Tkinteri ülevaade

Siin on antud ainult lühikokkuvõte. Et õppida Tkinterit kasutama on vajalik osavõtt tundidest ja vajadusel veebimaterjalidega ([Py], [V]) tutvumine.

	Süntaks	**Näited**
Import	`from tkinter import *` `from tkinter.ttk import *`	

Juur	`Tk()`
	`.mainloop()`
		`root = Tk()`
		
		`root.mainloop()`

Geomeetria	`.grid()`
	`.rowconfigure()`
	`.columnconfigure()`
		`elem.grid(column=0, row=1,`
		`  sticky=(N, S, E, W))`
		
		`root.rowconfigure(0,`
		`  weight=1)`
		`root.columnconfigure(0,`
		`  weight=1)`

Üldine seadistus	`.config(`*`omadus=väärtus`*`)`
	
	`.option_add()`
	`.destroy()`
		`root.config(menu=menubar)`
		`combo.config(values=vlist)`
		
		`root.option_add("*tearOff",`
		`FALSE)`

Elemendi kustutamine, töö lõpetamine	`.destroy()`	`somelabel.destroy()` `root.destroy()`

Sündmused	`.bind(`*`sündmus, funkts`*`)`	`cmb.bind("<<ComboboxSelected>>",`
		`  minu_fn)`
		`lb.bind("<Enter>", funkts)`
		`lb.bind("<Leave>", funkts)`
		`lb.bind("<1>", funkts)`
		`canvas.bind("<Button-1>", fn)`

Raam	`Frame()`	`f = Frame(root)`

Nupp	`Button()`	`b = Button(root,`
		`  text="Arvuta",`
		`  command=funktsioon)`

Kangas	`Canvas()`	`c = Canvas(root)`

Kangale joonistamine	`.create_line()`
	`.create_rectangle()`
	`.create_polygon()`
	`.create_oval()`
	
	`.itemconfigure()`
	`.delete()`
		`id1=c.create_line(0, 10, 50, 50)`
		`id2=c.create_rectangle(0,0,9,9)`
		`id3=c.create_text(0,5,text="hi")`
		`c.itemconfigure(id2, fill="red")`
		`c.delete("all")`

Kilpkonn (omab kõiki turtle objekti meetodeid)	`RawTurtle()`	`import turtle` `konn = turtle.RawTurtle(c)`
Tekstikast	`Entry()`	`sisend = Entry(root,`

		`   textvariable=sisendvar)`
		
		`valjund = Entry(root,`
		`   textvariable=vvar,`
		`   state="readonly")`

Stringimuutuja	`StringVar()`
	`.get()`
	`.set()`
		`sisendvar = StringVar()`
		`vvar = StringVar()`
		`s = sisendvar.get()`
		`valjundvar.set("Tere "+s)`

Silt	`Label()`	`l = Label(root,` `    text="Pikkus:")`

Valikukast	`Combobox()`	`c = Combobox(root,`
		`   textvariable=sisendvar)`
		`c.config(values=vlist)`

Menüü	`Menu()`
	`.add_command()`
	`.add_cascade()`
		`root.option_add("*tearOff",`
		`FALSE)`
		
		`m = Menu(root)`
		`root.config(menu=m)`
		`mf = Menu(m)`
		`mf.add_command(label="Ava"`
		`    , command=ava_funkts)`
		`m.add_cascade(menu=mf,`
		`   label="Fail")`

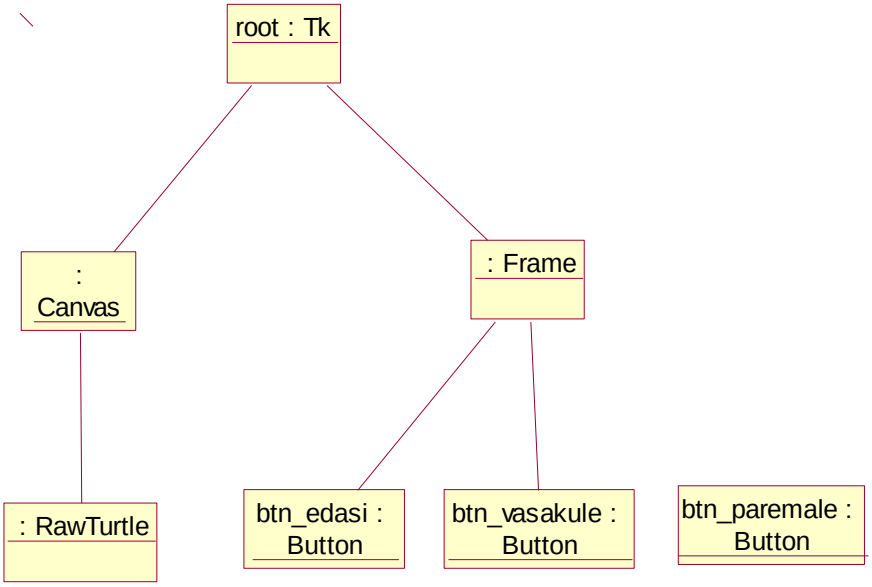
Standarddialoogid	`filedialog`
	`    .askopenfilename()`
	`    .asksaveasfilename()`
	`    .askdirectory`
	
	`messagebox`
	`    .showinfo()`
	`    .askyesno()`
		`from tkinter import filedialog`
		`fn=filedialog.askopenfilename()`
		
		`fn=filedialog.asksaveasfilename(`
		`   defaultextension=".csv",`
		`   filetypes=[("csv","*.csv")])`
		
		`from tkinter import messagebox`
		`jahei = messagebox.askyesno(`
		`    message="Format C:\?")`

Märkeruut	`Checkbutton()`	`valik_chk = Checkbutton(root,`
		`  text = "Paks kiri",`
		`  variable = valik_str,`
		`  onvalue = "Paks kiri",`
		`  offvalue = "Kaldkiri",`
		`  command = valik_muudetud_fn)`

Raadionupp	`Radiobutton()`	`keel = StringVar()`
		`eesti = Radiobutton(root,`
		`   text="Eesti",`
		`   variable=keel,`
		`   value="eesti")`
		`inglise = Radiobutton(root,`
		`   text="English",`
		`   variable=keel,`
		`   value="inglise")`

Ülevaade

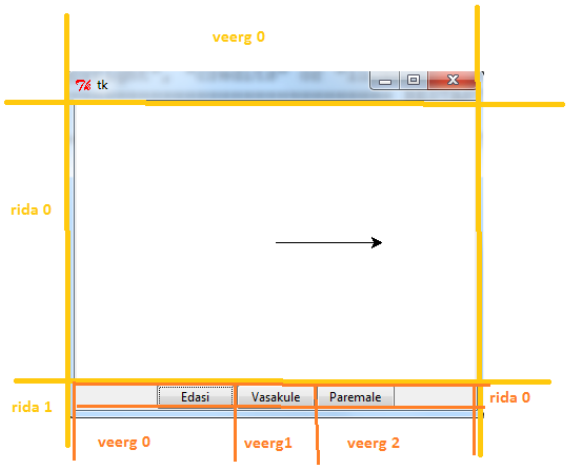
Graafiline kasutajaliides (*Graphical User Interface ehk GUI*) koosneb erinevat tüüpi objektidest, aknad, nupud, menüüd, tekstikastid, mis moodustavad puustruktuuri. Igal alamobjektil võib olla 0..1 ülemobjekti. Ülemobjekt on iga objekti tekitamisel esimene parameeter. Näiteks `c = Canvas(root)` tekitab objekti klassist Canvas (lõuend, tahvel) ja määrab selle ülemobjektiks akna objekti `root`.



Joonis 1. Tkinteri GUI puustruktuur objektidiagrammina.

Objektide omavahelise paigutuse ülemobjekti sees määrab tkinteri geomeetriahaldus (geometry manager). Siin vaatleme me levinud geomeetriahaldurit *grid* (ruudustik). Ruudustik jagab ülemobjekti veergudeks ja ridadeks, iga alamobjekti asukoht selles ruudustikus määratakse alamobjekti meetodi grid väljakutsega. Näiteks `c.grid(column=0, row=0, sticky=(N, S, W, E))` paigutab objekti c ülemobjekti (aken root) ritta 0 ja veergu 0 ja lisaks „kleebib“ selle objekti vastava ruudu kõigi nelja serva külge: kui ruutu suurendatakse, siis suureneb vastavalt ka objekt c. Veeru või rea suurendamine sõltub veeru või rea kaalust ülemobjektis. Kaalu määramiseks tuleb kutsuda välja ülemobjekti meetodeid `rowconfigure(rea_nr, weight=1)` või `columnconfigure(veeru_nr, weight=1)`. Kui kaal on 1, siis vaba ruumi tekkimisel seda rida või veergu suurendatakse.

Ruudustikku on võimalik organiseerida rekursiivselt: ülemobjekti ruudu sees võib olla oma ruudustik. Seda võimaldab klass *Frame* (raam).

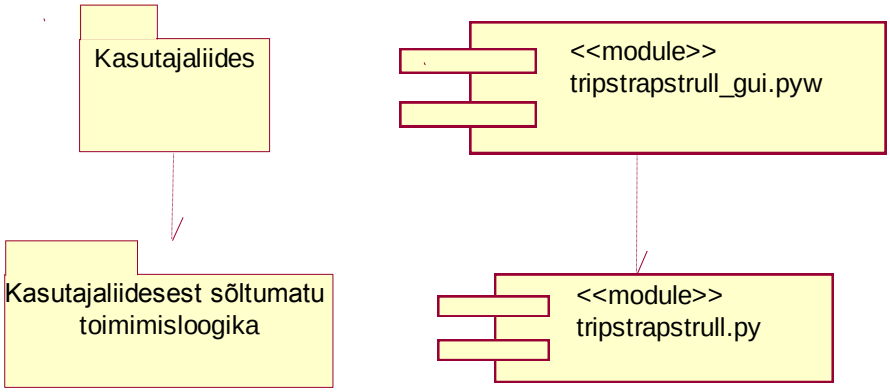


Joonis 2. Näide ruudustiku kohta. Akna ülemobjekti ruut rida=0, veerg=0 sisaldab Canvas tüüpi objekti. Akna ülemobjekti ruut rida=1, veerg=0 sisaldab Frame tüüpi objekti, millega on seotud oma alamruudustik (kolm veergu, üks rida), kuhu on paigutatud nupud.

Kasutajaliidesed on interaktiivsed, teatud kasutaja tegevuse (sündmuse) tagajärjel käivitatakse vastavad funktsioonid. Sobivad funktsioonid tuleb vastavate sündmustega ära siduda, kasutades nupu või menüü *command* omadust või *bind* meetodit. Funktsiooni tuleb seostamisel kasutada objektina, st anda ette funktsiooni nimi ilma sulgudeta.

Kihiline arhitektuur

Kihiline arhitektuur tähendab tarkvarasüsteemi jaotamist eraldiseisvateks kihtideks. Ülemised kihid sõltuvad alumistest, alumised ei sõltu ülemistest. Tüüpilisteks kihtideks on kasutajaliidese ja kasutajaliidesest sõltumatu toimimisloogika kiht. Pythonis võivad kihtidele vastata moodulid. Kasutajaliidese moodul kasutab ehk impordib toimimisloogika moodulit, mitte vastupidi.



Joonis 3. Kihiline arhitektuur üldiselt ja trips-traps-trulli Pythoni näites.

Sellise lähenemise eelised on:

1. Keerukuse haldamine. Kasutajaliidese ja toimimisloogika kood on eraldiseisvad, selge definitsiooniga osad terviksüsteemist.
2. Suurem paindlikkus ja korduvkasutuse soodustamine. Toimimisloogika moodulit saab kasutada erinevate kasutajaliidestega või ilma kasutajaliideseta. Kasutajaliidese vahetamine on lihtne.