Podloge za stručno usavršavanje učitelja osnovnih škola
za domenu
Računalno razmišljanje i programiranje

08_B
Primjeri vizualizacije u kornjačinu prozoru

Uz dozvolu izdavača korišteni su sadržaji iz priručnika:

Leo Budin, Predrag Brođanac, Zlatka Markučič
Smiljana Perić, Dejan Škvorc, Magdalena Babić

Računalno razmišljanje i programiranje u Pythonu
Element, Zagreb, 2017

Izrada slikovnog prikaza - vizualizacija

Tijekom stoljeća, kao i danas, mnogi prikazi i opisi i prirodnih i matematičkih spoznaja zasnivaju se na slikovnim prikazima koji su čovjeku vrlo bliski.

Kod rješavanja problema računalom vizualizacija nam služi za bolje razumijevanje problema, ali i za pronalaženje postupaka za njegovo rješavanje.

Pri izradi slikovnog prikaza u pravilu nastojimo da on što bolje predočuje zadani problem. Tu ne treba zanemariti i prisustvo svojevrsnog umjetničkog pristupa. Mnoge slike koje možemo stvoriti računalom mogu biti i umjetnička djela.

Brojevni pravac, vizualizacija zbrajanja i oduzimanja

Napišimo program s pomoću kojega ćemo na brojevnom pravcu vizualizirati zbrajanje i oduzimanje nenegativnih cijelih brojeva (prirodni brojevi i broj 0).

Zamislimo kako bi izgledalo rješenje ovog primjera. Očito se radi se o brojevnom pravcu s ishodištem 0. Za samu vizualizaciju problema bilo bi dobro da nam na pravcu bude označen određeni broj točaka. Ograničit ćemo se na cijele brojeve iz intervala [0, 20]. Program ćemo napraviti kao modul tako da se njegovim funkcijama možemo koristiti i za rješavanje drugih problema.

Prije no što krenemo s pisanjem programa važno je odrediti pojedine cjeline za koje ćemo u našem programu napraviti funkcije. Pogledajmo koje bi to funkcionalne cjeline bile:

- *crtanje okomite crtice na brojevnom pravcu*
- *brojevni pravac s oznakama*
- *crtanje točke koja označava vrijednost prvog operanda u računskoj operaciji i vrijednost rezultata*
- *crtanje polupravca s oznakama za zbrajanje*
- *crtanje polupravca s oznakama za oduzimanje*
- *glavna funkcija modula u kojoj ćemo odrediti prve vrijednosti operanda.*

```
#Brojevni pravac s prirodnim brojevima i nulom - program_8_14.py

from turtle import *

def crtica(vel):
    '''Crta crticu veličine 2 * vel na trenutačnom mjestu kornjače
       i to okomito na trenutačni smjer kornjače'''
    lt(90); fd(vel); bk(2 * vel); fd(vel); rt(90)

def brojevni_pravac(n=20, m=20):
    '''Crta brojevni pravac nenegativnih brojeva u intervalu
       [0, n] i jediničnom duljinom od m piksela
    '''
    tracer(False)
    reset(); pu(); bk(200); pd()                                    #1
    crtica(5)
    for i in range(n):                                              #2
        fd(m)
        crtica(3)
    fd(15); stamp()
    pu(); bk(n * m + 15)                                            #3
    rt(90); fd(25); lt(90)
    for i in range(n + 1):                                          #4
        write(str(i), align='center', font=('Aral',9,'bold'))
        fd(m)
    ht()
    tracer(True)
```

```
def točka(a, m=20):
    '''Postavlja točku na broj a brojevnog pravca'''
    pu()
    home()
    bk(200)
    fd(a * m)
    dot(8)                                                    #5

def zbroji(a, b, m=20):
    '''Postavlja: točku a, stelicu udesno s pribrojnikom b
       i točku na broj a + b brojevnog pravca
    '''
    točka(a)                                                  #6
    pu(); seth(90)
    fd(25); pd(); seth(0)
    crtica(15)
    fd(b * m); stamp()                                        #7
    pu(); bk(b * m // 2); lt(90); fd(5)
    write('+' + str(b), align='center', font=('Aral',9,'bold'))  #8
    točka(a + b)
```

```
def oduzmi(a, b, m=20):
    '''Postavlja: točku a, strelicu ulijevo s umanjiteljem b
       i točku na broj a - b brojevnog pravca
    '''
    točka(a)
    pu(); seth(90)
    fd(25); pd(); seth(180)
    crtica(15)
    fd(b * m); stamp()
    pu(); bk(b * m // 2); rt(90); fd(5)
    write('-' + str(b), align='center', font=('Arial',9,'bold'))
    točka(a - b)

if __name__ == '__main__':
    a = 7
    b = 5
    brojevni_pravac()
    zbroji(a, b)
    nastavak = input('Pritisnuti tipku (Unos) za nastavak! ')          #9
    brojevni_pravac()                                                  #10
    oduzmi(a, b)
```

Naredbama `tracer(False)` i `tracer(True)` na početku i na kraju definicije funkcije ubrzavamo iscrtavanje brojevnog pravca u grafičkom ekranu. Nakon što nizom naredbi (#1) postavimo početne parametre prozora, kornjaču ćemo pomaknuti ulijevo sa `bk(200)` te ćemo nacrtati početnu okomitu crticu brojevnog pravca. Početna će crtica biti nešto dulja od ostalih crtica tj. oznaka na brojevnom pravcu. Izvođenjem petlje `for` (#2) nacrtat ćemo `n` jediničnih duljina na čijem će kraju biti okomita crtica. Kako bismo nacrtali uobičajenu strelicu na kraju brojevne crte, napisali smo dvije naredbe od kojih će prva pomaknuti kornjaču za 15 piksela, a druga otisnuti sliku kornjače tj. strelice. Preostaje nam još ispisati brojeve ispod svake crtice. Nakon pomicanja na početak brojevnog pravca (#3) i postavljanjem kornjače za 15 piksela niže od pravca te usmjeravanjem kornjače udesno, možemo početi pisati brojeve. Brojeve ćemo ispisati izvođenjem petlje (#4) u kojoj se koristimo funkcijom `write()`.

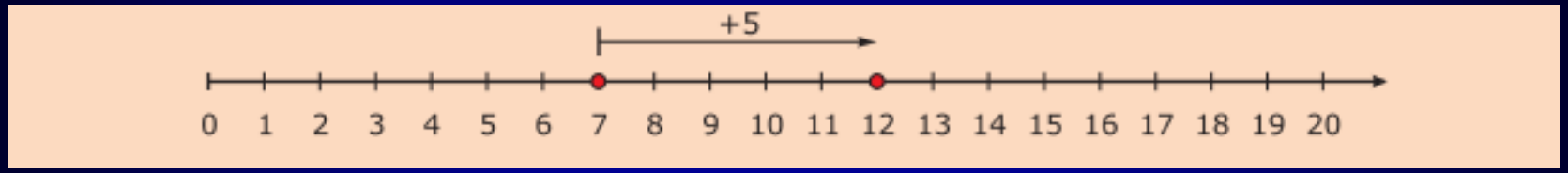
U funkciji `točka()` prvo ćemo funkcijom `home()` postaviti kornjaču u početni položaj, prisjetimo se da ta funkcija za razliku od funkcije `reset()` ne briše prethodno izrađen crtež. Zatim ćemo ju premjestiti na početak brojevnog pravca. Nakon toga ćemo naredbom `fd(a * m)` kornjaču pomicati do odgovarajućih crtica na brojevnom pravcu i nacrtati točku (#5).

U funkciji `zbroji()` prvo crtamo točku na poziciji koja odgovara prvom pribrojniku (#6). Nakon toga ćemo kornjaču pomaknut za 25 piksela iznad pravca s tim da će ostati usmjerena udesno. Obratite pažnju da smo se za to koristili naredbama `seth(90)` i `seth(0)`. To smo mogli postići i naredbama `lt(90)` i `rt(90)`. Pozivom funkcije `crtica()` nacrtat ćemo crticu i nakon toga naredbama u redu (#7) polupravac sa strelicom čija duljina odgovara vrijednosti pribrojnika `b`. Nakon što postavimo kornjaču na sredinu strelice, ispisat ćemo znak zbrajanja `'+'` i vrijednost drugog pribrojnika (#8). Konačno, nacrtat ćemo točku na mjestu koje odgovara zbroju `a + b`.

Funkcija `oduzmi()` vrlo se malo razlikuje od funkcije `zbroji()`. U njoj ćemo strelicu crtati u suprotnom smjeru te ćemo nad strelicom ispisivati znak oduzimanja `'-'` te vrijednost umanjitelja `b`. Na kraju ćemo nacrtati točku na mjestu koje odgovara razlici `a - b`.

U glavnom programu najprije se zadaju vrijednosti `a` i `b`.

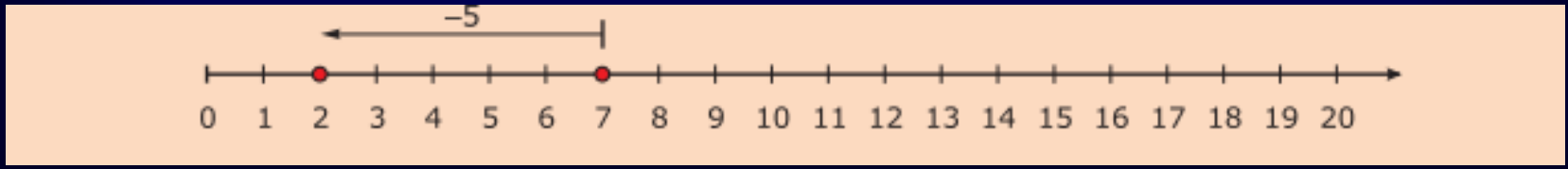
Nakon toga se pozivom funkcija `brojevni_pravac()` i `zbroji(a,b)` dobiva sljedeći crtež:



Kako bismo prije nastavka izvođenja crtež mogli u miru promotriti dodali smo naradbu `#9`. U interaktivnom sučelju pojavit će se ispis:

```
>>>
Pritisnuti tipku (Unos) za nastavak!
```

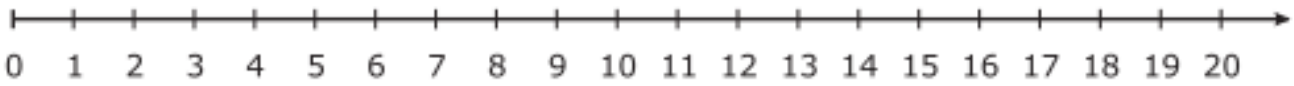
Pritiskom na tipku <unos> nstavlja se izvoditi narerdba #10 i dobivamo sljedeću sliku:



Nakon što je program pokrenut, njegove funkcije možemo pozivati i iz interaktivnog sučelja, tako da ćemo nakon poziva

```
>>> brojevni_pravac()
>>>
```

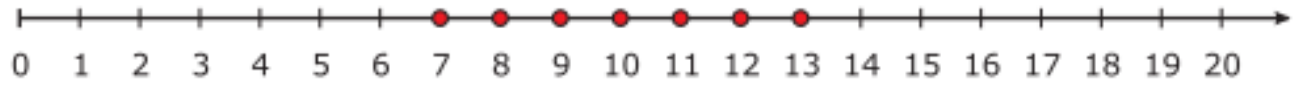
dobiti sliku brojevnog pravca bez označenih točaka



Osim vizualizacije zbrajanja i oduzimanja, funkcije iz ovog programa omogućuju nam slikovni prikaz otvorenih i zatvorenih intervala brojeva.

Tako ćemo, primjerice, cjelobrojne točke iz intervala [7, 14> izvođenjem sljedećih naredbi moći prikazati kao na slici

```
>>> brojevni_pravac()
>>> for i in range(21):
        if i >= 7 and i < 14:
            točka(i)
```



Vizualizacija zakona komutacije za zbrajanje

U prezentaciji 03 smo naučili kako se crtaju dužine i dužine koje nastaju nadovezivanjem dužina i definirali smo funkcije `crtica()`, `jedinična_dužina()` i `dužina()`.

Nadopunit ćemo taj program tako da zamjenom mjesta pribrojnika dobivamo isti zbroj.

Kao što smo već uvodno napomenuli, pri rješavanju ovog zadatka koristit ćemo se već poznatim funkcijama `crtica()` i `jedinična_dužina()` dok ćemo funkciju `dužina()` za potrebe ovog primjera nadopuniti tako da na sredini dužine ispisuje i njezinu duljinu. I već poznate funkcije ćemo za potrebe ovog programa ponovno napisati.

Osim ovih funkcija, trebat ćemo napisati i funkciju koja će nam kornjaču postavljati u početnu točku te funkciju koja će crtati dužine u različitom redoslijedu.

```
#Svojstvo komutativnosti - program_8_15.py

from turtle import *

def crtica(vel):
    '''Funkcija crta okomite crtice na mjestu na kojem
       se trenutačno nalazi. Crtica će bit dugačka 2 * vel piksela.
       Preporuka: za kraće crtice odabrati vel = 3,
                  za dulje crtice odabrati vel = 8.
    '''
    lt(90)
    fd(vel); bk(2 * vel); fd(vel)
    rt(90)

def jedinična_dužina(m=20):
    '''Funkcija crta jediničnu dužinu duljine m piksela.
       Na njezinu će se kraju nacrtati crtica.
    '''
    fd(m)
    crtica(3)

def dužina(dulj, m=20):
    '''Funkcija crta dužinu koja se sastoji od dulj jediničnih dužina.
          dulj - broj jediničnih dužina koje čine dužinu
          m - broj piksela jedinične dužine
    '''
    crtica(8)
    pu(); fd(dulj * m // 2); rt(90); fd(20)
    write(dulj, align='center', font=('Aral',9,'bold'))
    bk(20); lt(90); bk(dulj * m // 2); pd()
    for i in range(dulj):
        jedinična_dužina(m)
    crtica(8)
```

```
def početna_točka(pol):
    '''Početna točka crtanja postavlja se
       na položaj (-200, pol)
    '''
    pu(); seth(0); goto(-200, pol); pd()

def komutativnost_zbrajanja(a, b):
    '''Funkcija crta nadovezane dužine a i b
       te zatim dužinu a + b. Nakon toga crta
       nadovezane dužine b i a te zatim dužinu b + a
    '''
    reset(); pu(); ht()
    početna_točka(90); dužina(a); dužina(b)                    #1
    početna_točka(60); dužina(a + b)
    početna_točka(-60); dužina(b); dužina(a)                   #2
    početna_točka(-90); dužina(b + a)

if __name__ == '__main__':
    while True:
        a = input('Duljina dužine a (a <= 15)(za kraj (Unos)): ')
        if not a:
            break
        a = int(a)
        b = int(input('Duljina dužine b (b <= 15): '))
        komutativnost_zbrajanja(a, b)
```

Opišimo djelovanje novih funkcija. Funkcija `početna_točka()` postaviti će kornjaču u početni položaj pojedinih dužina tj. u početne točke `(-200, pol)`. Sve se početne točke nalaze 200 piksela lijevo od središnjeg položaja, a parametar `pol` određuje njihov okomiti odmak od središnjeg položaja.

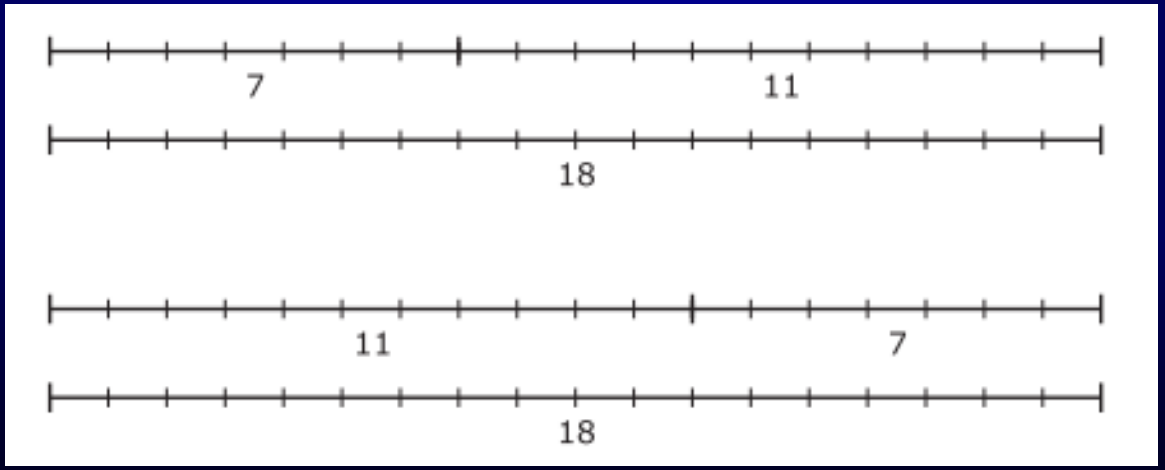
U funkciji `komutativnost_zbrajanja()` prvo će se nacrtati dužina duljine *a*, a zatim će se na nju nadovezati dužina duljine *b* (**#1**). Nakon toga će se ispod nacrtanih nadovezanih dužina nacrtati dužina čija je duljina jednaka zbroju dvaju pribrojnika.

Jednaki postupak će se izvesti ako prvo nacrtamo dužinu duljine drugog pribrojnika te nadovežemo dužinu duljine prvog pribrojnika (**#2**) i potom ispod njih dužinu koja predstavlja dužinu duljine zbroja drugog i prvog pribrojnika.

Glavnim ćemo programom omogućiti ponavljanja crtanja uz zadavanje različitih duljina. Program možemo završiti tako da umjesto upisivanja vrijednosti duljine pritisnemo samo tipku *(Unos)*.

Pokretanjem programa u interaktivnom sučelju treba utipkati duljine dužina što može izgledati ovako:

```
Duljina dužine a (a <= 15)(za kraj (Unos)): 7
Duljina dužine b (b <= 15): 11
Duljina dužine a (a <= 15)(za kraj (Unos)):
>>>
```

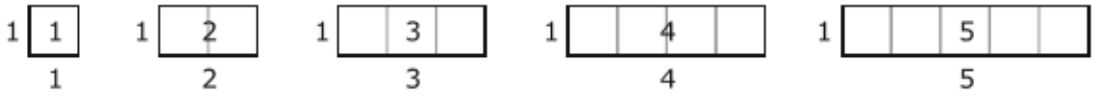


Vizualizacija množenja

Prisjetimo se da smo zbrajanje prirodnih brojeva prikazali kako nadovezivanje dužina čije su nam duljine predstavljale brojeve. Pritom smo se morali odlučiti koliko je velik grafički prikaz jedinične dužine. U prethodnim odjeljcima smo za naše grafičke prikaze odabrali da jedinica mjerenja iznosi 20 piksela, ali smo mogli odabrati i neki drugi broj piksela.

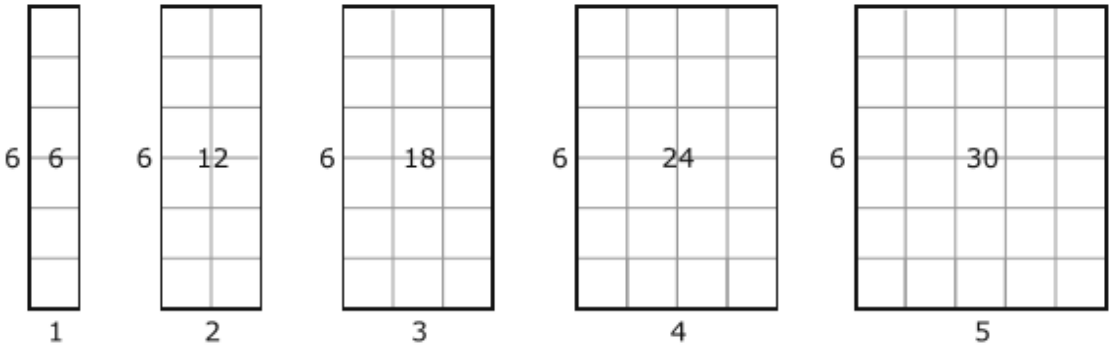
Množenje možemo vizualizirati tako da umnožak promatramo kao površinu pravokutnika. Kao što smo kod mjerenja dužine odabrali jediničnu dužinu, tako ćemo i ovdje kao jediničnu površinu odabrati površinu kvadrata čije stranice imaju duljinu *m* piksela. Na slici 8.35. možemo uočiti da produživanjem jedne stranice jediničnog kvadrata s faktorima 2, 3, 4 i 5 dobivamo pravokutnike čija će se površina povećavati za odgovarajući faktor.

Sl. 8.35.



Ako produživanje primijenimo i u okomitom smjeru, dobiveni pravokutnici će imati površine jednake umnošku faktora produživanja. Na slici 8.36. su svi pravokutnici sa slike 8.35. s produženim okomitim stranicama i to za jednaki faktor 6.

Sl. 8.36.



```
#Vizualizacija operacije množenja - program_mnozenje.py

from turtle import *

def mreža_jediničnih_kvadrata(a, b, xs, ys, m=30, boja='Silver'):
    '''Funkcija crta pravokutnik sastavljen od jediničnih kvadrata.
         a, b - stranice pravokutnika
         xs, ys - središte jediničnog kvadrata u donjem
                  lijevom kutu pravokutnika
         m - broj piksela jedinične dužine (postavljeno m=30)
         boja - boja olovke (postavljeno boja='Silver9)
    '''
    ht()
    pencolor(boja)
    for i in range(b):
        pu()
        goto(xs - m // 2, ys - m // 2 + i  * m)                    #1
        pd(); seth(0)
        for j in range(a):                                         #2
            for k in range(4):
                fd(m)
                lt(90)
            fd(m)
```

```
def umnožak(a, b, xs, ys, m=30, mreža=1):
    '''Funkcija crta pravokutnik
         a, b - stranice kvadrata
         xs, ys - središte jediničnog kvadrata u donjem
                  lijevom kutu pravokutnika
         m - broj piksela jedinične dužine (postavljeno m=30)
         mreža - određuje crtanje mreže jediničnih kvadrata
                 mreža = 1 (postavljeno) mreža se crta
                 mreža = 0 mreža se ne crta
    '''
    tracer(False)
    if mreža:
        mreža_jediničnih_kvadrata(a, b, xs, ys)
    pu(); ht()
    pencolor('Black')
    goto(xs + (a - 1) * m // 2, ys - 10 + (b - 1) * m // 2);                #3
    write(a * b, align='center',font=('Arial',10,'bold'))                   #4
    goto(xs + (a - 1) * m // 2, ys -  m // 2 - 18)                          #5
    write(a, align='center',font=('Arial',9,'bold'))                        #6
    goto(xs - m // 2 - 10, ys - 10 + (b - 1) * m // 2)                      #7
    write(b, align='center',font=('Arial',9,'bold'))                        #8
    goto(xs - m // 2, ys - m // 2)
    pd()
    for i in range(2):
        fd(a * m)
        lt(90)
        fd(b * m)
        lt(90)
    tracer(True)
```

```
if __name__ == '__main__':
    reset()
    umnožak(1, 1, -250, 0)
    umnožak(2, 1, -170, 0)
    umnožak(3, 1, -60, 0)
    umnožak(4, 1, 80, 0)
    umnožak(5, 1, 250, 0)
```

U dokumentacijskom stringu u definiciji funkcije `mreža_jediničnih_kvadrata()` opisane su uloge svih njenih parametara. Ta će funkcija sa zadanom nijansom sive boje `'Silver'`, koju naravno možemo pri pozivu promijeniti, nacrtati jedan ispod drugoga `b` redaka nizova od po `a` jediničnih kvadrata. Duljina jediničnih kvadrata je postavljena na 30 piksela. Uočimo da je `(xs, ys)` središte jediničnog kvadrata u donjem lijevom vrhu pravokutnika. Petlja (#2) crta kvadrate iz njegova lijevog donjeg vrha dok naredba (#1) određuje početnu točku crtanja prvog jediničnog kvadrata u svakom redu.

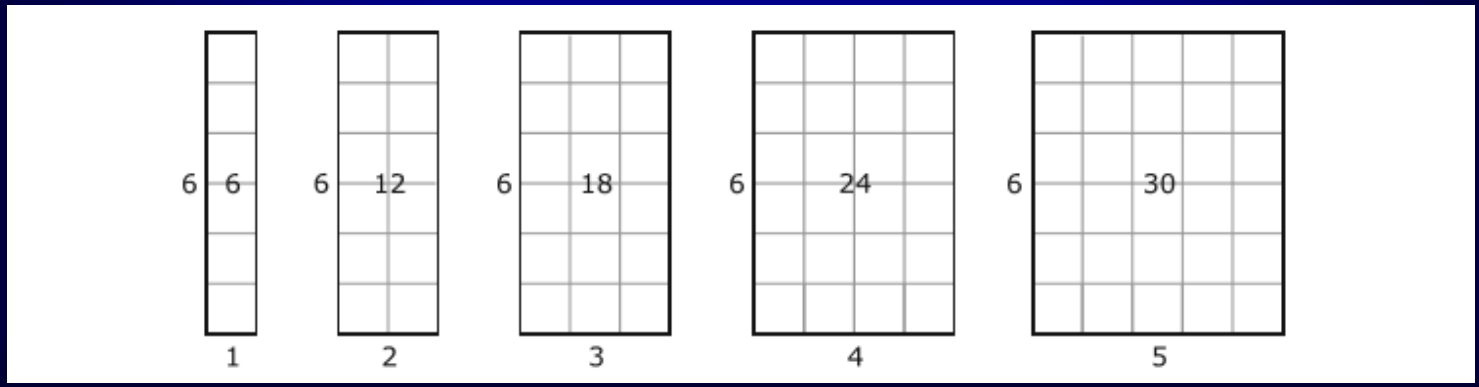
Funkcijom `umnožak()` nacrtat ćemo pravokutnik sa stranicama `a` i `b`. Ako se parametar mreža ne mijenja, ostat će mu postavljena vrijednost jednaka `1` i nacrtat ćemo i mrežu jediničnih kvadrata. Naredbe (#3) i (#4) će u sredini pravokutnika ispisati vrijednost umnoška. Naredbama (#5) i (#6) ćemo ispod vodoravne stranice ispisati vrijednost parametra `a`, dok ćemo izvođenjem naredbi (#7) i (#8) lijevo od uspravne stranice ispisati vrijednost parametra `b`. Obratimo pažnju na to kako smo naredbama (#3), (#5) i (#7) kornjaču doveli u točke na kojima će funkcija `write()` ispisivati odgovarajuće vrijednosti.

Slijedi crtanje pravokutnika unutar petlje `for`. Pravokutnik će biti obrubljen crnom bojom.

U glavnom se programu funkcija `umnožak()` poziva pet puta te će se nacrtati crtež kao na slici 8.35. Učimo da je u svim tim pozivima parametar `b` jednak `1`.

Izvođenjem programa `program_mnozenje_2.py` u kojem svi pozivi funkcije `umnožak()` imaju argument `b = 6` dobit ćemo crtež kao na slici 8.36.

```
# Vizualizacija operacije množenja - program_mnozenje_2.py

from program_mnozenje import *

def main():
    reset()
    umnožak(1, 6, -250, 0)
    umnožak(2, 6, -170, 0)
    umnožak(3, 6, -60, 0)
    umnožak(4, 6, 80, 0)
    umnožak(5, 6, 250, 0)

main()
```



Vizualizacija komutativnosti množenja

Napišimo program kojim ćemo crtajući jedinične kvadrate pokazati da vrijedi zakon komutacije za množenje, odnosno da će umnožak ostati jednak ako prilikom množenja zamijenimo faktore, tj. da vrijedi izraz $a \cdot b = b \cdot a$.

To svojstvo množenja možemo vizualizirati tako da u interaktivnom sučelju importiramo modul `program_mnozenje` *te nakon naredbe* `reset()` *pozovemo funkciju* `umnožak()` *s odgovarajućim parametrima.*

Primijetimo da ćemo kada utipkamo naziv funkcije `umnožak` te otvorimo zagradu, vidjeti dokumentacijski string koji smo napisali u definiciji .

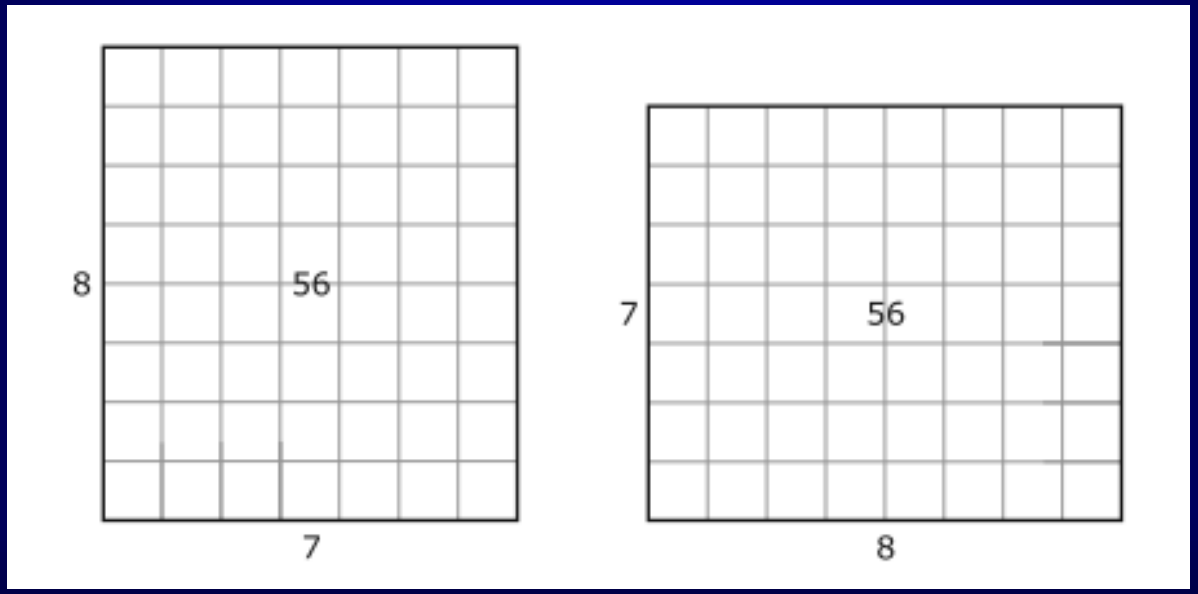
```
>>> reset()
>>> umnožak(
        (a, b, xs, ys, m=30, mreža=1)
        Funkcija crta pravokutnik
        a, b - stranica kvadrata
        xs, ys - središte jediničnog kvadrata u donjem
        lijevom vrhu pravokutnika
        m - broj piksela jedinične dužine (postavljeno m=30)
```

Dokumentacijski string je malo prevelik pa tako u ovom opisu nedostaje opis parametra `mreža=1`.

U interaktivnom sučelju sada možemo, primjerice pisati:

```
>>> reset()
>>> umnožak(7, 8, -250, 0)
>>> umnožak(8, 7, 0, 0)
```

i dobit ćemo crtež kao na slici 8.38. iz kojeg nam je jasno vidljivo da su dvije površine jednake i da prema tome svojstvo komutacije vrijedi.

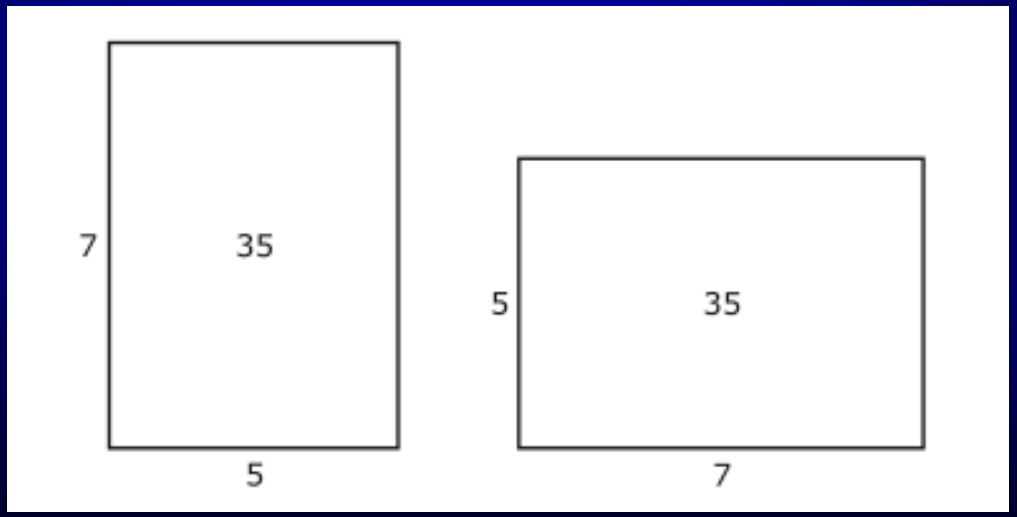


Želimo li pravokutnike čije su površine jednake umnošku duljina njihovih stranica crtati bez mreže, koja prikazuje sve jedinične kvadrate, možemo to načiniti tako da postavljeni parametar `mreža=1` promijenimo u `mreža=0`.

Tako ćemo sljedećim naredbama u interaktivnom sučelju:

```
>>> reset()
>>> ht ()
>>> umnožak(5, 7, -200, 0, mreža=0)
>>> umnožak(7, 5, 0, 0, mreža=0)
```

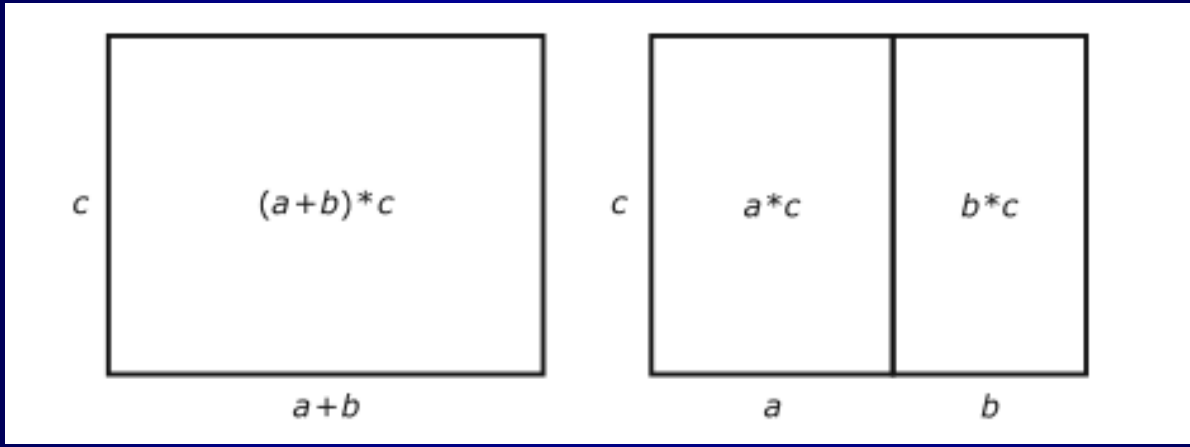
u kornjačinu prozoru dobiti crtež



Vizualizacija distributivnosti množenja prema zbrajanju

Distributivnost množenja prema zbrajanju je svojstvo koje nam pomaže u provođenju aritmetičkih operacija. Naime, za tri broja *a*, *b* i *c* uvijek vrijedi da je $(a + b) \cdot c = a \cdot c + b \cdot c$.

Vizualizacijom se možemo uvjeriti da je površina pravokutnika čije su stranice $a + b$ i c, jednaka zbroju površina dvaju pravokutnika: jednog sa stranicama *a* i *c* te drugog sa stranicama *b* i *c*.



Napišimo program kojim ćemo vizualizirati distributivnost množenja prema zbrajanju za konkretne vrijednosti varijabli `a`, `b` *i* `c`.

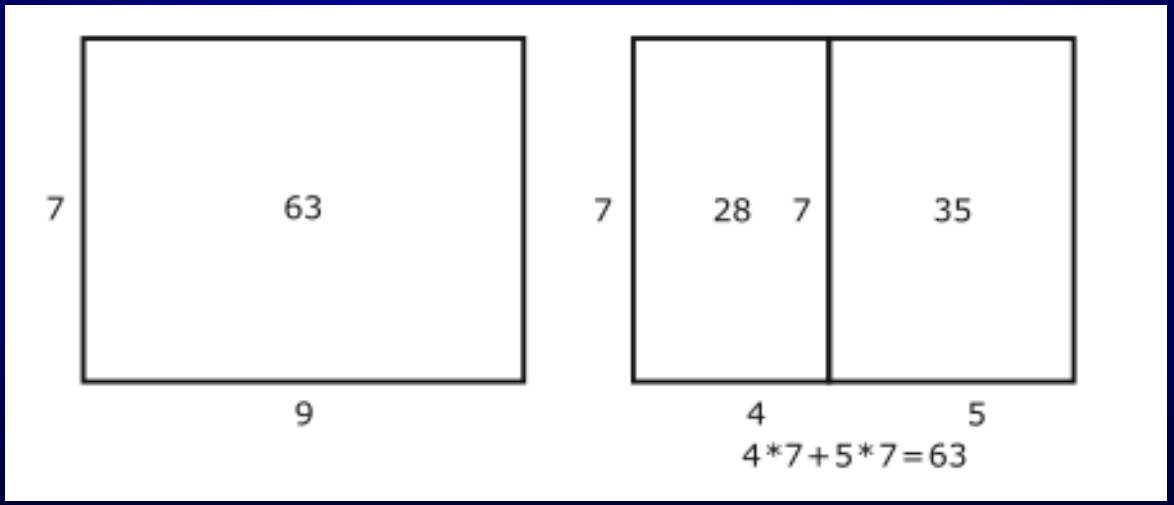
Za rješavanje ovog problema iskoristit ćemo funkciju `umnožak()` *iz programa* `program_mnozenje.py`. *Pogledajmo kako bi program mogao izgledati:*

```
#Distributivnost množenja prema zbrajanju - program_8_18.py

from program_mnozenje import *

def distributivnost(a, b, c):
    '''Funkcija prikazuje svojstvo distributivnosti množenja
       prema zbrajanju.
       Zbog jasnoće crteža treba biti a + b < 10
       te a >= 3 i b >= 3
    '''
    reset(); ht()
    jed = 20
    umnožak(a + b, c, -250, 0, m=jed, mreža=0)
    pu(); goto((a + b) * jed // 2 - 250, -50)                          #1
    ispis = '{0} * {1} = {2}'.format(a + b, c, (a + b) * c)
    write(ispis, align='center', font=('Arial',9,'bold'))              #2
    umnožak(a, c, 0, 0, m=jed, mreža=0)
    umnožak(b, c, a * jed, 0, m=jed, mreža=0)                          #3
    pu(); goto((a + b) * jed // 2, -50)                                #4
    ispis = '{0} * {2} + {1} * {2} = {3}'.format(a, b, c, a*c + b*c)
    write(ispis, align='center', font=('Arial',9,'bold'))              #5

if __name__ == '__main__':
    print('Zbog jasnoće crteža treba biti a + b < 10\n'\
          'te a >=3 i b >= 3')
    a = int(input('Upiši a: '))
    b = int(input('Upiši b: '))
    c = int(input('Upiši c: '))
    distributivnost(a, b, c)
```

U definiciji funkcije `umnožak()` ostavili smo mogućnost promjene parametra `m` koji određuje veličinu dužinske jedinice. U ovom programu odabrali smo da to bude 20 piksela. U pozivima funkcije `umnožak()` smo s `mreža=0` isključili crtanje jediničnih kvadrata. Važno je još uočiti da je u naredbi (#3) početna točka sa `a * jed` odabrana tako da se pravokutnici sa stranicama `a` i `b` nadovezuju.

Naredbom (#1) postavili smo kornjaču na mjesto gdje ćemo ispisati način izračuna cijele površine (#2), a naredbom (#4) smo pripremili položaj kornjače za ispis zbroja površine kao zbroja površina pojedinačnih pravokutnika (#5).

Uz upisane vrijednosti `a = 4`, `b = 5` i `c = 7` dobit ćemo crtež prikazan slikom 8.41.



Optičke iluzije

Ljudi su kroz stoljeća i tisućljeća otkrivali matematičke metode i rabili matematičke rezultate prije nego su oni teorijski dokazani. Kroz povijest su se matematičari oslanjali i na vizualne prikaze od kojih smo neke upoznali u ovom poglavlju.

Međutim, ljudski vizualno-misaoni sustav nije savršen. On se tijekom evolucije razvio kako bi omogućio čovjeku preživljavanje u prirodi. Zbog urođenih svojstava prepoznavanja raznih geometrijskih oblika neki slikovni prikazi (tzv. optičke iluzije) mogu nas navesti na krive zaključke. Zbog toga se u današnjoj matematici slike ne smatraju dokazima matematičkih istina iako one mogu pomoći u razumijevanju problema i pronalaženju načina za njegovo rješavanje.

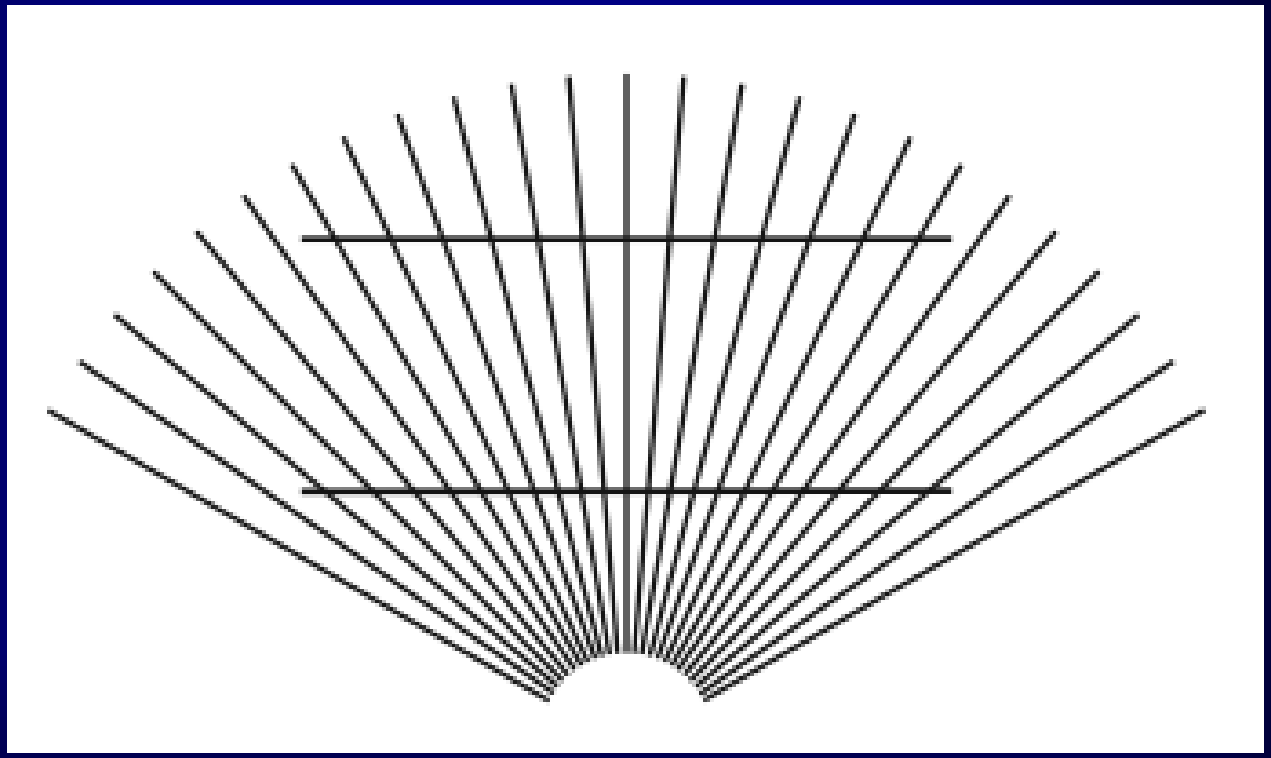
Tako u dosadašnjim primjerima vizualizacije nije bilo optičkih iluzija.

No, pogledat ćemo kako se neke od poznatih optičkih iluzija mogu ostvariti jednostavnim programima u *Pythonu*.

Dvije dužine na snopu pravaca

```
#Optička iluzija: dvije dužine na snopu polupravaca - program_8_19.py

from turtle import *

def snop():
    lt(30)
    for i in range(25):
        pu(); fd(40)
        pd(); fd(250); bk(250)
        pu(); bk(40)
        lt(5)

def dvije_dužine():
    seth(0)
    pu(); goto(-140, 110)
    pd(); fd(280)
    pu(); goto(-140, 220)
    pd(); fd(280)

def main():
    reset(); ht(); pensize(3)
    tracer(False)
    snop()
    dvije_dužine()
    tracer(True)

main()
```

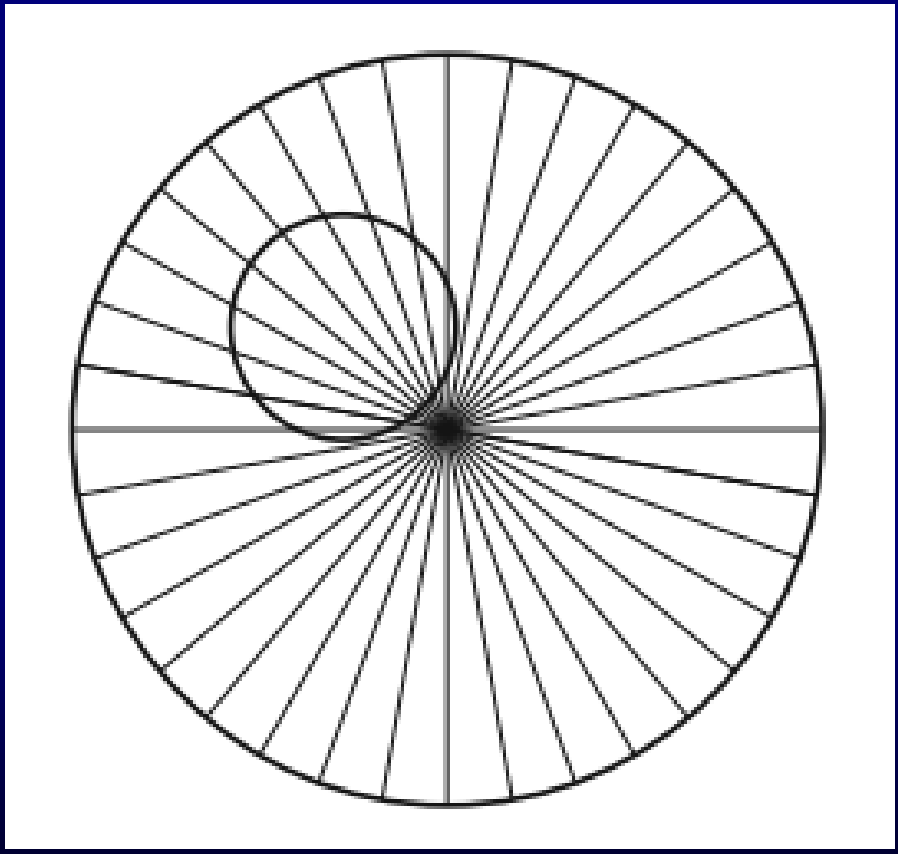
S obzirom na to da se polupravci snopa šire, dužina na dijelu snopa gdje su polupravci razmaknutiji prividno je kraća.



Privid izobličene kružnice

```
#Optička iluzija: privid izobličene kružnice - program_8_20.py

from turtle import *
from program_8_5 import kružnica

def promjeri(r, xs, ys, n):
    '''Funkcija crta n promjera kružnice polumjera r
       sa središtem u (xs,ys)
    '''
    pu(); goto(xs, ys); seth(0); pd()
    for i in range(n):
        fd(r); bk(2 * r); fd(r)
        lt(180 // n)

def main():
    ht(); pensize(1)
    tracer(False)
    kružnica(300, 0, 0)                                    #1
    kružnica(90, -70, 80)                                  #2
    tracer(True)
    input('Pritskom na tipku (Unos) ucrtati promjere')
    tracer(False)
    promjeri(300, 0, 0, 18)
    tracer(True)

    main()
```

Za izradu ovog programa iskoristit ćemo funkciju `kružnica()` iz programa `program_8_5.py` koji smo importirali. Program će prvo naredbama (`#1`) i (`#2`) nacrtati dvije kružnice: jednu s polumjerom 300 piksela i drugu s polumjerom 90 piksela. Nakon što pritisnemo tipku *(Unos)*, napomenimo da pritom mora bit aktivan prozor interaktivnog sučelja, iscrtat će se polumjeri veće kružnice i dobit ćemo traženu sliku na kojoj je manja kružnica prividno izobličena.

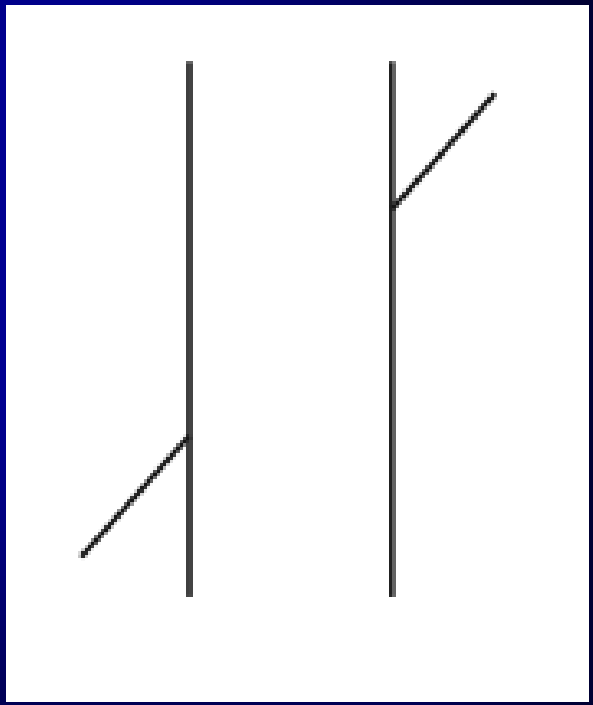


Presjecanje dvaju paralelnih pravaca

```
#Optička iluzija: pravac preko paralelnih pravaca - program_8_21.py

from turtle import *

def paralelni_pravci():
    pu(); goto(-50, -150)
    pd(); goto(-50, 150)
    pu(); goto(50, -150)
    pd(); goto(50, 150)

def prekinuti_pravac():
    pu(); goto(-100, -100);
    pd(); goto(-50, -50)
    pu(); goto(50, 50)
    pd(); goto(100, 100)
def spojnica():
    pu(); goto(-50, -50)
    pd(); goto(50, 50)

def main():
    reset(); ht(); pensize(4)
    paralelni_pravci()
    prekinuti_pravac()
    nastavak = input('Pritskom na tipku (Unos) nacrtati spojnicu! ')
    spojnica()

main()
```

Kada se dva paralelna pravca presjeku trećim i to tako da se njegov dio između paralelnih pravaca ne vidi, nastaje optička iluzija u kojoj su vidljivi dijelovi trećeg pravca prividno posmaknuti. Program koji će nam predočiti tu iluziju izgleda ovako:

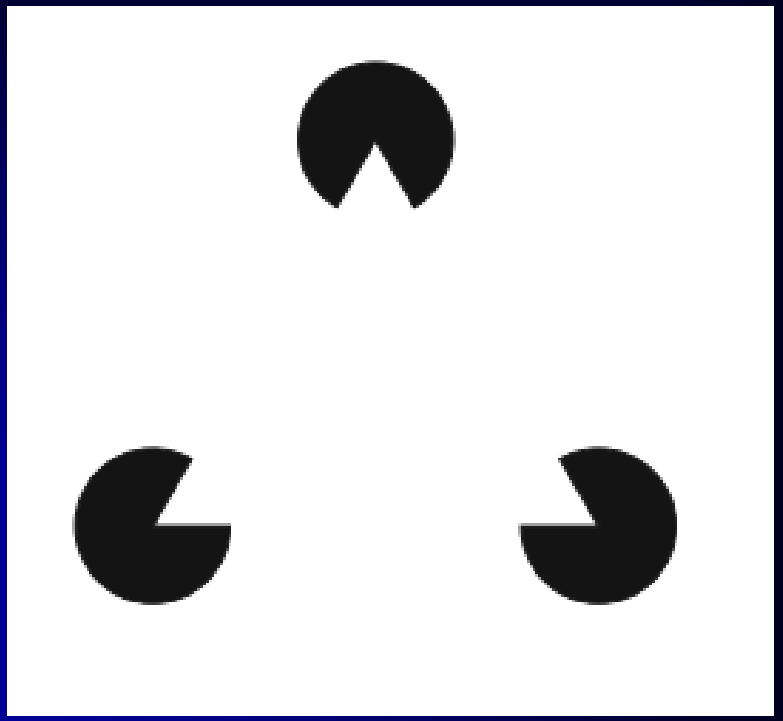
Sve su funkcije jednostavne i razumljive tako da nam ni ovdje nije potreban komentar. Izvođenjem programa dobit ćemo sliku 8.44.



Nakon što pritisnemo tipku *(Unos)* ucrtat će se spojnica i uvjerit ćemo se da je posmak dijelova pravca samo privid.

Prividni trokut

Pogledajmo još i poznatu optičku iluziju u kojoj se prepoznaje oblik trokuta iako trokut nije stvarno nacrtan već su samo naznačeni njegovi vrhovi i to kao kružni isječci triju krugova. Taj je iluzija prikazana slikom 8.45.



Pripremit ćemo program u kojem ćemo rabiti funkciju za crtanje kružnih isječaka što smo definirali u programu `program_8_6.py`.

Pri definiciji te funkcije napisali smo opširni dokumentacijski string pa se sada lako možemo prisjetiti što ta funkcija radi i koji su joj parametri,

```
>>> reset()
>>> kružni_isječak(
        (r, xs, ys, kut_početni, kut_luka)
Funkcija crta kružni isječak gdje su:
r - polumjer kružnice
xs, ys - središte kružnice
kut_početni - početni kut isječka
kut_luka - središnji kut isječka
```

```
#Optička iluzija: prividni trokut - program_8_22.py

from turtle import *
from program_8_6 import kružni_isječak

def prividni_trokut():
    pu(); home()                                                    #1
    lt(90); fd(150);
    xs, ys = position()
    begin_fill()
    kružni_isječak(50, xs, ys, 300, 300)
    end_fill()
    pu(); home()                                                    #2
    lt(210); fd(150)
    xs, ys = position()
    begin_fill()
    kružni_isječak(50, xs, ys, 60, 300)
    end_fill()
    pu(); home();                                                   #3
    lt(330); fd(150)
    xs, ys = position()
    begin_fill()
    kružni_isječak(50, xs, ys, 180, 300)
    end_fill()

def main():
    reset(); ht()
    prividni_trokut()

main()
```

U funkciji `prividni_trokut()` triput se ponavlja jednaki niz naredbi (#1), (#2) i (#3) no s drugim vrijednostima parametara. Prvi vrh određen je tako da kornjaču iz njenog početnog položaja usmjerimo prema gore i pomaknemo ju za 150 piksela.

Funkcija `positon()` očitat će taj položaj i to će biti središte oko kojeg se crta kružni isječak. Kružni isječak ima 300° te mu je početni kut 300°. Nacrtani isječak ćemo obojiti u crno.

Na jednaki ćemo način nacrtati i ostale kružne isječke, ali na drugim pozicijama. Za crtanje drugoga isječka kornjaču ćemo usmjeriti prvo na 210° (tj. 90° + 120°), a za crtanje trećega na 330° (tj. 90° + 2×120°). Za svaki od vrhova treba još odrediti početni kut isječka, dok je središnji kut luka za sve isječke jednak i iznosi 300°.

Ovo razmatranje upućuje na to da funkciju `prividni_trokut()` možemo napisati i kraće na sljedeći način:

```
def prividni_trokut():
    for i in range(3):
        pu(); home()
        lt(90 + i * 120); fd(150)
        xs, ys = position()
        begin_fill()
        kružni_isječak(50, xs, ys, (300 + i * 120) % 360, 300)
        end_fill()
```