable=nameVar)
        .....................
        frame2 = Frame(win) # Row of buttons
        ............................
        frame3 = Frame(win) # select of names
        ..............
        return win
```

```
def whichSelected () :
def addEntry () :
def updateEntry() :
def deleteEntry() :
def loadEntry () :
```

```
def setSelect () :
        phonelist.sort()
        select.delete(0,END)
        for name,phone in phonelist :
                select.insert (END, name)
win = makeWindow()
setSelect ()
win.mainloop()
```

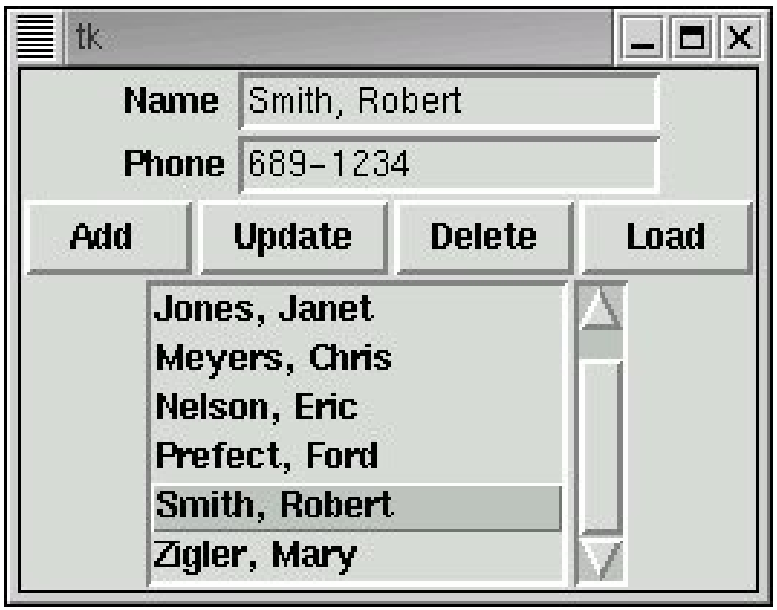
ponovo

- Tkinter
  - win = Tk() - root window
  - Button(win,text="One") - gumbi
  - Label(win, text="This is a label") - oznake
  - pack(side=LEFT) - pakiranje, grid menađer
  - b1.configure(command=but1) - naknadno definiranje komandi i osobina
  - Entry(win,textvariable=v), gdje je v=StringVar()

ponovo

- lb=Listbox(win, height=3) - LISTBOX
  - lb.insert(END,"first entry") - insertiranje u list box
  - lb.curselection()
- sb=Scrollbar(win,orient=VERTICAL)
  - sb.configure(command=lb.yview)
  - lb.configure(yscrollcommand=sb.set)

GUI Tkinter



Početna lista: phones.py

```
phonelist = [ ['Meyers, Chris', '343-4349'],
 ['Smith, Robert', '689-1234'],
 ['Jones, Janet', '483-5432'],
 ['Barnhart, Ralph','683-2341'],
['Nelson, Eric', '485-2689'],
['Prefect, Ford', '987-6543'],
['Zigler, Mary', '567-8901'],
['Smith, Bob', '689-1234'] ]
```

zadatak

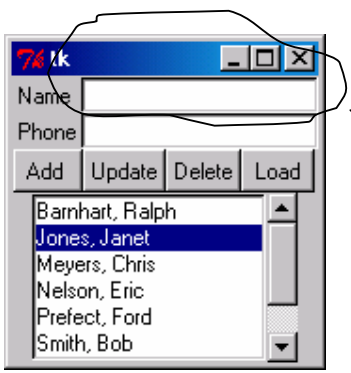
Napravi GUI, koristi funkcije setSelect i select.curselection()

```
from Tkinter import *
from phones import *
def makeWindow () :
        global nameVar, phoneVar, select
        win = Tk()
        frame1 = Frame(win);            frame1.pack()
        Label(frame1, text="Name").grid(row=0, column=0, sticky=W)
        nameVar = StringVar()
        name = Entry(frame1, textvariable=nameVar)
        .....................
        frame2 = Frame(win) # Row of buttons
        ............................
        frame3 = Frame(win) # select of names
        ..............
        return win
```

```
def whichSelected () :
def addEntry () :
def updateEntry() :
def deleteEntry() :
def  loadEntry () :
```

tel. imenik

from phones import * -- phonelist globalna varijabla



gumb Load

def loadEntry () :

def setSelect () :

kompletna funkcija

sortira phonelist listu

insertira u listbox **select**

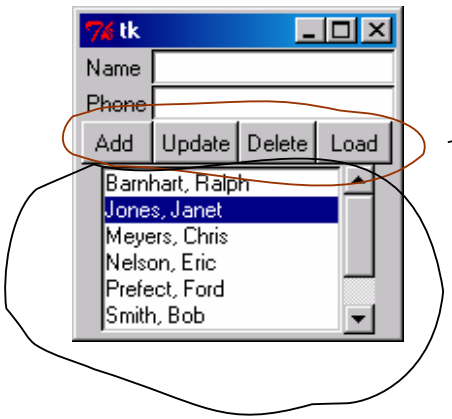
Add ⟶ def addEntry () :

Update ⟶ def updateEntry() :

Delete ⟶ def deleteEntry() :

frame1 = Frame(win); frame1.pack()
Label(frame1, text="Name").grid(row=0, column=0, sticky=W)
nameVar = StringVar()
name = Entry(frame1, textvariable=nameVar)
name.grid(row=0, column=1, sticky=W)

imenik



frame2 - gumbi

frame3 - listbox+scrollbar

```
FRAME2
b1 = Button(frame2,text=" Add  ",command=addEntry)
.....
b1.pack(side=LEFT);
```

```
FRAME3
scroll = Scrollbar(frame3, orient=VERTICAL)
 select = Listbox(frame3,.....
```

```
def whichSelected () :
print "At %s of %d" % (select.curselection(), len(phonelist))
  return int(select.curselection()[0])
```

Valovi

```
class wave :
   def   init    (self, points=400, formula=None) :
      self.data = [0.0]*points
      self.points= points
      if formula :
         for p in range(points) :
            x = p*pi*2/points
            self.data[p] = eval(formula)
```

data = podaci dobiveni formulom

Kad se kreira objekt wave, broj točaka u argumentu se rasporede u intervalu od 0 do 2 pi. Formulu definiramo stringom, npr. "sin(x)". Sinusni val s 400 točaka

```
>>> import wave
>>> w = wave.wave(formula="sin(x)",points=400)
>>> print
w.data[0],w.data[100],w.data[200],w.data[300]
0.0 1.0 1.22460635382e-16 -1.0
```

valovi: zbrajanje, množenje

```
def __add__ (self, other) :
    target = wave(points=self.points)
    for i in range(self.points) :
        target.data[i] = self.data[i] + other.data[i]
    return target
```

target je novi objekt "val"

oduzimanje dobivamo zamjenom + u -.

```
def __mul__ (self, other) :
    target = wave(points=self.points)
    if type(other) == type(5) or type(other) ==
type(5.0) :
        for i in range(self.points) :
            target.data[i] = self.data[i] * other
    else :
        for i in range(self.points) :
            target.data[i] = self.data[i] *
other.data[i]
```

Definiramo množenje vala brojem (skalarom), odnosno drugim valom gdje se sve točke valova množe.

GUI, Tk plot

```
>>> from Tkinter import *
>>> win = Tk()
```

root window

```
>>> canvas = Canvas(win,height=400,width=400)
>>> canvas.pack()
```

Canvas - mjesto gdje crtamo točke.

```
>>> canvas.create_line(0,0,200,200)
```

linija od gornjeg lijevog kuta do doljnjeg desnog kuta.

```
>>> win.mainloop()
```

napravi prozor i daljnje izvršavanje programa je blokirano dok se prozor ne zatvori.

plot

```
def plot (self, title="??", pixHeight=None,
maxY=None, others=[]) :
   if not pixHeight : pixHeight = self.points*2/3   #
Pleasant ratio
   pixWidth = self.points
   # find max and min data to scale
   if not maxY :
      maxY = max (max(self.data), -min(self.data))
      offset = pixHeight/2
      scale = offset/maxY
```

odredimo skaliranje za "y" os

```
...
   win = Tk()
   win.title (title)
   canvas =
Canvas(win,width=pixWidth,height=pixHeight)
   # create zero line
   canvas.create_line(0,offset,pixWidth,offset)
   canvas.pack()
```

"window" napravimo

nacrtamo 0 os na Canvasu

plot

```
...
   self.plotOne (canvas, pixWidth, scale, offset)
   for i in range(len(others)) :
      others[i].plotOne (canvas, pixWidth, scale,
offset)
      if sys.platform == "win32" : win.mainloop()
```

pozovemo funkciju plotOne za crtanje "vala".

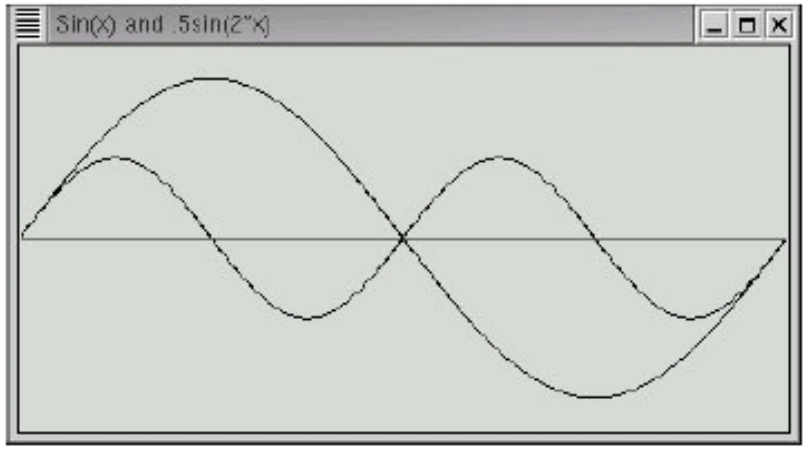
```
def plotOne (self, canvas, pixWidth, scale, offset) :
   for x in range(pixWidth) :
      y = offset - self.data[x] * scale
      if x : canvas.create_line(x-1,yprev,x,y)
      yprev = y
```

plotOne - crta liniju za svaku točku u valu.

plot primjeri

Definiramo dva vala različitih frekvencija.

```
>>> import wave
>>> a = wave.wave(formula="sin(x)")        #
fundamental
>>> b = wave.wave(formula=".5*sin(2*x)")   # harmonic
```



Nacrtamo ih zajedno.

```
>>> a.plot(maxY=1.2, pixHeight=200, title="Sin(x) and
.5sin(2*x)", others=[b])
```

Direktorij vjezbe6: waverun1, waverun2, waverun3.py. wave.py - library

PythonPlot

Metadata-Version: 1.0

Name: PythonPlot

Version: 0.2.9

Summary: UNKNOWN

Home-page: UNKNOWN

Author: Bernd Aberle

Author-email: bernd.aberle@arcor.de

License: UNKNOWN

Description: UNKNOWN

Platform: UNKNOWN

Direktorij: vjezbe6\PythonPlot-0.3.1

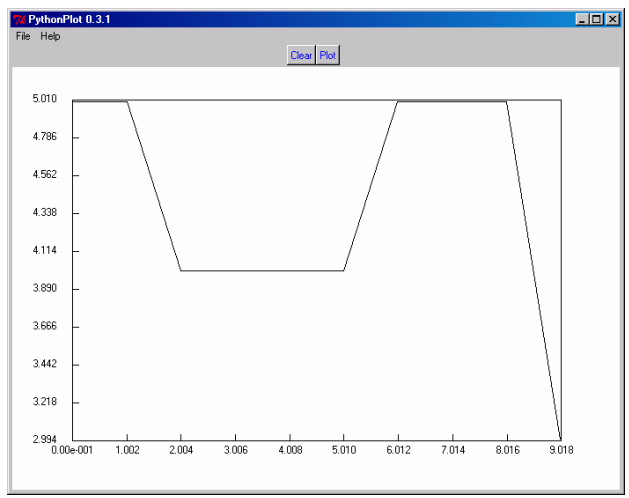
Prepun pogrešaka, ali je relativno jednostavan i napisan u Tk.

Ispis u ps formatu, može čitati i podatke iz file-a.

Primjer

- 2 primjera
  - PythonPlotExample1 - čita podatke iz file-a i crta ih
  - PythonPlotExample2 - crta podatke iz programa

```
from PythonPlot import *

master = Tk()

xaxis = [0, 1, 2, 5, 6, 8, 9]

yaxis = [5, 5, 4, 4, 5, 5, 3]

plot = PythonPlot(master)

plot.plotData(xaxis, yaxis)

master.mainloop()
```



MSDie

- MSDie - hiperkocka ima stranica koliko želimo
  - Moramo znati:
    - koliko stranica imamo
    - koja je trenutna vrijednost

```
>>> die1 = MSDie(6)

>>> die1.getValue()

1
>>> die1.roll()
>>> die1.getValue()

4
>>> die2 = MSDie(13)

>>> die2.getValue()

1
>>> die2.roll()
>>> die2.getValue()
```

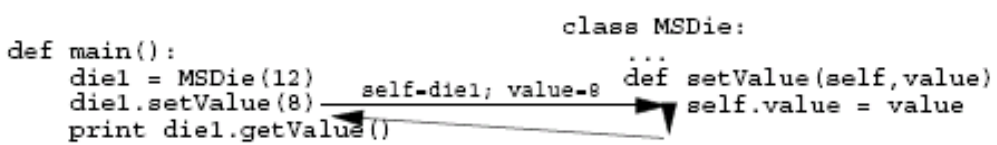
Klasa

```
# msdie.py
# Class definition for an n-sided die.
from random import randrange
class MSDie:
        def __init__(self, sides):
        self.sides = sides
        self.value = 1
```

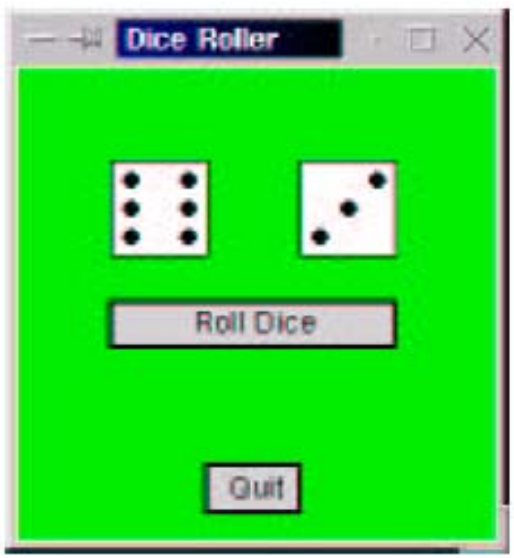
Klasa

```
def roll(self):
        self.value = randrange(1,self.sides+1)
def getValue(self):
        return self.value
def setValue(self, value):
        self.value = value
```

metode



GUI hiperkocka



Metode:

- konstruktor stvara gumb u prozoru
- activate - stanje gumba aktivno
- deactivate - pasivno stanje gumba
- clicked - ako je gumb aktivan, odredi da li je klik bio u kvadratu gumba.
- getLabel- vrati oznaku gumba

metode

```
def activate(self):
    "Sets this button to 'active'."
    self.label.setFill('black')
    self.rect.setWidth(2)
    self.active = 1
```

aktivan, oznaka crna i rub debljine 2

```
def deactivate(self):
    "Sets this button to 'inactive'."
    self.label.setFill('darkgrey')
    self.rect.setWidth(1)
    self.active = 0
```

nije aktivan, oznaka siva i rub debljine 1

```
def clicked(self, p):
    "RETURNS true if button is active and p is inside"
    return self.active and \
           self.xmin <= p.getX() <= self.xmax and \
           self.ymin <= p.getY() <= self.ymax
```

klik - ako je aktivan i u kvadratu vrati true

sva tri zahtjeva moraju biti istinita

klasa gumb

```
# button.py
from graphics import *

class Button:


 """A button is a labeled rectangle in a window.
 It is activated or deactivated with the activate()
 and deactivate() methods. The clicked(p) method
 returns true if the button is active and p is inside it."""

 def __init__(self, win, center, width, height, label):
     """ Creates a rectangular button, eg:
     qb = Button(myWin, Point(30,25), 20, 10, 'Quit') """
```

klasa gumb

```
w,h = width/2.0, height/2.0
x,y = center.getX(), center.getY()
self.xmax, self.xmin = x+w, x-w
self.ymax, self.ymin = y+h, y-h
p1 = Point(self.xmin, self.ymin)
p2 = Point(self.xmax, self.ymax)
self.rect = Rectangle(p1,p2)
self.rect.setFill('lightgray')
self.rect.draw(win)
self.label = Text(center, label)
self.label.draw(win)
self.deactivate()
```

```
def getLabel(self):
    "RETURNS the label string of this button."
    return self.label.getText()

def activate(self):
    "Sets this button to 'active'."
    self.label.setFill('black')
    self.rect.setWidth(2)
    self.active = 1
def deactivate(self):
    "Sets this button to 'inactive'."
    self.label.setFill('darkgrey')
    self.rect.setWidth(1)
    self.active = 0
```

```
def clicked(self, p):
    "RETURNS true if button active and p is inside"
    return self.active and \
           self.xmin <= p.getX() <= self.xmax and \
           self.ymin <= p.getY() <= self.ymax
```

GUI za kocku

```
class DieView:
    """ DieView is a widget that displays a graphical representation
    of a standard six-sided die."""

    def __init__(self, win, center, size):
        """Create a view of a die, e.g.:
           d1 = GDie(myWin, Point(40,50), 20)
        creates a die centered at (40,50) having sides
        of length 20."""

        # first define some standard values
        self.win = win              # save this for drawing pips later
        self.background = "white" # color of die face
        self.foreground = "black" # color of the pips
        self.psize = 0.1 * size    # radius of each pip
        hsize = size / 2.0         # half the size of the die
        offset = 0.6 * hsize       # distance from center to outer pips

        # create a square for the face
        cx, cy = center.getX(), center.getY()
```

GUI za kocku

```
p1 = Point(cx-hsize, cy-hsize)
p2 = Point(cx+hsize, cy+hsize)
rect = Rectangle(p1,p2)
rect.draw(win)
rect.setFill(self.background)

# Create 7 circles for standard pip locations
self.pip1 = self.__makePip(cx-offset, cy-offset)
self.pip2 = self.__makePip(cx-offset, cy)
self.pip3 = self.__makePip(cx-offset, cy+offset)
self.pip4 = self.__makePip(cx, cy)
self.pip5 = self.__makePip(cx+offset, cy-offset)
self.pip6 = self.__makePip(cx+offset, cy)
self.pip7 = self.__makePip(cx+offset, cy+offset)

# Draw an initial value
self.setValue(1)
```

metode

```
 def __makePip(self, x, y):
     "Internal helper method to draw a pip at (x,y)"
     pip = Circle(Point(x,y), self.psize)
     pip.setFill(self.background)
     pip.setOutline(self.background)
     pip.draw(self.win)
     return pip

def setValue(self, value):
    "Set this die to display value."
    # turn all pips off
    self.pip1.setFill(self.background)
    self.pip2.setFill(self.background)
    self.pip3.setFill(self.background)
    self.pip4.setFill(self.background)
    self.pip5.setFill(self.background)
    self.pip6.setFill(self.background)
    self.pip7.setFill(self.background)
    # turn correct pips on
    if value == 1:
        self.pip4.setFill(self.foreground)
    elif value == 2:
        self.pip1.setFill(self.foreground)
        self.pip7.setFill(self.foreground)
    elif value == 3:
        self.pip1.setFill(self.foreground)
        self.pip7.setFill(self.foreground)
        self.pip4.setFill(self.foreground)
    else:
        self.pip1.setFill(self.foreground)
        self.pip2.setFill(self.foreground)
        self.pip3.setFill(self.foreground)
        self.pip5.setFill(self.foreground)
        self.pip6.setFill(self.foreground)
        self.pip7.setFill(self.foreground)
```

Main

```
from random import randrange
from graphics import GraphWin, Point

from button import Button
from dieview import DieView

def main():

    # create the application window
    win = GraphWin("Dice Roller")
    win.setCoords(0, 0, 10, 10)
    win.setBackground("green2")

    # Draw the interface widgets
    die1 = DieView(win, Point(3,7), 2)
    die2 = DieView(win, Point(7,7), 2)
    rollButton = Button(win, Point(5,4.5), 6, 1, "Roll Dice")
    rollButton.activate()
    quitButton = Button(win, Point(5,1), 2, 1, "Quit")
```

Main

```
# Event loop
pt = win.getMouse()
while not quitButton.clicked(pt):
    if rollButton.clicked(pt):
        value1 = randrange(1,7)
        die1.setValue(value1)
        value2 = randrange(1,7)
        die2.setValue(value2)
        quitButton.activate()
    pt = win.getMouse()

# close up shop
win.close()
```

Na početku je roll gumb aktivan

U petlji se aktivira quit gumb ako kliknemo na roll gumb.

Korisnik je prisiljen da barem jednom baci kocku prije korištenja quit gumba.

Direktorij: vjezbe6, roller.py je glavni program, potrebni su graphics.py, dieview.py i button.py. Nismo komentirali graphics.py - klasu za Tk.

Graphics.py

graphics.pdf Graphics Module Reference

John M. Zelle

Version 3.2, Spring 2005

```
from graphics import *

def main():
    win = GraphWin("My Circle", 100, 100)
    c = Circle(Point(50,50), 10)
    c.draw(win)
    win.getMouse() # pause for click in window
    win.close()

main()
```

- **GraphWin Objects**
- Graphics Objects
- Point Methods
- Line Methods
- Circle Methods
- Rectangle Methods
- Oval Methods

Oval Methods

Text Methods

Image Manipulation

Literatura

- **An Introduction to Tkinter** by Fredrik Lundh
- http://effbot.org/tkinterbook/
- http://home.arcor.de/bernd.aberle/PythonPlot/index.htm
- http://ibiblio.org/obp/py4fun/index.html
- Python Programming: An Introduction to Computer Science, John M. Zelle, Ph.D.

zadaci

1. Napišite klasu Button koja napravi okrugle gumbe.
2. Napišite klasu za kocku, mora imati konstruktor i metode koje površinu i volumen.
3. Napišite klasu za sferu, kao u 2-om zadatku.
4. Proširite klasu (pod 2) inside koja odredi da li se određena točka nalazi u geometrijskom objektu.
5. Napišite klasu za simulaciju igre kartama. Napišite program koji podijeli slučajni niz karata i prikaže ih grafički.

Literatura

- **An Introduction to Tkinter** by Fredrik Lundh
- http://effbot.org/tkinterbook/
- http://home.arcor.de/bernd.aberle/PythonPlot/index.htm
- http://ibiblio.org/obp/py4fun/index.html