5. Korisnička grafička sučelja

Svi programi koje smo do sada napisali oslanjali su se na tekstualna sučelja. Bilo kakva komunikacija s programom tražila je unošenje naredbi preko tipkovnice. Takav način komunikacije s programom smatran je zastarjelim još u ranim osamdesetim godinama prošlog stoljeća. Osim toga, programi s tekstualnim sučeljima zahtjevaju određeno predznanje od strane korisnika. Korisnik se mora upoznati s nizom naredbi koje program koristi, te ih upamtiti ukoliko želi da rad na takvom programu teče glatko i bez zadrški.

S druge strane, mogućnost programiranja koristeći grafička korisnička sučelja (*GUI – Graphical user interface*) otvara nam mogućnost pisanja programa za širu publiku. Programi s grafičkim sučeljima su intuitivniji, lakši za uporabu i korisnici se puno brže mogu naučiti raditi na njima.

5.1. Modul tkinter

U programskom jeziku Python postoji nekoliko alata za izradu grafičkih sučelja, no jedini alat koji je dostupan u osnovnom instalacijskom paketu je tkinter [*te-ka-inter*].

Tkinter je Pythonovo sučelje za Tk. Tk je open source alat koji se koristi u brojnim programskim jezicima za izradu grafičkih sučelja. Sučelje tkinter je implementirano u modulu *tkinter.py* koji je sastavni dio Pythona.

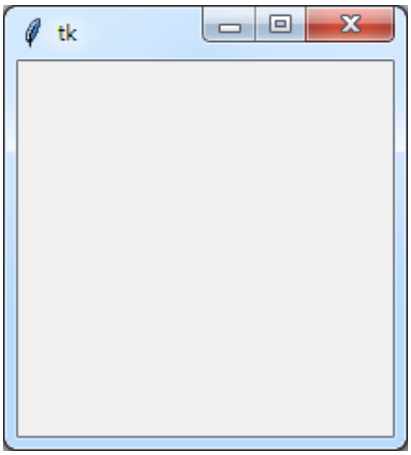
Kako bismo mogli započeti s izradom grafičkog sučelja moramo učitati modul *tkinter*.

```
from tkinter import *
```

Sada smo spremni za rad. Redom ćemo uvoditi naredbe za rad s grafičkim elementima. Neke od njih bit će nužno prihvatiti u obliku kakve jesu bez posebnog dodatnog objašnjenja.

5.1.1. Prozor

Osnovni element svakog grafičkog sučelja je **prozor**. Izrada prozora u tkinteru iznimno je jednostavna. Promotrimo sljedeći dio koda.




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

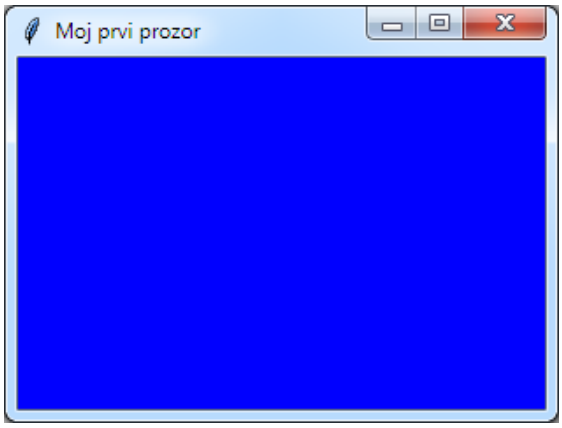
Nakon pokretanja ovog koda dobijamo naš prvi prozor.

Prozor je, iako prazan, sasvim funkcionalan. Na njemu se nalaze sva obilježja operacijskog sustava u kojem je kreiran uključujući i tri standardna gumba na vrhu prozora.

Objekt prozora kreirali smo naredbom *prozor = Tk()*. Na taj smo način kreirali jedan konkretan prozor kojeg ćemo zvati *prozor*. Posebno obratimo pozornost na zadnju liniju koda i metodu *mainloop()* koji smo pozvali. Čemu ona služi? Mainloop pokreće jednu beskonačnu petlju koja konstantno osluškuje sve događaje (pomicanje pokazivača miša, pritisak tipke na mišu, pritisak tipke na tipkovnici itd.) koji se događaju dok je prozor unutar fokusa. Bez pozivanja metode *mainloop()* pri pokretanju programa prozor se ne bi pojavio.

Naravno, ovo nije sve što se može napraviti na prozoru. Promotrimo sljedeći, malo složeniji kod.

```
from tkinter import *
prozor = Tk()
prozor.geometry('300x200+100+100')
prozor.resizable(False, True)
prozor.config(background = 'blue')
prozor.title('Moj prvi prozor')
prozor.mainloop()
```



Metodom *geometry()* definiramo veličinu prozora *prozor*. Pritom se 300x200 odnosi na dužinu i širinu prozora, a „+100+100“ na poziciju gornjeg lijevog kuta prozora unutar ekrana. Za početnu koodinatu (0,0) uzima se gornji lijevi kut ekrana.

Metoda *resizable()* specificira hoće li se prozor moći proširivati ili sužavati po širini i visini. Prvi parametar definira podešavanje po širini, a drugi po visini. Vrijednost False onemogućuje, a vrijednost True dozvoljava podešavanje.

Metoda *title()* postavlja naslov na prozor.

Metoda *config()* omogućuje podešavanje raznih opcija na wigetima. Background je jednja od njih. Svaki widget posjeduje ovu metodu.

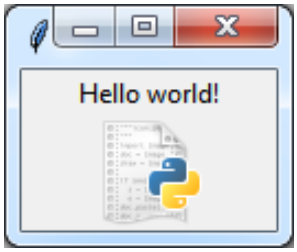
5.2. Grafički elementi (*eng. Widget*)

Iako smo vidjeli kako prozor možemo učiniti ljepšim, on je i dalje poprilično prazan. Tkinter pruža nekolicinu widgeta (visual gadgets). Ovu riječ je najbolje prevesti kao grafički element. Ipak, nadalje ćemo koristiti izraz widget. Neke od njih ćemo obraditi u nastavku.

5.2.1. Naljepnica (*eng. Label*)

Naljepnica je widget koji se koristi za prikazivanje kratkog teksta ili slike. Sljedeći primjer pokazuje kako se kreira jednostavna naljepnica s kratkim tekstom i naljepnica sa slikom.

```
from tkinter import *
prozor = Tk()
labelTekst = Label(prozor, text = 'Hello world!')
slika = PhotoImage(file = 'idle.png')
labelSlika = Label(prozor, image = slika)
labelTekst.pack()
labelSlika.pack()
prozor.mainloop()
```



Linija koda `labelTekst = Label(prozor, text = 'Hello world!')` kreira novu naljepnicu kojoj je roditelj naš prozor i koja u sebi ima tekst „Hello world“, dok linija koda `labelSlika = Label(image = slika)` kreira naljepnicu sa slikom koju smo prethodno

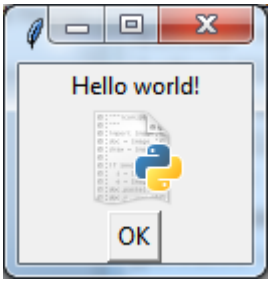
učitali u program linijom koda `slika = PhotoImage(file = 'idle.png').` Jasno je da se ta slika treba nalaziti u istom direktoriju gdje se nalazi i kod programa. Treba napomenuti da *tkinter* prihvaća samo slike s ekstenzijom *png* i *gif*. Prilikom kreiranja widgeta potrebno je na prvom mjestu unutar zagrada, kao prvi parametar istaknuti kojem prozoru widget pripada, jer ukoliko imamo više prozora napravit će se prava zbrka s izgubljenim widgetima.

Ključne linija koda u ovom primjeru su `labelTekst.pack()` i `labelSlika.pack()`. Svaki widget posjeduje metodu *pack()* koja ih postavlja na roditeljski prozor. Metoda *pack()* postavlja widgete jedan ispod drugoga na prozor, ukoliko joj ne kažemo drugačije. Postoji još metoda za postavljanje widgeta na prozor, ali o njima ćemo kasnije.

5.2.2. Gumb (*eng. Button*)

Gumb je jedan od najbitnijih widgeta kod izrade grafičkih sučelja. Gumbima možemo inicirati aktivnosti na prozoru. Dodajmo u prethodni primjer sljedeće naredbe:

```
gumb = Button(prozor, text = 'OK')
gumb.pack()
```



Dobili smo gumb na našem prozoru!

Gumb sam po sebi ne radi ništa. Kako bi ga aktivirali možemo iskoristiti jednu od metoda za rad s gumbima. Na primjer, kada bi pritiskom na gumb htjeli promijeniti pozadinsku boju prozora te promijeniti tekst unutar naljepnice učinili bi to na sljedeći način. Dodajmo u prethodni primjer sljedeće linije koda.

```
def gumbAkcija(event):
    prozor.config(bg = 'yellow')
    labelTekst.config(text = 'Dobar dan')
    return
...
gumb.bind('<Button>', gumbAkcija)
...
```

Prvim parametrom u metodi *bind()* definiramo na koji događaj koji se dogodi na gumbu će metoda reagirati. U ovom slučaju koristimo parametar *'<Button>'* koji označava pritisak tipke miša. Drugi parametar u metodi je ime funkcije koja se pokreće u trenutku kada se dogodi zadani događaj. Zahvaljujući tome pritiskom na gumb oživljavamo prozor.

Postoji još nekoliko vrsta događaja koji se mogu zadavati kao prvi parametar metodi *bind()*. Neki od njih su:

Naziv parametra	Funkcija se pokreće:
<Button>	na pritisak gumba tipkom miša

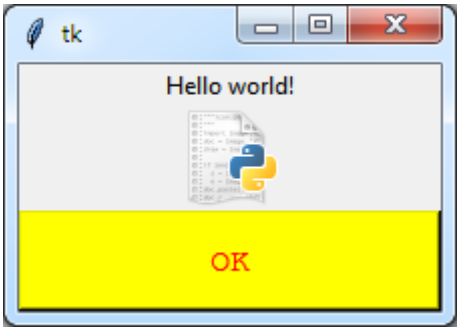
<Motion>	na pokret pokazivača miša nad gumbom
<ButtonRelease>	na puštanje pritisnutog gumba
<Double-Button>	na dvostruki klik nad gumbom
<Enter>	na ulaz pokazivača nad gumb
<Leave>	po izlasku pokazivača sa gumba
<InFocus>	po dolasku gumba u fokus

Prilikom kreiranja gumba koristili smo atribut *text* za definiranje teksta koji će se pojaviti na gumbu. To nije jedini atribut. Upoznajmo još neke od njih.

bg ili background	Postavlja željenu boju pozadine gumba
fg ili foreground	Postavlja boju teksta na gumbu
font	Postavlja željeni font na gumb
height i width	Specificira visinu i širinu gumba u pikselima
image	Postavlja objekt tipa PhotoImage na gumb
padx i pady	Proširuje prazan prostor oko teksta na gumbu za navedenu vrijednost

Ukoliko ove atribute ne želimo postaviti prilikom kreiranja gumba uvijek ih možemo postaviti pozivajući metodu *config*. Na primjer:

```
gumb.config(bg = 'yellow', fg = 'red', font = 'Courier', width = 20, pady = 10)
```

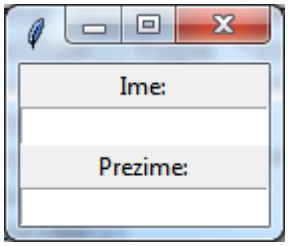


5.2.3. Unos (*eng. Entry*)

Svrha Unos widgeta je prikazivanje i modificiranje jedne linije teksta. Unos je savršen za kreiranje dijelova grafičkih sučelja u kojima korisnik mora unijeti kratke linije teksta. Napravit ćemo jedan primjer obrasca za unos imena i prezimena.

```
from tkinter import *

prozor = Tk()
```

```
labelIme = Label(prozor, text = 'Ime:')
labelPrezime = Label(prozor, text = 'Prezime:')
entryIme = Entry(prozor)
entryPrezime = Entry(prozor)
labelIme.pack()
entryIme.pack()
labelPrezime.pack()
entryPrezime.pack()
prozor.mainloop()
```



U ovaj obrazac korisnik može unijeti svoje ime i prezime. Uoči da smo uzastopnim primjenama metode *pack()* na objektima složili te objekte u prikazanom redoslijedu. Međutim, kako program može pročitati ono što je korisnik napisao i kako sam program može upisati nešto u Unos? Za te aktivnosti iznimno su nam korisne metode koje widget Unos posjeduje.

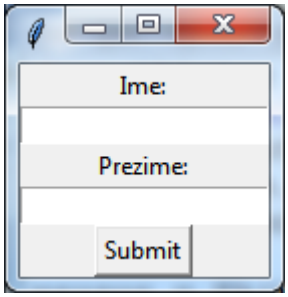
get() – vraća trenutni tekst koji se nalazi u Unos-u u obliku stringa

insert(indeks, s) – unosi string s u Unos na prvo mjesto prije indeksa *index*.

delete(prvi, zadnji) – briše znakove iz Unos-a počevši od prvog indeksa do *zadnji-1* indeksa

Dodajmo u prethodni primjer sljedeće naredbe:

```
def gumbSubmitAkcija(entry):
    print(entryIme.get() + ' ' + entryPrezime.get())
    entryIme.delete(0, END)
    entryPrezime.delete(0, END)
    return
...
gumbSubmit = Button(prozor, text = 'Submit')
gumbSubmit.bind('<Button>', gumbSubmitAkcija)
...
gumbSubmit.pack()
```

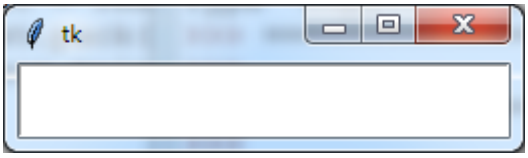


Na pritisak gumba Submit (zbog parametra *'<Button>'*), metoda *bind()* pokreće funkciju *gumbSubmitAkcija* koja pročita sve što je napisano u oba Unos-a te sadržaj ispiše na ekran. Nakon toga funkcija obriše sadržaj iz Unos-a. Indeks *0* u metodi *delete()* predstavlja početni indeks stringa u Unos-u, dok index *END* predstavlja poziciju iza posljednjeg znaka u stringu.

5.2.4. Text widget

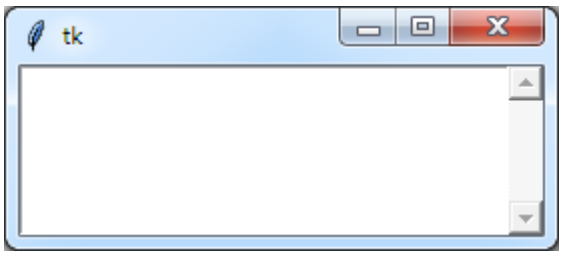
Text widget se koristi kao prostor za zapisivanje više linija teksta. Ovo je dosta moćan widget jer se od njega mogu napraviti prostori za pisanje raznih veličina, od širine jedne linije teksta do širine cijelog ekrana.

```
from tkinter import *
prozor = Tk()
tekst = Text(prozor, height = 2, width = 30)
tekst.pack()
prozor.mainloop()
```



Naredbom `tekst = Text(prozor, height = 2, width = 30)` kreiramo jedan text widget koji je širine dva reda teksta i duljine trideset znakova. Sav tekst koji je dulji od toga će se napisati, ali će se prikazivati samo dva posljednja retka. Po napisanom tekstu možemo se kretati strelicama na tipkovnici, ali to nije najelegantniji način. Uz tekst widget možemo kreirati novi widget, pomičnu traku, koji nam pomaže u kretanju po tekstu. Taj widget zove se Scrollbar. Pogledajmo primjer vezanja scrollbar widgeta i text widgeta.

```
from tkinter import *
prozor = Tk()
tekst = Text(prozor, height = 5, width = 30)
traka = Scrollbar(prozor)
traka.config(command = tekst.yview)
tekst.config(yscrollcommand = traka.set)
traka.pack(side = RIGHT, fill = Y)
tekst.pack(side = LEFT, fill = Y)
prozor.mainloop()
```



Povezivanje dva widgeta odvija se u *config()* metodama jednog i drugog na navedeni način. Potom se oba widgeta postave na prozor metodama *pack()* kojima smo dodatno napisali atribute *side* i *fill*. Atribut *side* definira na kojoj strani prozora će widget biti, a atribut *fill = Y* određuje da će se widget proširiti po cijeloj visini prozora, ma koliko prozor bio velik.

Kao i kod Unos widgeta i text widget ima mnoštvo metoda. Izdvojit ćemo samo neke:

delete(indeks1, indeks2) – metoda briše znakove za widgeta od *indeksa1* do *indeksa2* uključivo

get(index1, index2) – metoda vraća tekst u obliku stringa koji se nalazi između dva indeksa

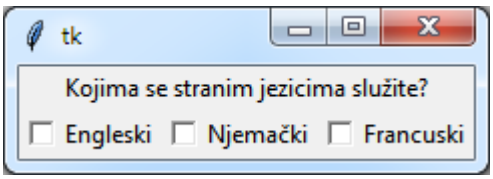
insert(index1, t) – metoda unosi tekst t na widget počevši od mjesta *index1*

5.2.5. Checkbutton i radiobutton

Checkbutton, poznat i kao checkbox je widget koji omogućuje korisniku višestruki odabir od nekoliko ponuđenih izbora. Jedan checkbutton ima dva stanja: neoznačeno i označeno. Ovaj widget je jako često u uporabi kako u aplikacijama tako i na web stranicama.

Kreirajmo jedan par checkbutton-a.

```
from tkinter import *

prozor = Tk()

izborEng = IntVar()

izborGer = IntVar()

izborFre = IntVar()

pitanje = Label(prozor, text = 'Kojima se stranim jezicima
služite?')

pitanje.pack()

Checkbutton(prozor, text = 'Engleski', variable =
izborEng).pack(side = LEFT)

Checkbutton(prozor, text = 'Njemački', variable =
izborGer).pack(side = LEFT)

Checkbutton(prozor, text = 'Francuski', variable =
izborFre).pack(side = LEFT)

prozor.mainloop()
```



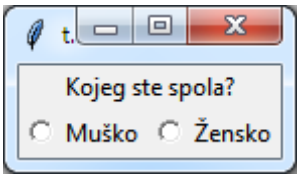
Kao što vidite, checkbutton se ne pridružuje varijabli jer se izbor pojedinog checkbuttona pamti u posebnoj varijabli, objektu tipa *IntVar*. Ako je checkbutton označen onda se u pripadnom *IntVar* objektu nalazi jedinica, ako nije, nalazi se nula. Vrijednosti koja je zapisana u *IntVar* objektu pristupamo uz pomoć metode *get()*. Na primer:

```
izbor = izborEng.get()
```

Kod radiobuttona je moguće odabrati samo jedan izbor od nekoliko ponuđenih. Uporaba radiobuttona je jako slična uporabi checkbuttona uz jednu razliku. Kako je moguće odabrati samo jedan izbor, tada imamo samo jedan objekt tipa *IntVar* u kojem će se čuvati informacija koji je izbor odabran.

```
from tkinter import *

prozor = Tk()

izbor =IntVar()

pitanje = Label(prozor, text = 'Kojeg ste spola?')

pitanje.pack()

Radiobutton(prozor, text = 'Muško', variable = izbor, value =
1).pack(side = LEFT)
```

```
Radiobutton(prozor, text = 'Žensko', variable = izbor, value =
2).pack(side = LEFT)
prozor.mainloop()
```

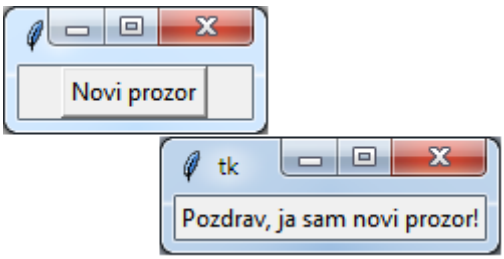


Uoči da smo prilikom kreiranja radiobuttona koristili atribute *text*, za definiranje teksta uz radiobutton, *value*, za definiranje vrijednosti koju nosi radiobutton, te *variable* za definiranje varijable u kojoj će ta vrijednost biti upisana. Svaki radiobutton mora imati različite vrijednosti u atributu *value*. Kao što vidite, moguće je odabrati samo jedan izbor i do njega možemo doći opet koristeći metodu *get()* na objektu *izbor*.

5.2.6. Toplevel

Često pri izradi aplikacija s grafičkim sučeljima potrebno je kreirati novi prozor iz već postojećeg prozora. To se može obaviti naredbom *Toplevel*. U sljedećem primjeru ćemo na pritisak gumba kreirati novi prozor.

```
from tkinter import *
def gumbAkcija(event):
    noviProzor = Toplevel(prozor)
    poruka = Label(noviProzor, text = 'Pozdrav, ja sam novi
prozor!')
    poruka.pack()
    noviProzor.mainloop()
    return
prozor = Tk()
gumb = Button(prozor, text = 'Novi prozor')
gumb.bind('<Button>', gumbAkcija)
gumb.pack()
prozor.mainloop()
```

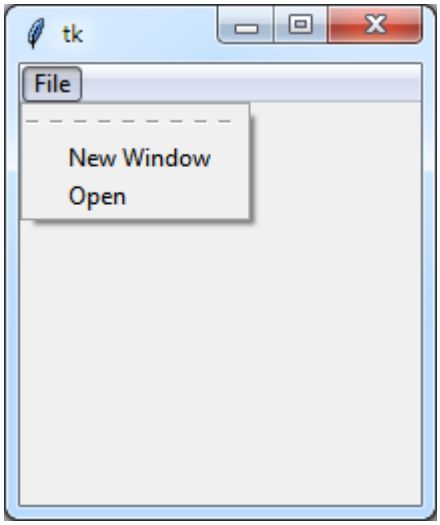


Novi prozor je dijete svom roditeljskom prozoru i bez njega ne može postojati. Ukoliko se zatvori prozor roditelj tada će se zatvoriti i prozor dijete. Primjetimo da smo pri kreiranju

poruke na novom prozoru naznačili da naljepnica pripada novom prozoru. Ukoliko to ne naznačimo poruka će se poslije pritiska gumba za novi prozor pojaviti u roditeljskom prozoru.

5.2.7. Izbornici *(eng. menu bar)*

Izbornik je sastavni dio računalnih aplikacija još od početka izrađivanja programa s grafičkim sučeljima. Izbornik je koristan način pospremanja raznih funkcionalnosti na prostorno štedljiv način.




```
from tkinter import *
def noviProzor():
    novi = Toplevel()
    return
def otvori():
    filedialog.askopenfilename()
    return
prozor = Tk()
menubar = Menu(prozor)
podmenu = Menu(menubar)
podmenu.add_command(label = 'New Window', command = lambda :
noviProzor())
podmenu.add_command(label = 'Open', command = lambda : otvori())
menubar.add_cascade(label = 'File', menu = podmenu)
prozor.config(menu = menubar)
prozor.mainloop()
```

Novi izbornik kreiramo naredbom `menubar = Menu(prozor)` te ga vežemo uz prozor, zatim kreiramo još jedan izbornik naredbom `podmenu = Menu(menubar)` i vežemo ga za prethodni izbornik. Tako smo stvorili mogućnost padajućeg izbornika. Potom dodajemo nove kategorije u podmenu, a to su *NewWindow* i *Open* te ih pomoću *command* opcije i *lambda* funkcije povezujemo s pripadnim funkcijama koje pokreću odgovarajuće akcije. Ovdje je prilika podsjetiti da se neke naredbe trebaju samo koristiti bez dubljeg ulaženja u njihovo značenje.

5.3. Upravitelji rasporeda (*eng. Layout menagers*)

Dosada smo se upoznali s jednim načinom postavljanja widgeta na prozor. To je bila metoda *pack()* koja je najjednostavnija za uporabu. Kod nje ne definiramo precizno gdje će se pojaviti widget na prozoru već samo okvirno i to u odnosu na druge widgete. Ipak neke stvari kod metode *pack()* možemo preciznije definirati.

Koristeći *fill* opciju možemo specificirati hoće li widget ispunjavati cijelu dužinu ili širinu prozora. Na primjer:

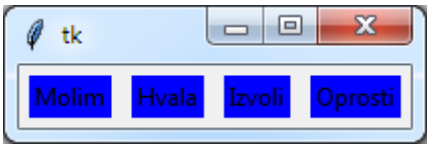
```
from tkinter import *
```

```
prozor = Tk()
poruka1 = Label(prozor, bg = 'blue', text = 'Molim')
poruka2 = Label(prozor, bg = 'blue', text = 'Hvala')
poruka3 = Label(prozor, bg = 'blue', text = 'Izvoli')
poruka4 = Label(prozor, bg = 'blue', text = 'Oprosti')
poruka1.pack()
poruka2.pack(fill = X)
poruka3.pack(fill = X)
poruka4.pack()
prozor.mainloop()
```

Poruka „Hvala“ i „Izvoli“ je proširena na cijelu veličinu ekrana. Njihova širina će uvijek pratiti širinu prozora.

Sljedeća opcija koju možemo koristiti je *side*. Njome određujemo na kojoj strani prozora će se widget smjestiti. Ukoliko promijenimo dio prethodnog koda u sljedeći:

```
poruka1.pack(side = LEFT, padx = 5, pady = 5)
poruka2.pack(side = LEFT, padx = 5, pady = 5)
poruka3.pack(side = LEFT, padx = 5, pady = 5)
poruka4.pack(side = LEFT, padx = 5, pady = 5)
```

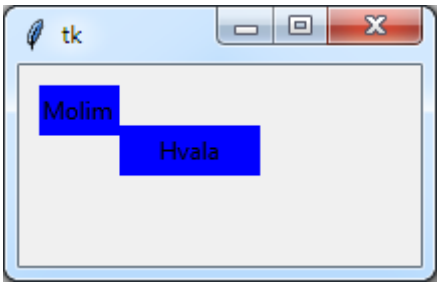


Sve poruke teže smještanju na lijevu stranu, stoga su sve poredane u jedan red. Opcije *padx* i *pady* pružaju mogućnost proširenja praznog prostora oko widgeta, što radi preglednosti može biti jako korisno.

Sljedeća metoda postavljanja widgeta na prozor je *place()*. Ona je sušta suprotnost *pack()* načinu postavljanja. Kod metode *place()* definiramo točnu veličinu i položaj widgeta na prozoru.

```
from tkinter import *
prozor = Tk()
prozor.geometry('200x100+100+100')
```

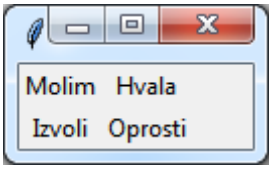
```
poruka1 = Label(prozor, bg = 'blue', text = 'Molim')
poruka2 = Label(prozor, bg = 'blue', text = 'Hvala')
poruka1.place(x = 10, y = 10, width = 40, height = 25)
poruka2.place(x = 50, y = 30, width = 70, height = 25)
prozor.mainloop()
```



Ovaj način postavljanja widgeta na prozor programeri izbjegavaju jer je izrazito nefleksibilan.

Posljednja metoda za postavljanje elemenata na prozor je *grid()*. Grid postavlja dane mu widgete u dvodimenzionalnu strukturu, poput matrice.

```
from tkinter import *
prozor = Tk()
poruka1 = Label(prozor, text = 'Molim')
poruka2 = Label(prozor, text = 'Hvala')
poruka3 = Label(prozor, text = 'Izvoli')
poruka4 = Label(prozor, text = 'Oprosti')
poruka1.grid(row = 0, column = 0)
poruka2.grid(row = 0, column = 1)
poruka3.grid(row = 1, column = 0)
poruka4.grid(row = 1, column = 1)
prozor.mainloop()
```



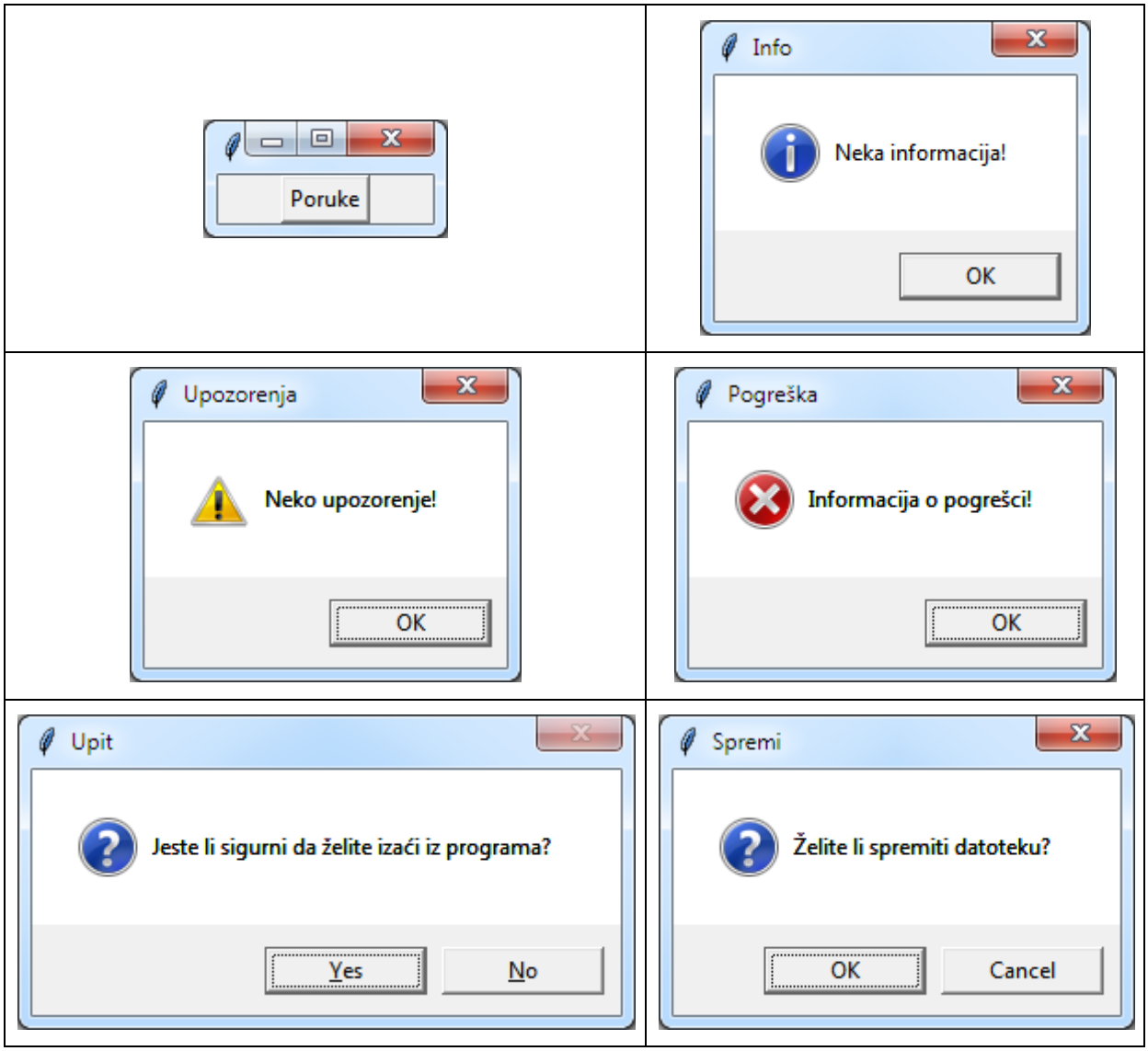
Metoda *grid* prima dva atributa *row* i *column* u kojima određujemo na koordinate pojedinog widgeta.

5.4. Poruke

Tkinter posjeduje nekoliko standardnih prozorčića kojima možemo prikazivati kratke poruke korisniku te zahtjevati potvrdu od njega.

```
from tkinter import *
def gumbAkcija(event):
```

```
    messagebox.showinfo('Info', 'Neka informacija!')
    messagebox.showwarning('Upozorenja','Neko upozorenje!')
    messagebox.showerror('Pogreška','Informacija o pogrešci!')
    messagebox.askyesno('Upit','Jeste li sigurni da želite izaći iz
programa?')
    messagebox.askokcancel('Spremi','Želite li spremiti datoteku?')
    return
prozor = Tk()
gumb = Button(text = 'Poruke')
gumb.bind('<Button>', gumbAkcija)
gumb.pack()
prozor.mainloop()
```



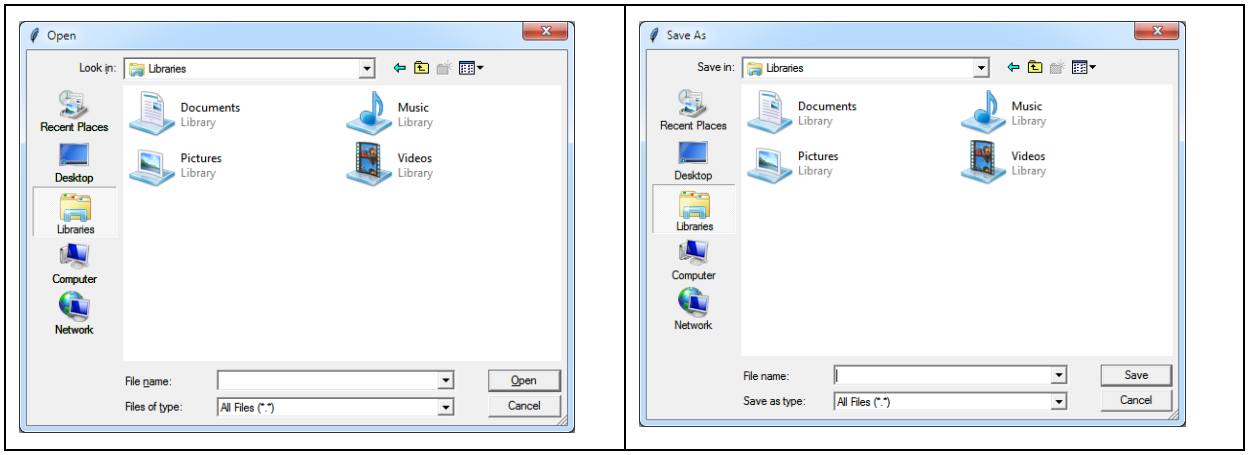
Funkcijom *gumbAkcija* realizirali smo otvaranje ovih prozorčića s pripadajućim porukama. Pri tome smo koristili sljedeće naredbe iz paketa messagebox:

- sa *showinfo* otvorili smo informacijski prozorčić sa slovom *i* u plavom krugu;
- sa *showwarning* otvorili smo prozorčić upozorenja s uskličnikom u žutom trokutu;
- sa *showerror* otvorili smo prozorčić o pogrešci sa slovom x crvenom krugu;
- s *askyesno* otvorili smo prozorčić s izborom Da/Ne odgovora na postavljeno pitanje;
- s *askokcancel* otvorili smo prozorčić s OK/Cancel odgovor na postavljeno pitanje.

Tkinter pruža i mogućnost poziva dijaloga za otvaranje datoteka ili spremanje datoteka koji su ugrađeni u operativni sustav na kojem izvršavamo program.

```
odabranaDatoteka = filedialog.askopenfilename()
```

```
odabranaPutanja = filedialog.asksaveasfilename()
```



Ovi dijalozi su posebno korisni jer nam pružaju uporabu kompletne funkcionalnosti operacijskog sustava pri otvaranju i spremanju datoteka.

5.5. Od problema do rješenja

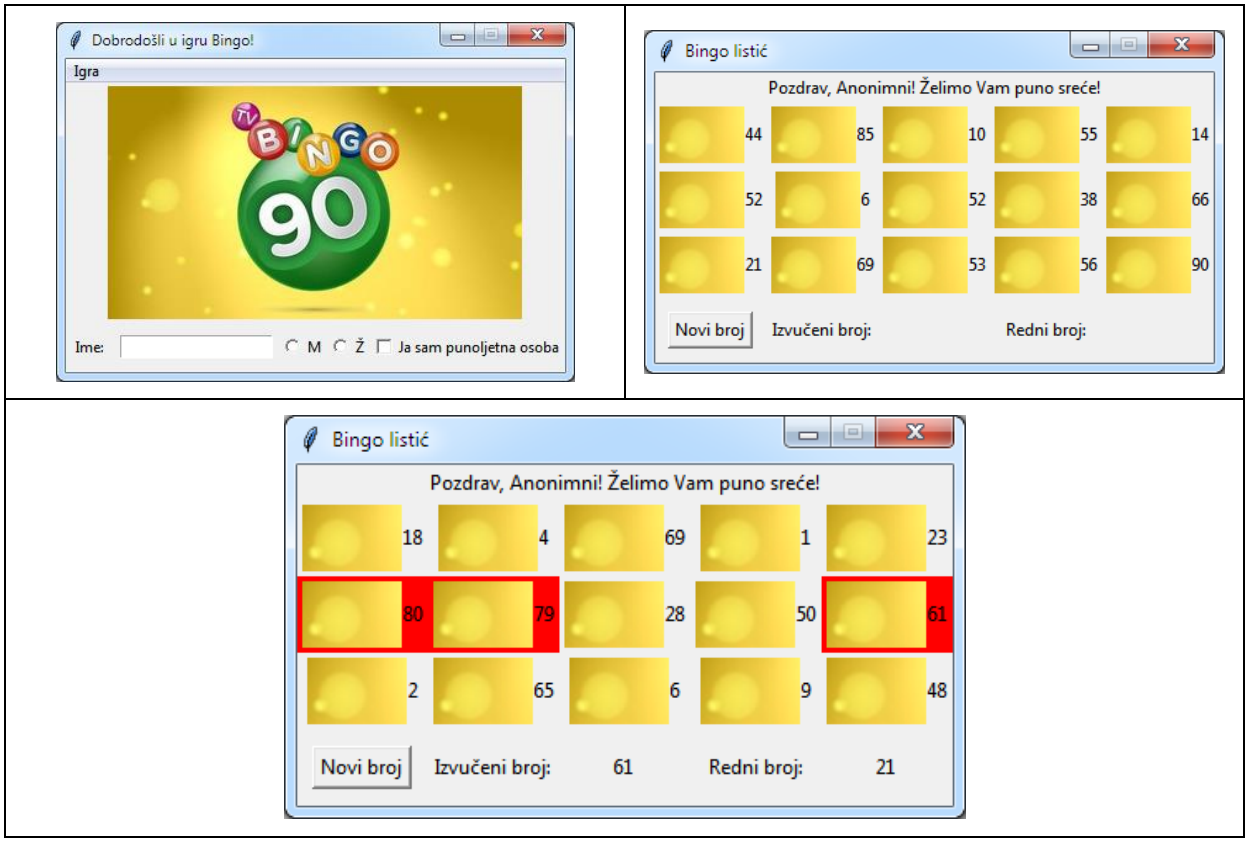
Nakon što smo uveli neke od najznačajnijih elemenata za rad s grafičkim sučeljima, vjerojatno smo u stanju osmisliti i izraditi jedan projekt. Tijekom izrade projekta trebat ćemo koristiti većinu uvedenih elemenata.

5.5.1. Ideja projekta

Bingo je igra na sreću. Opišimo jednu verziju ove igre koju ćemo u nastavku implementirati. Igrač na početku igre dobije listić s 15 brojeva. Svi brojevi su veći od jedan i manji od 90 (uključivo) te se mogu ponavljati. Voditelj igre iz bubnja, u kojem se nalaze svi brojevi od jedan do 90, redom izvlači po jedan od njih. Svaki put kada voditelj izvuče jedan broj, igrač provjerava ima li tog broja na njegovom listiću. Ako ima, nekako ga treba označiti. Cilj igre je što prije označiti sve brojeve na listiću. Ovisno o tome koji je po redu bio zadnji označeni broj na listiću, igrač dobiva pripadajuću nagradu. Pri tome vrijedi ako igrač uspije označiti sve brojeve prije:

- 30. izvučenog broja iz bubnja, tada dobiva „SuperBingo“;
- 33. izvučenog broja iz bubnja, tada dobiva „Bingo33“;
- 36. izvučenog broja iz bubnja, tada dobiva „Bingo36“;
- 39. izvučenog broja iz bubnja, tada dobiva „Bingo39“;

Ako mu to uspije nakon 40. izvučenog broja, tada dobiva „Bingo40+“. U konačnici, naš program bi trebao izgledati kao što je prikazano na slikama.



Nakon što smo definirali pravila ove igre, krenimo u njeno implementiranje.

5.5.2. Realizacija projekta

Projekt ćemo započeti izradom početnog prozora s jednom naljepnicom koja nosi sliku igre na sreću. Početni prozor će imati jedan izbornik na sebi, te prostor za unos imena, spola i potvrdu da je korisnik aplikacije punoljetan. Potvrda starosti ključni je dio jer bez toga korisnik neće moći nastaviti igru. Unos imena, te spola je opcionalan i nije obvezan za nastavak rada programa.

Krenimo redom. U ovom dijelu koda kreirat ćemo prozor, postaviti mu naslov i onemogućiti promjenu veličine prozora. Potom ćemo pripremiti i postaviti sliku *Bingo15_90.png* na *labelSlika*, te istu tu naljepnicu pomoću metode *pack()* postaviti na prozor.

```
prozor = Tk()
prozor.title('Dobrodošli u igru Bingo!')
prozor.resizable(False, False)
slika = PhotoImage(file = 'Bingo15_90.png')
labelSlika = Label(image = slika)
labelSlika.pack()
```

Zatim ćemo kreirati izbornik i dvije kategorije u njemu, Novi listić i Pravila igre. Njih ćemo povezati s funkcijama *noviListić(prozor)* i *pravilaIgre(prozor)*. Tim dvjema funkcijama ćemo se posvetiti malo kasnije.

```
menubar = Menu(prozor)

igramenu = Menu(menubar)

igramenu.add_command(label = 'Novi listić', command = lambda :
noviListic(prozor))

igramenu.add_command(label = 'Pravila igre', command = lambda :
pravilaIgre(prozor))

menubar.add_cascade(label = 'Igra', menu = igramenu)

prozor.config(menu = menubar)
```

Kreiranjem Unos-a za unos imena, radio gumba za odabir spola i check gumba završit ćemo izgled glavnog prozora.

```
labelIme = Label(prozor, text = 'Ime:')

labelIme.pack(side = 'left', padx = 5, pady = 10)

entryIme = Entry(prozor)

entryIme.pack(side = 'left', padx = 5, pady = 10)

spol = IntVar()

Radiobutton(prozor, text = 'M', variable = spol, value =
1).pack(side = 'left')

Radiobutton(prozor, text = 'Ž', variable = spol, value =
2).pack(side = 'left')

punoljetan = IntVar()

Checkbutton(prozor, text = 'Ja sam punoljetna osoba', variable =
punoljetan).pack(side = 'left')

prozor.mainloop()
```

Daljnja funkcionalnost programa nastavlja se odabirom kategorije iz izbornika. Ukoliko korisnik odabere kategoriju Pravila igre, pokrenut će se funkcija *pravilaIgre(prozor)* koja prima kao argument ime prozora iz kojeg pozivamo funkciju.

Funkcija *pravilaIgre()* otvara novi prozor u kojem se nalazi text widget s pomičnom trakom i tekstom pravila igre.

```
def pravilaIgre(roditelj):
    pravila = Toplevel(roditelj)
    pravila.title('Pravila igre')
    pravila.resizable(False, False)
    tekst = 'Na licu svakog listića nalazi se petnaest (15)
različitih brojeva, iz niza od 1 do 90.\nBINGO \nPetnaest pogođenih
```

```
brojeva u sva tri reda unutar jedne kombinacije:\ndo zaključno 30.
izvučenog broja – SUPERBINGO\nod 31. do 33. izvučenog broja – BINGO
33\nod 34. do 36. izvučenog broja – BINGO 36\nod 37. do 39.
izvučenog broja – BINGO \n39od 40. izvučenog broja do ostvarenja
dobitka – BINGO 40+'
    S = Scrollbar(pravila)
    T = Text (pravila, height = 5, width = 60)
    S.pack(side = 'right', fill = Y)
    T.pack(side = 'left', fill = Y)
    S.config(command = T.yview)
    T.config(yscrollcommand = S.set)
    T.insert(END, tekst)
    return
```

Ukoliko korisnik odabere kategoriju Novi listić iz izbornika tada će se pokrenuti funkcija *noviListic(prozor)*. Na početku ove funkcije trebamo provjeriti je li osoba punoljetna provjeravajući je li check gumb pritisnut. Ukoliko nije pritisnut onda se javlja upozorenje i funkcija završava s radom. Inače, ako je osoba punoljetna kreiramo novi prozor s Bingo listićem.

```
def noviListic(root):
    listaIzvucenih = []
    if(punoljetan.get() != 1):
        messagebox.showwarning('Upozorenje!', 'Maloljetnim osobama
je zabranjeno igranje igre na sreću!')
        return
    else:
        listic = Toplevel(root)
        listic.title('Bingo listić')
        listic.resizable(False, False)

        if(entryIme.get() == ''):
            ime = 'Anonimni'
        else:
            ime = entryIme.get()
        labelPozdrav = Label(listic, text = 'Pozdrav, ' + ime + '!
Želimo Vam puno sreće!')
        labelPozdrav.grid(row = 0, column = 0, columnspan = 5)
```

Dodatno smo kreirali pozdravnu poruku pomoću jedne naljepnice koju smo nazvali *labelPozdrav* te je postavili u matricu rasporeda u prvi red. Opcijom *columnspan=5* naglašavamo da će prvi red biti širine pet stupaca.

Sada ćemo napraviti matricu provjere koja je veličine 3x5 te je popunjena nulama. Matrica provjere će nam služiti kao zapis na kojem je mjestu pogođen broj u igri na sreću.

```
    matricaProvjere = []
        redak = []
        for i in range(3):
            for j in range(5):
                redak.append(0)
            matricaProvjere.append(redak)
            redak = []
```

Pošto se Bingo listić sastoji od matrice 3x5 s brojevima, kreiramo novu matricu 3x5 koja u sebi sadrži naljepnice. Naljepnice imaju na sebi i tekst i sliku. Osobito je važna opcija *compound='left'* jer ona specificira na kojoj će strani biti smještena slika unutar naljepnice. U ovom slučaju slika je postavljena lijevo od broja.

```
    labeli = []
        redak = []
        slikaBroj = PhotoImage(file = 'Bingo_broj.png')
        for i in range(3):
            for j in range(5):
                redak.append(Label(listic, text = randint(1, 90),
image = slikaBroj, compound = 'left'))
            labeli.append(redak)
            redak = []


        for i in range(3):
            for j in range(5):
                labeli[i][j].grid(row = i+1, column = j)
```

Potom pravimo gumb na kojem piše novi broj i pomoću metode *bind()* povezujemo ga s funkcijom *noviBrojAkcija* koju ćemo napisati kasnije. Kreiramo naljepnice na kojima će nam pisati trenutno izvučeni broj i koji je redni broj izvučenog broja.

```
        noviBroj = Button(listic, text = 'Novi broj')
        noviBroj.bind('<Button>', noviBrojAkcija)
        noviBroj.grid(row = 4, column = 0, pady = 10)
        izvuceniBrojLabel = Label(listic, text = 'Izvučeni broj:')
```

```
        izvuceniBrojLabel.grid(row = 4, column = 1, pady = 10)
        prikazBroja = Label(listic)
        prikazBroja.grid(row = 4, column = 2, pady = 10)
        redniBrojLabel = Label(listic, text = 'Redni broj:')
        redniBrojLabel.grid(row = 4, column = 3, pady = 10)
        brojIzvucenih = Label(listic)
        brojIzvucenih.grid(row = 4, column = 4, pady = 10)
        listic.mainloop()
        return
```

Funkcija *noviBrojAkcija* ima za ulogu odabir slučajnog broja i provjeru je li neki broj na listiću identičan slučajnom broju. U isto vrijeme funkcija će provjeravati je li Bingo ostvaren. Funkcija prvo provjerava je li izvučeno svih 90 brojeva. Ukoliko jest, javlja poruku i prekida rad funkcije. Inače bira novi slučajni broj vodeći računa da taj broj nije već izvučen. Pozivom *config()* metode postavlja se vrijednost izvučenog broja i redni broj na pripadne naljepnice. Potom funkcija provjerava je li izvučeni broj jednak kojem broju u matrici naljepnica. Ako su jednaki tada se ta informacija bilježi u matricu provjere na odgovarajuće mjesto.

```
def noviBrojAkcija(event):
        if(len(listaIzvucenih) >= 90):
            messagebox.showinfo('Bingo', 'Izvučeni su svi brojevi')
           return
        else:
            zastavica = 0
            while(zastavica == 0):
                pom = randint(1, 90)
                if (pom not in listaIzvucenih):
                    listaIzvucenih.append(pom)
                    randomBroj = pom
                    zastavica = 1

            prikazBroja.config(text = randomBroj)
            brojIzvucenih.config(text = len(listaIzvucenih))
            for i in range(3):
                for j in range(5):
                    if(labeli[i][j].cget('text') ==
prikazBroja.cget('text')):
```

```
                    labeli[i][j].config(bg = 'red')
                    matricaProvjere[i][j] = 1
            provjeraBingo()
        return
```

Na kraju funkcije poziva se funkcija *provjeraBingo()*. Ova funkcija provjerava je li ostvaren dobitak. Zbrajaju se vrijednosti iz *matricaProvjere* i ukoliko je zbroj jedak 15 funkcija kreira poruke BINGO. Ovisno o tome do kojeg je broja Bingo izvučen, takva poruka se i ispisuje.

```
  def provjeraBingo():
        if (len(listaIzvucenih) <= 30):
            suma = 0
            for i in range(3):
                for j in range(5):
                    suma += matricaProvjere[i][j]
            if (suma == 15):
                messagebox.showinfo('Čestitamo', 'SUPERBINGO!')
        elif(len(listaIzvucenih) >= 31 and len(listaIzvucenih) <=
33):
            suma = 0
            for i in range(3):
                for j in range(5):
                    suma += matricaProvjere[i][j]
            if (suma == 15):
                messagebox.showinfo('Čestitamo', 'BINGO 33!')
        elif(len(listaIzvucenih) >= 34 and len(listaIzvucenih) <=
36):
            suma = 0
            for i in range(3):
                for j in range(5):
                    suma += matricaProvjere[i][j]
            if (suma == 15):
                messagebox.showinfo('Čestitamo', 'BINGO 36!')
        elif(len(listaIzvucenih) >= 37 and len(listaIzvucenih) <=
39):
            suma = 0
            for i in range(3):
```

```
            for j in range(5):
                suma += matricaProvjere[i][j]
        if (suma == 15):
            messagebox.showinfo('Čestitamo', 'BINGO 39!')
    else:
        suma = 0
        for i in range(3):
            for j in range(5):
                suma += matricaProvjere[i][j]
        if (suma == 15):
            messagebox.showinfo('Čestitamo', 'BINGO 40+!')
    return
```