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

09_A
Racionalni brojevi u Pythonu, modul `fractions`

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

Definiranje razlomaka u modulu `fractions`

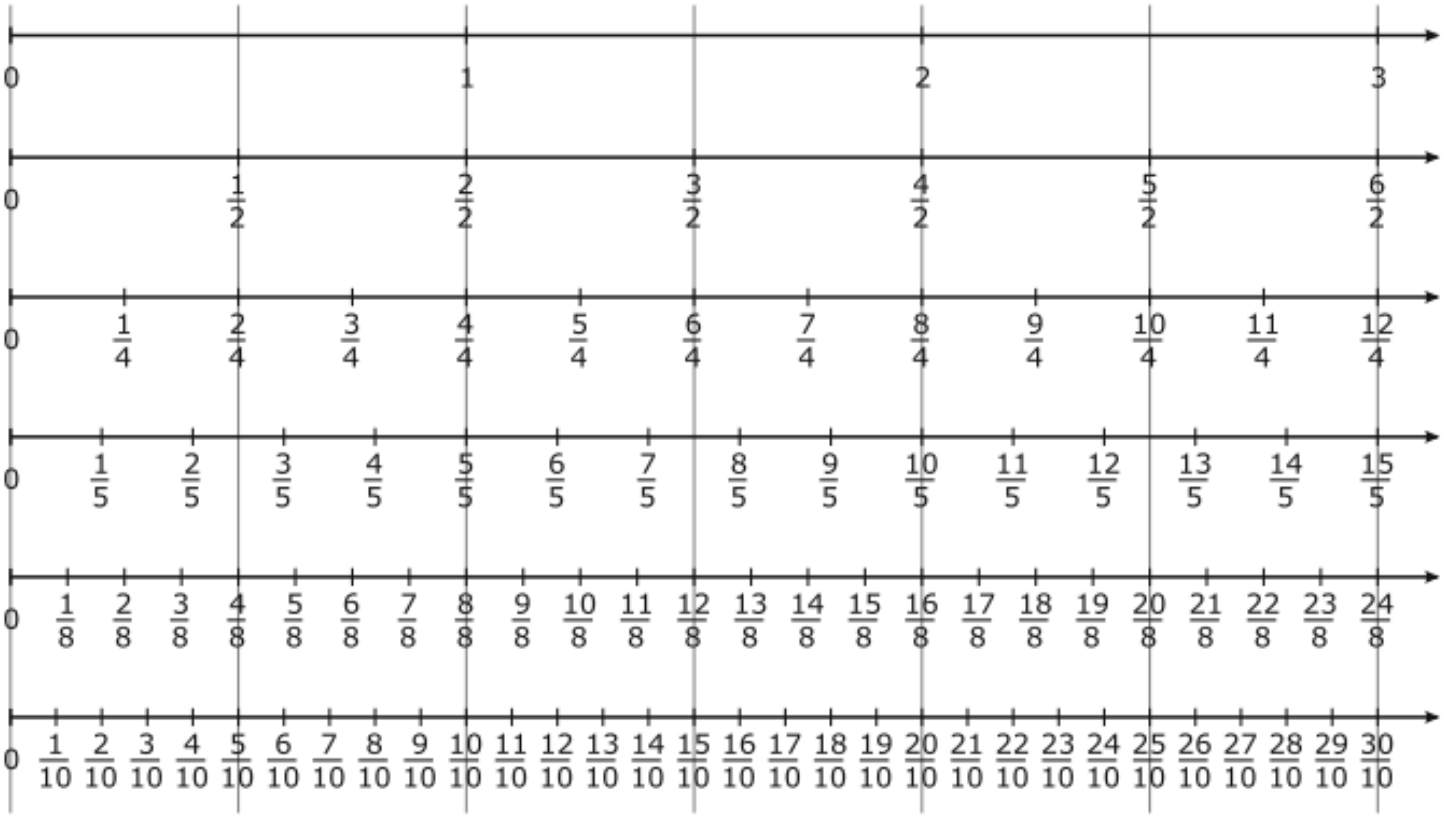
U matematici smo upoznali razlomke i aritmetičke operacije s razlomcima. Prisjetimo se da se s pomoću razlomka može prikazati dio jedinice ili nekoliko jednakih dijelova jedinice. U *Pythonu* će nam prikaz razlomaka i operacije s razlomcima omogućiti modul `fractions`.

Prilikom pisanja razlomaka u *Pythonu* koristit ćemo se oblikom zapisa s ukošenom razlomačkom crtom. Tako ćemo razlomak $\frac{3}{4}$ zapisivati 3/4.

U mjerenju udaljenosti možemo odabrati neku jedinicu mjere te s pomoću nje izmjeriti udaljenost od početnog položaja (koji smo označili sa 0). Na taj način nastaje brojevni pravac na kojem smo dosad prikazivali samo prirodne brojeve i broj 0.

Želimo li naš položaj izmjeriti preciznije, morali bismo odabrati manju jedinicu mjere ili možemo odabranu jedinicu mjere podijeliti na manje jednake dijelove. Te manje dijelove možemo prikazati na brojevnom pravcu. Promotrimo to na slici 9.1.

Sl. 9.1.



Na vrhu slike prikazan je pravac s trima jedinicama mjerenja. Na njemu su označeni brojevi 0, 1, 2 i 3.

Na sljedećem prikazu taj je isti pravac podijeljen na odsječke jednake polovini jedinične dužine (jedinice). Polovine počinjemo mjeriti od početne točke pa vidimo da smo do točke 1 došli tako da smo odmjerili dvije polovine i zatim smo do točaka 2 i 3 odmjerili 4, odnosno 6 polovina. Na sljedećem je pravcu mjerna jedinica podijeljena na četvrtine i na njemu vidimo da dvije četvrtine čine jednu polovinu i četiri četvrtine jednu cijelu jedinicu.

Na sljedeća tri prikaza pravca jedinica je podijeljena redom na petine, osmine i desetine i nacrtane su sve točke na brojevnom pravcu čije su udaljenosti od ishodišta odmjerene tim manjim mjerama. Što je nazivnik veći, te su točke međusobno sve bliže.

Prema tome, razlomci su zapravo brojevi koji određuju točke na brojevnom pravcu. Takve brojeve nazivamo **racionalnim brojevima**. Naziv dolazi od latinske riječi *ratio* koja znači omjer, razlomak. Racionalan je broj koji se može prikazati kao razlomak. Znamo da postoje i brojevi koji se ne mogu prikazati kao razlomci. Takve brojeve nazivamo **iracionalnim brojevima**.

Skraćivanje razlomaka

Na slici 9.1. vidimo da se isti racionalni broj može prikazati s pomoću više razlomaka. Tako razlomci 4/8, 2/4 i 1/2 predstavljaju isti broj. Naime, ako brojnik i nazivnik imaju zajedničke faktore, oni se mogu podijeliti tim faktorima pri čemu se vrijednost razlomka ne mijenja. Taj postupak zovemo skraćivanjem razlomka. Skraćivanje se može provoditi tako dugo dok brojnik i nazivnik ne postanu relativno prosti. Umjesto da skraćivanje provodimo postupno, možemo naći najveći zajednički djelitelj brojnika i nazivnika i skraćivanje obaviti u jednom koraku. Razlomke obično zapisujemo u takvom skraćenom obliku u kojem su brojnik i nazivnik relativno prosti brojevi.

Pravi i nepravi razlomci

Pogledajmo ponovno sliku 9.1. i ustanovimo da razlomci koji se nalaze u intervalu (0, 1) imaju brojnik manji od nazivnika. Takve razlomke obično nazivaju **pravim razlomcima**. Razlomci čija je vrijednost veća od jedan, tj. razlomci čiji su brojnici veći od nazivnika nazivaju se i **nepravim razlomcima**.

Prvi način definiranja razlomaka

Razlomak možemo definirati tako da varijabli pridružujemo vrijednost `Fraction(brojnik, nazivnik)`. Pogledajmo to u interaktivnom sučelju:

```
>>> from fractions import *                                    #1
>>> a = Fraction(1, 2)                                         #2
>>> a                                                          #3
Fraction(1, 2)
>>> print(a)
1/2
>>> b = Fraction(2, 8)                                         #4
>>> b                                                          #5
Fraction(1, 4)
>>> print(b)                                                   #6
1/4
```

Nakon što smo naredbom (#1) importirali modul `fractions`, naredbom (#2) definirali smo prvi razlomak. U naredbi (#3) upisivanjem imena varijable možemo vidjeti kako je taj razlomak pohranjen u spremniku računala, a naredba `print()` ispisat će razlomak s ukošenom razlomačkom crtom.

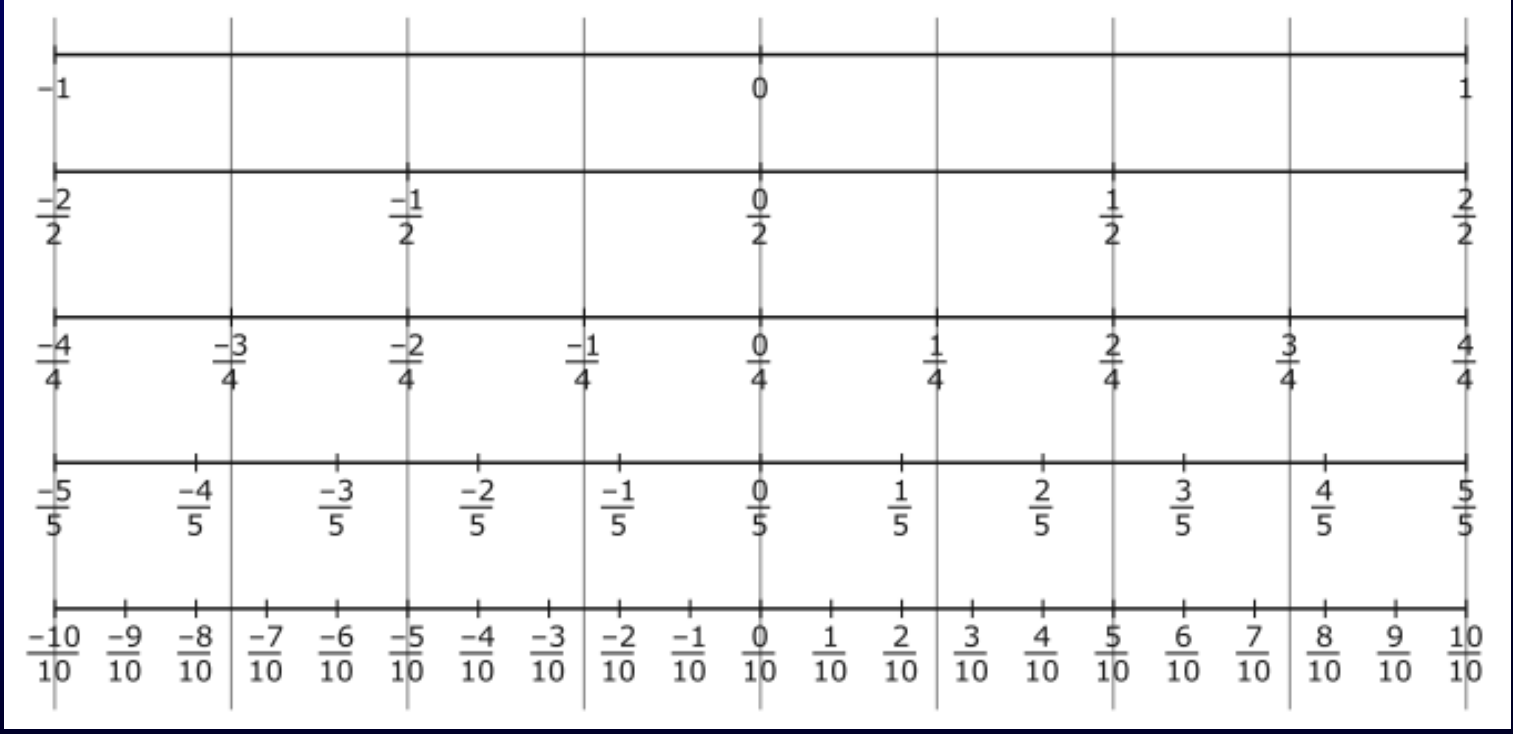
Naredbom (#4) želimo pripisati varijabli `b` vrijednost `2/8`. Međutim, modul `fractions` pronalazi da brojnik i nazivnik nisu međusobno prosti i automatski obavlja dijeljenje brojnika i nazivnika sa `2` i u varijablu `b` pohranjuje skraćeni oblik razlomka. U to se možemo uvjeriti pogledom na pohranjenu vrijednost naredbom (#5) ili ispisom njegove vrijednosti naredbom (#6).

Pogledajmo sliku 9.1. i uvjerimo se da racionalni brojevi u intervalu [0, 1] koji su međusobno razmaknuti za 1/8 mogu biti zapisani u skraćenom obliku sljedećim nizom 0, 1/8, 1/4, 3/8, 1/2, 5/8, 3/4, 7/8, 1. Upravo će nam te vrijednosti ispisati sljedeća petlja:

```
>>> for i in range(9):
        print(Fraction(i, 8), end=' ')

0 1/8 1/4 3/8 1/2 5/8 3/4 7/8 1
```

Razlomci mogu poprimiti i negativne vrijednosti. U *Pythonu* negativni predznak dolazi uvijek uz brojnik razlomka. Na slici 9.2. prikazani su dijelovi brojevnih pravaca na kojima su prikazane neke vrijednosti pozitivnih i negativnih razlomaka iz intervala [−1, +1].



Sl. 9.2.

Pogledajmo:

```
>>> a = Fraction(-3, 8)
>>> a
Fraction(-3, 8)
>>> b = Fraction(10, -15)
>>> b
Fraction(-2, 3)
>>> c = Fraction(-4, -7)
>>> c
Fraction(4, 7)
```

Drugi način definiranja razlomaka

Drugi način zadavanja razlomaka je s pomoću stringova oblika `'brojnik/nazivnik'`.

```
>>> a = Fraction('1/2')                                          #1
>>> a
Fraction(1, 2)
>>> b = Fraction('-7/9')                                         #2
>>> b
Fraction(-7, 9)
>>> c = Fraction('5/-7')                                         #3
Traceback (most recent call last):
  File "<pyshell#30>", line 1, in <module>
    c = Fraction('5/-7')
C:\Userso\AppData\Local\Programs\Python\Python35-32\lib\fractions.py",
line 146, in __new__
    numerator)
ValueError: Invalid literal for Fraction: '5/-7'
```

U naredbi (#1) definirali smo razlomak 1/2 i u naredbi (#2) negativni razlomak -7/9. Ovdje se negativni predznak ne može napisati uz nazivnik što vidimo iz neuspjelog pokušaja zadavanja naredbom (#3).

Ovakav je način zadavanja vrijednosti razlomka stringom prikladan kada razlomke unosimo funkcijom `input()`. Znamo da ta funkcija prihvaća znakove utipkane na tipkovnici i vraća ih kao string. Tako možemo pisati:

```
>>> d = input('Upisati razlomak u obliku brojnik/nazivnik: ')          #4
Upisati razlomak u obliku brojnik/nazivnik: 3/7
>>> d
'3/7'
>>> d = Fraction(d)                                                    #5
>>> d
Fraction(3, 7)
```

Naredba (#4) zahtijeva upis stringa, a naredba (#5) pretvorit će taj string u razlomak `Fraction(3, 7)`.

Naučili smo već kod učitavanja brojeva tipa `int` da je razumno učitavanje i pretvorbu obaviti istom naredbom pa ćemo to načiniti i ovdje sljedećom naredbom.

```
>>> e = Fraction(input('Upisati razlomak u obliku brojnik/nazivnik: '))
Upisati razlomak u obliku brojnik/nazivnik: 6/18
>>> e
Fraction(1, 3)
```

Kada razlomke upisujemo putem tipkovnice, činit ćemo to na opisani način.

Cijeli brojevi su također racionalni brojevi i uobičajeno ih pohranjujemo kao tip `int`. Međutim, ako želimo, možemo ih prevesti u tip `Fraction`. Brojnik dobivenog razlomka bit će jednak vrijednosti cijelog broja, a njegov će nazivnik imati vrijednost 1. Pogledajmo:

```
>>> a = 5
>>> b = Fraction(a)                                    #6
>>> b
Fraction(5, 1)
>>> c = -7
>>> d = Fraction(c)                                    #7
>>> d
Fraction(-7, 1)
>>> print(b)                                           #8
5
>>> print(d)                                           #9
-7
>>>
```

Vidimo da će naredba (#6) prevesti broj `a` u tip `Fraction` i da će on interno biti tako prikazan i to s nazivnikom vrijednosti 1. Isto će načiniti naredba (#7) s brojem `c`. No funkcija `print()` u naredbama (#8) i (#9) uredno te razlomke prepoznaje kao cijele brojeve i tako ih ispisuje.

Zbrajanje i oduzimanje razlomaka

Ako su nazivnici razlomaka jednaki, tada se oni mogu zbrajati i oduzimati tako da se im se zbrajaju, odnosno oduzimaju brojnici. Ako su nazivnici različiti, tada prvo moramo naći zajednički nazivnik, svesti razlomke na taj zajednički nazivnik i nakon toga obaviti zbrajanje ili oduzimanje brojnika.

Modul `fractions` obavlja operacije zbrajanja i oduzimanja u skladu sa svim spomenutim pravilima računanja.

```
#Ispis zbroja dvaju učitanih razlomaka u obliku razlomka - program_9_1.py
from fractions import Fraction
a = Fraction(input('Upiši prvi razlomak: '))
b = Fraction(input('Upiši drugi razlomak: '))
c = a + b
print('{} + {} = {}'.format(a, b, c))
```

Ispis za tri primjera:

```
Upiši prvi razlomak: 3/8
Upiši drugi razlomak: 5/6
3/8 + 5/6 = 29/24
>>>
Upiši prvi razlomak: -1/4
Upiši drugi razlomak: 7/8
-1/4 + 7/8 = 5/8
>>>
Upiši prvi razlomak: 1/2
Upiši drugi razlomak: -11/16
1/2 + -11/16 = -3/16
>>>
```

Uspoređivanje razlomaka

Napišimo program kojim ćemo usporediti deset parova nasumično odabranih razlomaka i ispisati je li prvi razlomak veći, jednak ili manji od drugog razlomka.

U petlji ćemo generirati po dva razlomka čiji su brojnici odnosno nazivnici nasumično odabrani brojevi iz intervala [1, 9]. Složenom naredbom `if-elif-else` usporedit ćemo generirane razlomke i ispisati rezultate usporedbe. Jasno je da ćemo pri svakom izvođenju te petlje dobiti neke druge nasumične razlomke.

```
#Program_9_2.py

from fractions import Fraction
from random import randint

for i in range(10):
    r1 = Fraction(randint(1, 9), randint(1, 9))
    r2 = Fraction(randint(1, 9), randint(1, 9))
    if r1 > r2:
        print('{} > {}'.format(r1, r2))
    elif r1 = r2:
        print('{} = {}'.format(r1, r2))
    else:
        print('{} < {}'.format(r1, r2))
```

Ispis:

```
5/4 < 4/3
7/2 > 1/3
5/6 < 6
4/5 < 7
1/2 = 1/2
9/7 < 4
6 = 6
1/2 < 1
4/3 < 2
1 > 4/7
```

Množenje razlomaka

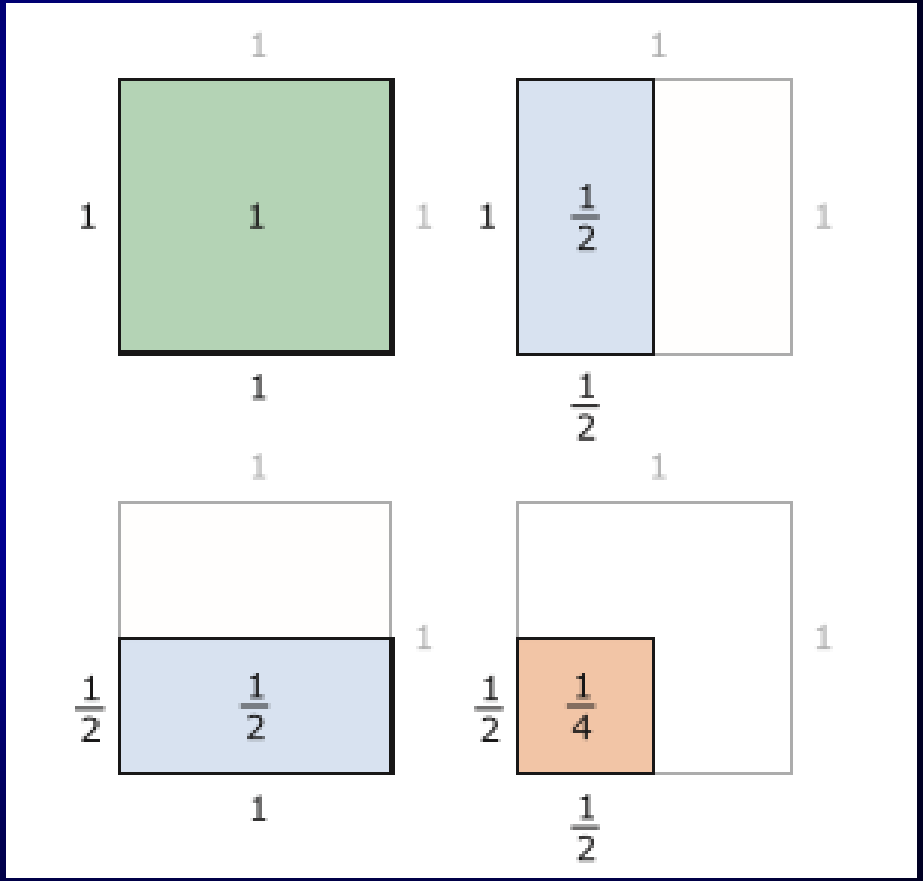
Pri vizualizaciji prirodnih brojeva odabrali smo za jedinicu mjere površinu kvadrata sa stranicama jediničnih dužina.

Nakon toga smo ustanovili da produženjem jedne stranice za faktore koji su prirodni brojevi dobivamo pravokutnike (čija je druga stranica i dalje jedinične dužine). Ako nakon toga produžimo tu drugu stranicu za neki faktor, tada ćemo dobiti površinu koja je jednaka umnošku tih dvaju faktora produljenja.

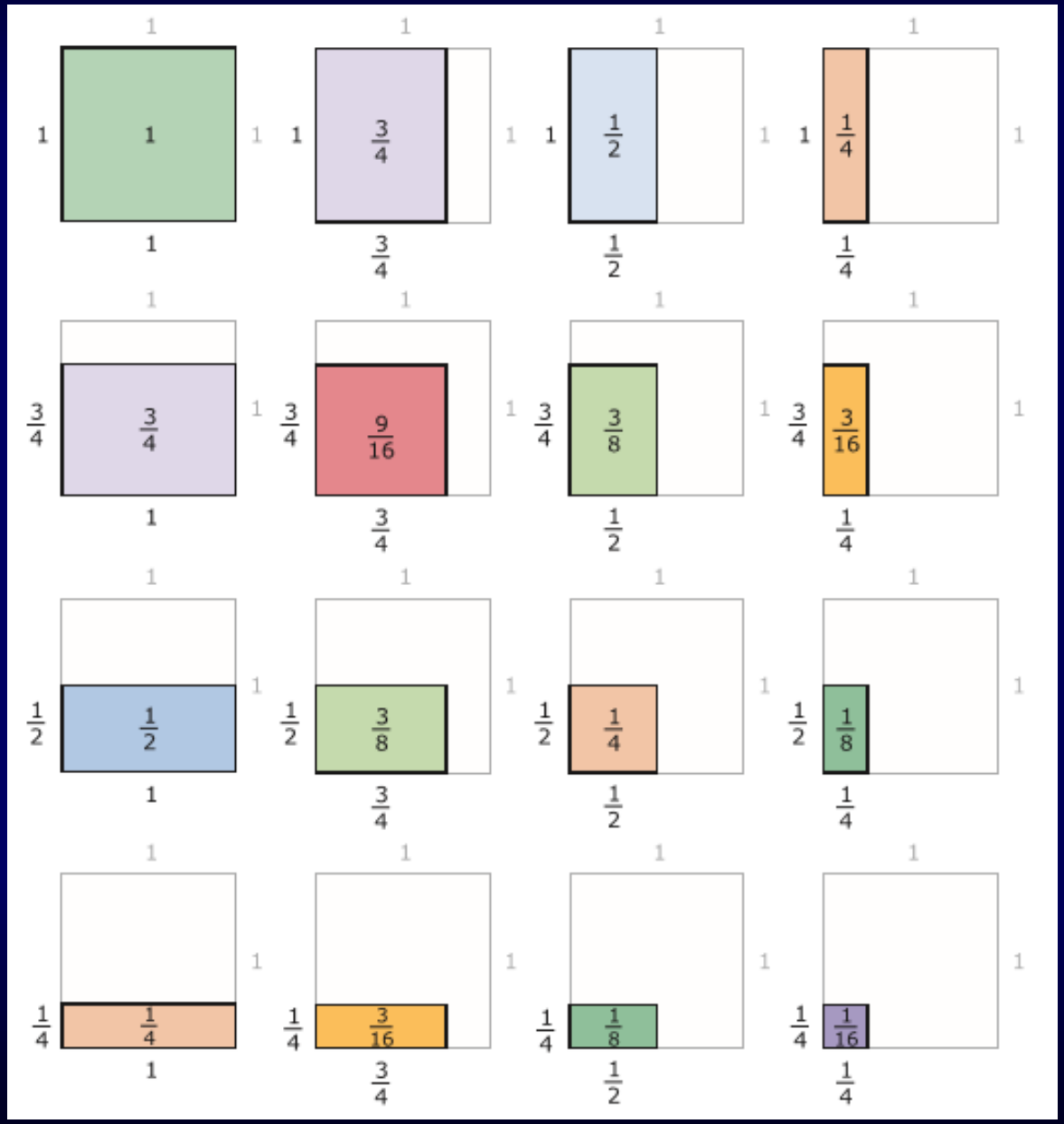
Naziv operacije "množenje" navodi nas na to da se operacijom nešto povećava. Množenjem dvaju prirodnih brojeva dobivamo veći broj.

Množenje razlomaka možemo vizualizirati na sličan način. Opet ćemo pretpostaviti da je jedinica za mjerenje površine jednaka površini kvadrata čije su stranice duljine jedan. Jasno je da će se površina smanjiti ako duljine stranica budemo smanjivali.

Kvadrat sa stranicama 1 ima površinu jednaku 1. Smanjimo li jednu stranicu na vrijednost 1/2, površina dobivenog pravokutnika bit će jednaka $1 \cdot (1/2) = 1/2$. Po pravilu komutacije faktore množenja možemo i zamijeniti pa ćemo opet dobiti jednaku površinu $(1/2) \cdot 1 = 1/2$. Smanjimo li obje stranice za isti faktor, dobit ćemo površinu jednaku $(1/2) \cdot (1/2) = 1/4$.



Umnošci razlomaka s faktorima 1/4, 2/4, 3/4, 4/4



sl. 9.5.

Za razlomke (odnosno racionalne brojeve) u modulu `fractions`, to ćemo napraviti s operatorom množenja `*`. Pogledajmo to u interaktivnom sučelju:

```
>>> from fractions import Fraction                                    #1
>>> a = Fraction (5, 8)
>>> b = Fraction (2, 3)
>>> a                                                                 #2
Fraction(5, 8)
>>> b                                                                 #3
Fraction(2, 3)
>>> a * b                                                             #4
Fraction(5, 12)
>>> print('{} * {} = {}'.format(a, b, a * b))
5/8 * 2/3 = 5/12
```

Nakon importiranja modula `fractions` naredbom (#1) pridružili smo varijablama `a` i `b` vrijednosti razlomaka. Naredbama (#2) i (#3) vidimo kako su pohranjeni racionalni brojevi u tim varijablama. Naredba (#4) obavit će operaciju množenja i skraćivanje najvećim zajedničkim djeliteljem dobivenih brojnika i nazivnika.

Primjer:

Napišimo program koji će izračunati sve umnoške razlomaka 1/4, 2/4, 3/4 i 4/4, koje smo vizualizirali slikom 9.5.

```
#Program_9_4.py

from fractions import Fraction

for i in range(4):
    for j in range(4):
        a = Fraction (4 - i, 4)
        b = Fraction (4 - j, 4)
        print('{} * {} = {}'.format(a, b, a * b))
```

Ispis:

```
1 * 1 = 1
1 * 3/4 = 3/4
1 * 1/2 = 1/2
1 * 1/4 = 1/4
3/4 * 1 = 3/4
3/4 * 3/4 = 9/16
3/4 * 1/2 = 3/8
3/4 * 1/4 = 3/16
1/2 * 1 = 1/2
1/2 * 3/4 = 3/8
1/2 * 1/2 = 1/4
1/2 * 1/4 = 1/8
1/4 * 1 = 1/4
1/4 * 3/4 = 3/16
1/4 * 1/2 = 1/8
1/4 * 1/4 = 1/16
```

Ispis koji smo dobili malo je neuredan. Prisjetimo se da smo u odjeljku 5.7. naučili kako se ispisni string može ljepše oblikovati. Ako broj želimo poravnati lijevo, koristit ćemo znak '<', za desno poravnanje koristit ćemo znak '>', a za ispis u sredini zadanog polja koristit ćemo znak '^'.

Pripremimo li ispisni string na sljedeći način:

```
print('{:>3} * {:>3} = {}'.format(str(a), str(b), str(a * b)))
```

dobit ćemo ovakav ispis:

```
  1 *   1 = 1
  1 * 3/4 = 3/4
  1 * 1/2 = 1/2
  1 * 1/4 = 1/4
3/4 *   1 = 3/4
3/4 * 3/4 = 9/16
3/4 * 1/2 = 3/8
3/4 * 1/4 = 3/16
1/2 *   1 = 1/2
1/2 * 3/4 = 3/8
1/2 * 1/2 = 1/4
1/2 * 1/4 = 1/8
1/4 *   1 = 1/4
1/4 * 3/4 = 3/16
1/4 * 1/2 = 1/8
1/4 * 1/4 = 1/16
```

Primjer:

Napišimo program za vježbanje množenja razlomaka kod kojeg se u funkciji `main()` *traži upis broja zadataka* `n` *koje bi program trebao zadavati i najveći nazivnik* `naz_maks` *koji bi se u zadanim zadatcima trebao pojaviti. Brojnik i nazivnik trebaju biti nasumični brojevi pri čemu nazivnik treba biti veći ili najviše jednak brojniku. Odabirom najvećeg nazivnika možemo odrediti težinu zadataka koji će program zadavati. Program zahtijeva upisivanje rezultata tako dugo dok se ne upiše točan rezultat.*

Uz modul `fractions` *importirat ćemo i funkciju* `randint` *iz modula* `random`*. Prvo ćemo definirati funkciju* `razlomak()` *u kojoj ćemo generirati razlomak s brojnikom koji je nasumičan broj iz intervala* `[1, naz_maks]` *te nazivnikom koji će biti nasumičan broj veći ili jednak njegovu brojniku tj. iz intervala* `[brojnik, naz_maks]`*.*

Zatim ćemo definirati funkciju `množenje()` *u kojoj ćemo u petlji* `for` *odrediti* `n` *zadataka tako što ćemo za svaki novi razlomak pozivati funkciju* `razlomak()`*. Nakon što odredimo dva razlomka s pomoću petlje* `while` *ćemo omogućiti da se ponavlja upis rezultata tako dugo dok se ne upiše točan rezultat. Tu će nam dobro doći naredba* `break` *čije smo djelovanje opisali u 5. poglavlju.*

```python
#Vježbanje množenja razlomaka - program_9_5.py

from random import randint
from fractions import Fraction

def razlomak(naz_maks):
    '''Funkcija će odrediti pravi razlomak s
       nazivnikom ne većim od naz_maks
    '''
    brojnik = randint(1, naz_maks)
    nazivnik = randint(brojnik, naz_maks)
    return Fraction (brojnik, nazivnik)

def množenje(naz_maks, n):
    '''Funkcija će zadati n zadataka množenja pravih razlomaka '''
    for i in range(n):
        print(40 * '-')
        a = razlomak(naz_maks)
        b = razlomak(naz_maks)
        while True:
            c = Fraction (input('Izračunaj: {} * {} = '.format(a, b)))
            if c == a * b:
                break
            print('Odgovor je netočan!')
        print('Odgovor je točan!')
        print('Skraćeni oblik razlomka je: {}'.format(a * b))

def main():
    n = int(input('Upiši broj zadataka n = '))
    naz_maks = int(input('Najveći nazivnik neka bude = '))
    množenje(naz_maks, n)
main()
```

Kao točan rezultat prihvaća se i neskraćeni oblik razlomka, ali će na kraju zadatka biti ispisan njegov skraćeni oblik (i to bez obzira jesmo li upisali skraćeni ili neskraćeni oblik). Izvođenjem programa dobili smo sljedeći ispis (jasno, kod svakog izvođenja to će biti neki drugi brojevi).

```
Upiši broj zadataka n = 3
Najveći nazivnik neka bude = 20
----------------------------------------
Izračunaj: 8/13 * 7/9 = 56/117
Odgovor je točan!
Skraćeni oblik razlomka je: 56/117
----------------------------------------
Izračunaj: 1/2 * 7/18 = 7/36
Odgovor je točan!
Skraćeni oblik razlomka je: 7/36
----------------------------------------
Izračunaj: 12/13 * 3/16 = 12/52
Odgovor je netočan!
Izračunaj: 12/13 * 3/16 = 36/208
Odgovor je točan!
Skraćeni oblik razlomka je: 9/52
```

Dijeljenje razlomaka

Iz matematike nam je poznato da se operacija dijeljenja provodi na taj način da se djeljenik pomnoži recipročnom vrijednošću djelitelja.

Pogledajmo neke primjere:

- $(3/4) : (1/2) = (3/4) \cdot (2/1) = (3/2)$ – dobili smo nepravi razlomak koji možemo napisati kao mješoviti broj $1+(1/2)$
- $(3/7) : (8) = (3/7) \cdot (1/8) = (3/56)$ – jer je recipročna vrijednost broja 8 jednaka 1/8
- $(2) : (5) = (2) \cdot (1/5) = (2/5)$ – pravilo vrijedi i za cijele brojeve.

U *Pythonu* se upravo kao operator dijeljenja rabi i kosa crta `/`. Sjetimo se da smo do sada (dok smo se bavili cijelim brojevima, brojevima tipa `int`) kao znak cjelobrojnog dijeljenja rabili dvostruku kosu crtu `//`. Takvo dijeljenje kao rezultat daje količnik, a ostatak (ako djeljenik nije djeljiv djeliteljem) dobivamo primjenom operatora `%`. Nadalje, upoznali smo i funkciju `divmod()` koja vraća količnik i ostatak dijeljenja. Prisjetimo se toga na jednom primjeru:

```
>>> a = 17
>>> b = 3
>>> a // b
5
>>> a % b
2
>>> količnik, ostatak = divmod(17, 3)
>>> količnik
5
>>> ostatak
2
```

Kod racionalnih brojeva (znamo da oni uključuju razlomke i cijele brojeve) nema problema djeljivosti – operacijom dijeljenja u kojima su djeljenik i djelitelj racionalni brojevi dobivamo rezultat koji je također racionalni broj.

Dijeljenje racionalnih brojeva ćemo obilježavati jednostrukom kosom crtom.

```
>>> from fractions import Fraction
>>> a = Fraction(17)                                    #1
>>> b = Fraction(3)                                     #2
>>> a / b                                               #3
Fraction(17, 3)
>>> print(a / b)                                        #4
17/3
>>> d = Fraction(1)                                     #5
>>> d
Fraction(1, 1)
>>> d / 5                                               #6
Fraction(1, 5)
>>> print(d / 5)                                        #7
1/5
```

Naredbama (#1) i (#2) definirali smo brojeve 17 i 3 kao racionalne brojeve pa će operacija dijeljenja u naredbi (#3) izračunati racionalni količnik 17/3. Interni prikaz tog količnika pokazuje nam naredba (#3), a njegov ispis u interaktivnom sučelju naredba (#4). Dovoljno je da samo jedan operand bude definiran kao racionalni broj kao u naredbi (#5) dok drugi operand može biti i cijeli broj pa da rezultat bude racionalni broj kao što smo dobili naredbom (#6), odnosno (#7). To pravilo vrijedi i za sve druge aritmetičke operacije s racionalnim brojevima.

S operatorom dijeljenja `/` treba biti oprezan. On će djelovati ovako kako smo ga opisali samo za racionalne brojeve. Kada ga primijenimo na brojeve tipa `int`, dobit ćemo neke sasvim drukčije rezultate:

```
>>> 7 / 8
0.875
>>> 3 / 7
0.42857142857142855
>>> 2 / 5
0.4
```

Decimalnom brojevima bavit ćemo se kasnije.

Racionalni broj 0 možemo dobiti na dva načina :

```
>>> nula_prva = Fraction(0)
>>> nula_prva
Fraction(0, 1)
>>> nula_druga = Fraction()
>>> nula_druga
Fraction(0, 1)
```

Primjer:

Napišimo program koji će u funkciji izračunavati zbroj prvih n *recipročnih vrijednosti brojeva. Takav zbroj možemo prikazati kao:* `1 + 1/2 + 1/3 + … + 1/n`. *U svakom redu ispisa broj recipročnih vrijednosti se povećava za 1.*

```
#Program_9_6.py

from fractions import Fraction

def zbroj_recipročnih_vrijednosti(n):
    zbroj = Fraction()                                  #8
    for i in range(1, n+1):
        if i != n:
            print(1 / Fraction(i), end=' + ')           #9
        else:
            print(1 / Fraction(i), end=' = ')           #10
        zbroj += 1 / Fraction(i)
    print(zbroj)                                        #11

def main():
    n = int(input('Koliko brojeva: '))
    for i in range(1, n + 1):
        zbroj_recipročnih_vrijednosti(i)
main()
```

Naredbom (#8) definirali smo zbroj jednak nuli. Nakon toga u petlji `for` ispisujemo razlomke u istom redu i stavljamo naredbom (#9) znak zbrajanja, a nakon zadnjeg razlomka naredbom (#10) znak jednakosti iza kojeg će naredba (#11) ispisati zbroj recipročnih vrijednosti. Uočimo da smo svaki puta nazivnik prikazali kao racionalni broj.

Izvođenjem programa `program_9_6.py` dobivamo ispis:

```
Koliko brojeva: 10
1 = 1
1 + 1/2 = 3/2
1 + 1/2 + 1/3 = 11/6
1 + 1/2 + 1/3 + 1/4 = 25/12
1 + 1/2 + 1/3 + 1/4 + 1/5 = 137/60
1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 = 49/20
1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 = 363/140
1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 + 1/8 = 761/280
1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 + 1/8 + 1/9 = 7129/2520
1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 + 1/8 + 1/9 + 1/10 = 7381/2520
```

Primjer:

Napišimo program koji će u funkciji zadati `n` zadataka dijeljenja razlomaka s nasumično zadanim razlomcima čiji su brojnici i nazivnici prirodni brojevi iz intervala `[1, gg]`.

U svakom zadatku program treba ispisati najprije dva racionalna broja i između njih znak dijeljenja te nakon toga prvi razlomak i nakon znaka za množenje recipročnu vrijednost drugog razlomka. Na taj način će nam biti olakšan posao jer ćemo računati umnožak dvaju racionalnih brojeva.

```
#Vježbanje dijeljenja razlomaka - program_9_7.py

from random import randint
from fractions import Fraction

def dijeljenje(gg, n):
    '''Funkcija će zadati n zadataka dijeljenja razlomaka
       s nasumično zadanim razlomcima čiji su brojnici
       i nazivnici prirodni brojevi iz intervala [1, gg]
    '''
    for i in range(n):
        print(50 * '-')
        a = Fraction(randint(1, gg), randint(1, gg))                    #12
        b_brojnik = randint(1, gg)                                      #13
        b_nazivnik = randint(1, gg)                                     #14
        b = Fraction (b_brojnik, b_nazivnik)                            #15
        b_rec = Fraction(b_nazivnik, b_brojnik)                         #16
        while True:
            c = Fraction(input('Izračunaj: ({0}) / ({1}) = ({0}) * ({2}) = '\
                               .format(a, b, b_rec, )))                 #17
            if c == a / b:
                break
            print('Odgovor je netočan!')
        print('Odgovor je točan!')

def main():
    n = int(input('Upiši broj zadataka n = '))
    gg = int(input('Brojnik i nazivnik u intervalu [1, gg], gg = '))
    dijeljenje(gg, n)

main()
```

U funkciji `dijeljenje()` zadajemo dva nasumična razlomka. Prvi razlomak određuje se naredbom (#12) tako da brojnik i nazivnik mogu poprimiti nasumične vrijednosti iz intervala [1, gg]. Kod drugog razlomka se naredbama (#13) i (#14) određuju brojnik i nazivnik. Nakon toga se naredbom (#15) definira razlomak, a naredbom (#16) njegova recipročna vrijednost.

Uočimo da smo u ispisnom stringu morali u naredbi (#17) brojevima označiti parametre metode `format()` jer se ispis razlomka `a` pojavljuje na dva mjesta.

U petlji **while** se na jednak način kao i u programu `program_9_6.py` zadaju i rješavaju zadatci. U funkciji `main()` zadaje se broj zadataka `n` i gornja granica intervala [1, gg].

U ovom smo programu razlomke napisali u zagradama kako bismo istaknuli da kosa crta / unutar zagrade predstavlja razlomačku crtu, a jednaka kosa crta / između dvaju razlomaka predstavlja operator dijeljenja.

Primjer:

```
Upiši broj zadataka n = 2
Brojnik i nazivnik u intervalu [1, gg], gg = 10
-------------------------------------------------
Izračunaj: (8/3) / (8/5) = (8/3) * (5/8) = 40/24
Odgovor je točan!
-------------------------------------------------
Izračunaj: (9/2) / (8/3) = (9/2) * (3/8) = 9/16
Odgovor je netočan!
Izračunaj: (9/2) / (8/3) = (9/2) * (3/8) = 27/16
Odgovor je točan!
```

Nepravi razlomci i mješoviti brojevi

Nepravim razlomcima zovemo one koji imaju brojnik veći od nazivnika. To su brojevi koji su veći od jedan. Znamo da se oni mogu napisati kao mješoviti brojevi .

Tako su, primjerice, 7/5 i 17/3 nepravi razlomci koje možemo napisati kao mješovite brojeve 1 + 2/5 i 5 + 2/3.

U *Pythonu* bismo to mogli predstaviti na sljedeći način:

```
>>> from fractions import Fraction
>>> a = Fraction(7, 5)                                    #1
>>> ac, ar = 1, Fraction(2, 5)                            #2
>>> ac + ar                                               #3
Fraction(7, 5)
```

Naredbom (#1) definirali smo nepravi razlomak 7/5. U naredbi (#2) taj smo razlomak predstavili kao mješoviti broj i to kao par koji se sastoji od cijelog dijela `ac` koji je jednak 1 i pravog razlomka `ar` koji je 2/5. S obzirom na to da je dozvoljeno zbrajanje cijelog broja i razlomka, vrlo ćemo jednostavno naredbom (#3) iz tog mješovitog broja dobiti pripadni nepravi razlomak 7/5.

Djelovanje funkcije `divmod()` na vrijednosti tipa `Fraction`

U gornjem smo primjeru naredbom (#2) definirali mješoviti broj tj. odredili cijeli i razlomljeni dio nepravog razlomka. To možemo načiniti i uporabom funkcije `divmod()` na sljedeći način:

```
>>> from fractions import Fraction
>>> b = Fraction(17, 3)
>>> bc, br = divmod(b, 1)                                    #4
>>> bc, br
(5, Fraction(2, 3))
>>> bc + br                                                  #5
Fraction(17, 3)
```

Funkcija `divmod(b, 1)` u naredbi (#4) odredit će cijeli i razlomljeni dio mješovitog broja. Njezino djelovanje možemo protumačiti na sljedeći način: funkcija će odrediti koliko je jedinica sadržano u razlomku `b` i vratiti tu vrijednost kao cijeli dio nepravog razlomka, a ostatak kao razlomljeni dio mješovitog broja. Taj par vrijednosti pohranjujemo u varijable `bc` i `br`. Zbrajanjem tih vrijednosti (#5) dobit ćemo izvorni nepravi razlomak.

Pogledajmo još nekoliko primjera:

```
>>> c = Fraction(2, 3)                                    #6
>>> cc, cr = divmod(c, 1)                                 #7
>>> cc, cr
(0, Fraction(2, 3))
>>> d = Fraction(5)                                       #8
>>> dc, dr = divmod(d, 1)
>>> dc, dr
(5, Fraction(0, 1))
```

Naredbom (#6) definirali smo racionalni broj manji od jedan. U mješovitom prikazu koji dobivamo naredbom (#7) njegov će cijeli dio biti jednak nuli.

Cijeli broj (5) preveden naredbom (#8) u racionalni broj imat će cijeli dio koji je upravo taj broj (5), dok će mu razlomljeni dio biti jednak nuli.

Primjer:

Po uzoru na prethodne primjere napišimo program koji će zadavati n *nasumičnih mješovitih brojeva i tražiti od korisnika da ih napiše u obliku razlomka. Program treba ponavljati zadatak sve dok se ne upiše točan odgovor.*

Kako bismo povećali šansu da brojnik bude veći od nazivnika, gornju granicu brojnika ćemo pomnožiti primjerice sa 5.

```
#Mješoviti brojevi u razlomke - program_9_8.py

from random import randint
from fractions import Fraction

def mješoviti_u_razlomak(gg, n):
    for i in range(n):
        print(50 * '-')
        brojnik = randint(1, 5 * gg)                                    #9
        nazivnik = randint(1, gg)                                       #10
        b = Fraction(brojnik, nazivnik)                                 #11
        bc, br = divmod(b, 1)                                           #12
        while True:
            c = Fraction(input('Mješoviti broj {}+{} je razlomak: '\
                               .format(bc, br)))
            if c == b:
                break
            print('Odgovor je netočan!')
        print('Odgovor je točan!')

def main():
    n = int(input('Upiši broj zadataka n = '))
    print('''Program će u obliku mješovitog broja zadavati neprave
             razlomke s brojnicima iz intervala [1, 5 * gg] i
             nazivnicima iz intervala [1, gg]
          ''')
    gg = int(input('gg = '))
    mješoviti_u_razlomak(gg, n)

main()
```

U naredbi (#9) odredit će se nasumični brojnik u intervalu [1, 5 * gg], a u naredbi (#10) nazivnik u intervalu [1, gg]. Iz ta dva nasumična broja dobit ćemo razlomak `b` (#11) te njemu pripadni mješoviti broj `bc`, `br` naredbom (#12). Dio programa koji ispisuje pitanje i vrednuje odgovor već smo razmatrali u prethodnim programima.

Jednim izvođenjem programa dobili smo sljedeći ispis:

```
Upiši broj zadataka n = 5
Program će u obliku mješovitog broja zadavati neprave
razlomke s brojnicima iz intervala [1, 5 * gg] i
nazivnicima iz intervala [1, gg]
gg = 30
-----------------------------------------------
Mješoviti broj 2+4/23 je razlomak: 50/23
Odgovor je točan!
-----------------------------------------------
Mješoviti broj 4+4/27 je razlomak: 112/27
Odgovor je točan!
-----------------------------------------------
Mješoviti broj 10+1/2 je razlomak: 21/2
Odgovor je točan!
-----------------------------------------------
Mješoviti broj 0+1/14 je razlomak: 1/14
Odgovor je točan!
-----------------------------------------------
Mješoviti broj 6+11/23 je razlomak: 149/23
Odgovor je točan
```

Upisivanje više podataka jednim ulaznim stringom, metoda `split()`

Znamo da funkcija `input()` vraća jedan string koji smo utipkali. Prema tome, za upisivanje više podataka trebalo bi više puta pozivati funkciju `input()`. Metoda `split()` pomoći će nam da taj problem jednostavnije riješimo. Prisjetimo se, metoda je svojevrsna funkcija koju pozivamo tako da je napišemo iza vrijednosti (iza varijable) i to odvojenu točkom.

Metoda `split()` podijelit će ulazni string na manje podstringove i smjestiti ih u listu. U ulaznom stringu moraju se označiti mjesta podjele odabranim znakom ili nizom znakova. Uobičajeno je da taj znak bude zarez ili znak za bjelinu (razmak). Pogledajmo:

```
>>> s = input('Upiši nizove znakova razdvojenih zarezima: ')
Upiši nizove znakova razdvojenih zarezima: 1, Ana, 2, Marko, 3, Pero
>>> s
'1, Ana, 2, Marko, 3, Pero'
>>> lista = s.split(',')                                          #1
>>> lista                                                         #2
['1', ' Ana', ' 2', ' Marko', ' 3', ' Pero']
>>> int(lista[0])                                                 #3
1
>>> int(lista[1])                                                 #4
Traceback (most recent call last):
  File "<pyshell#53>", line 1, in <module>
    int(s_r[1])
ValueError: invalid literal for int() with base 10: ' Ana'
>>> int(lista[2])
2
```

U našem smo primjeru utipkali tri broja i tri imena odvojena zarezom. Zarez je razdjelnik ili separator koji u tom stringu razdvaja podstringove. U naredbi (#1) primijenili smo metodu `split(',')` na ulazni string `s` pri čemu se razdjelnik `','` pojavljuje kao parametar metode. Naredbom (#2) vidimo da je ulazni string podijeljen na podstringove koji su smješteni u listu.

Elemente te liste možemo pojedinačno dohvaćati i dalje ih pojedinačno obrađivati. Tako smo naredbom (#3) element liste `lista[0]` iz stringa `'1'` pretvorili u vrijednost `1` tipa `int`. Pokušaj prevođenja elementa `lista[1]` (#4) ne uspijeva jer funkcija `int()` pronalazi u podstringu znakove koje ne može prevesti u tip `int`.

Ako za separator odaberemo prazninu, možemo postići jednak učinak. Pogledajmo:

```
>>> s = input('Upiši nizove znakova razdvojenih bjelinom: ')
Upiši nizove znakova razdvojenih bjelinom: 1 Ana 2 Marko 3 Pero
>>> s
'1 Ana 2 Marko 3 Pero'
>>> lista = s.split(' ')
>>> lista
['1', 'Ana', '2', 'Marko', '3', 'Pero']
```

Međutim, ako pri upisivanju znakova nismo pažljivi, može nam se dogoditi da utipkamo koju bjelinu (razmak) više. Tada možemo dobiti neočekivanu listu:

```
>>> s = input('Upiši nizove znakova razdvojenih bjelinom: ')
Upiši nizove znakova razdvojenih bjelinom: 1  Ana  2  Marko  3  Pero
>>> s
'1  Ana  2  Marko  3  Pero'
>>> lista = s.split(' ')
>>> lista                                                          #5
['1', '', 'Ana', '', '2', '', 'Marko', '', '3', '', 'Pero']
```

Naredbom (#5) ustanovili smo da su se u listi pojavili elementi koje nismo očekivali – bjeline kao elementi liste. Ovakve neželjene pojave možemo izbjeći ako primijenimo metodu `split()` bez separatora, tj. bez parametra unutar zagrada. Metoda će u tom slučaju izbaciti sve suvišne praznine.

Evo primjera:

```
>>> s = input('Upiši niz brojeva s proizvoljnim razmacima: ')
Upiši niz brojeva s proizvoljnim razmacima: 1 3   78  9 56 12 4  7
>>> s
'1 3   78  9 56 12 4  7'
>>> lista = s.split()
>>> lista                                                          #6
['1', '3', '78', '9', '56', '12', '4', '7']
>>> x = [int(a) for a in lista]                                    #7
>>> x
[1, 3, 78, 9, 56, 12, 4, 7]
```

Nakon što smo utipkali niz brojeva s različitim brojem razmaka i primijenili metodu `split()` bez parametara, vidimo da je metoda `split()` "izbacila" sve suvišne bjeline i one se ne pojavljuju u listi `lista` (#6). Prisjetimo se kako se konstruiraju liste. Naredbom (#7) pretvorili smo listu stringova `lista` u listu brojeva `x` kao što to možemo vidjeti nakon ispisa nove liste.

Primjer:

Napišimo funkciju za upis liste cijelih brojeva tako da se svi brojevi unose u jednom redu s razmakom između svaka dva broja.

```
>>> def upis_liste_brojeva():
        s = input('Upiši elemente liste cijelih brojeva: ')
        return [int(a) for a in s.split()]

>>> ulazna_lista = upis_liste_brojeva()
Upiši elemente liste cijelih brojeva: 7  -3 2 4 -15 8
>>> ulazna_lista
[7, -3, 2, 4, -15, 8]
```

Ovakav upis niza ulaznih podataka može se lako prilagoditi raznim potrebama.

Primjer:

Napišimo funkciju za upis liste racionalnih brojeva.

```
>>> from fractions import Fraction
>>> def upis_liste_razlomaka():
        s = input('Upiši elemente liste razlomaka: ')
        return [Fraction(a) for a in s.split()]
```

Za razliku od funkcije `upis_liste_brojeva()` gdje smo se prilikom konstruiranja liste koristili funkcijom `int()`, u funkciji `upis_liste_razlomaka()` smo se u konstruktoru koristili funkciju `Fraction()`.

Provjerimo izvođenje i ove funkcije:

```
>>> lista_razlomaka = upis_liste_razlomaka()
Upiši elemente liste razlomaka: 1/2   -1/3  2/7    13/15                    #8
>>> lista_razlomaka                                                         #9
[Fraction(1, 2), Fraction(-1, 3), Fraction(2, 7), Fraction(13, 15)]
>>> for x in lista_razlomaka:                                               #10
        print(x, end='   ')                                                 #11

1/2   -1/3   2/7   13/15
```

Nakon poziva funkcije i upisa razlomaka (#8), naredba (#9) ispisuje nam vrijednosti i to kako su one pohranjene u spremniku računala. Naredbama (#10) i (#11) ispisali smo elemente liste u obliku u kojem smo ih utipkali.

Primjer:

Napišimo program koji će u funkciji n *puta zadavati nasumične razlomke i tražiti upis pripadnog mješovitog broja. Brojnici trebaju biti iz intervala [1, 5*gg], a nazivnici iz [1, gg]. Prave razlomke treba upisati kao 0+brojnik/nazivnik.*

Iskoristit ćemo metodu `split()` *i preurediti* `program_9_8.py` *tako da ispisujemo nasumično generirane razlomke te tražimo upisivanje pripadnog mješovitog broja. Mješovite brojeve utipkavat ćemo na način opisan na početku poglavlja. Tako ćemo nepravi razlomak 7/3 zapisati kao 2+1/3. Prave razlomke primjerice 3/4 pisat ćemo kao 0+3/4, a cijele brojeve (bez razlomljenog dijela) kao primjerice 3+0/1. Preuređeni program bi mogao izgledati ovako:*

```python
#Razlomci u mješovite brojeve - program_9_11.py

from random import randint
from fractions import Fraction

def razlomak_u_mješoviti(gg, n):
    for i in range(n):
        print(50 * '-')
        brojnik = randint(1, 5 * gg)
        nazivnik = randint(1, gg)
        b = Fraction(brojnik, nazivnik)
        while True:
            s = input('Upiši {} kao mješoviti broj: '.format(b))
            lista = s.split('+')                                    #12
            c = Fraction(lista[0]) + Fraction(lista[1])             #13
            if c == b:
                break
            print('Odgovor je netočan!')
        print('Odgovor je točan!')

def main():
    n = int(input('Upiši broj zadataka n = '))
    print('''Program će zadavati  nasumične razlomke s brojnicima
             iz intervala [1, 5 * gg] i nazivnicima iz intervala [1, gg] i
             tražiti upis razlomka u obliku mješovitog broja''')
    gg = int(input('gg = '))
    razlomak_u_mješoviti(gg, n)

main()
```

Osnovna razlika u odnosu na `program_9_8.py` je u naredbama (#12) i (#13). U naredbi (#12) metoda `split('+')` sa separatorom `'+'` rastavlja utipkani string na cijeli i razlomljeni dio mješovitog broja. U naredbi (#13) se dva dobivena elementa liste prevode u racionalne brojeve i zbrajaju te se dobiva razlomak koji treba usporediti sa zadanim generiranim razlomkom.

Jednim izvođenjem tog programa dobiven je sljedeći ispis:

```
Upiši broj zadataka n = 5
Program će zadavati  nasumične razlomke s brojnicima
iz intervala [1, 5 * gg] i nazivnicima iz intervala [1, gg] i
tražiti upis razlomka u obliku mješovitog broja
gg = 20
-----------------------------------------------
Upiši 41/6 kao mješoviti broj: 6+5/6
Odgovor je točan!
-----------------------------------------------
Upiši 37/4 kao mješoviti broj: 9+1/4
Odgovor je točan!
-----------------------------------------------
Upiši 14 kao mješoviti broj: 14+0/1
Odgovor je točan!
-----------------------------------------------
Upiši 1/2 kao mješoviti broj: 0+1/2
Odgovor je točan!
-----------------------------------------------
Upiši 81/4 kao mješoviti broj: 20+1/4
Odgovor je točan!
```