INFORMATIČKI KLUB

FUTURA

LIGA PROGRAMIRANJA #2

python

LIGA PROGRAMIRANJA U PYTHONU ZA OSNOVNE ŠKOLE – 4. RADIONICA

Mario Miličević, Informatički klub FUTURA

Dubrovnik, 13. veljače 2016.

Creative Commons

- **slobodno možete:**
  - **Dijelite dalje** — možete umnažati i redistribuirati materijal u bilo kojem mediju ili formatu
  - **Stvarajte prerade** — možete remiksirati, mijenjati i prerađivati djelo
- **pod slijedećim uvjetima:**
  - **Imenovanje** — Morate **adekvatno navesti autora**, uvrstiti link na licencu i **naznačiti eventualne izmjene**. Možete to učiniti na bilo koji razuman način, ali ne smijete sugerirati da davatelj licence izravno podupire Vas ili Vaše korištenje djela.
  - **Nekomercijalno** — Ne smijete koristiti materijal u **komercijalne svrhe**.
  - **Dijeli pod istim uvjetima** — Ako remiksirate, mijenjate ili prerađujete materijal, Vaše prerade morate distribuirati pod **istom licencom** pod kojom je bio izvornik.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela. Najbolji način da to učinite je linkom na ovu internetsku stranicu.

Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.

Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

`Tekst licence preuzet je s http://creativecommons.org/`

Raspored Lige programiranja

- 05.12.2015. – 2. radionica
- 19.12.2015. – 2. kolo Lige programiranja
- 16.01.2016. – 3. radionica
- 30.01.2016. – 3. kolo Lige programiranja
- 13.02.2016. – **4. radionica**
- 27.02.2016. – **4. kolo Lige programiranja**
- …
- Web stranica Lige programiranja:

www.futura.com.hr/liga-programiranja-u-pythonu-2015-2016/

Ponavljanje - liste

- Kako stvoriti niz od 10 cijelih brojeva?
- 1. pokušaj (10 varijabli s 10 vrijednosti):

```
>>> br1, br2, br3, br4, br5, br6, br7, br8,
br9, br10 = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

- 2. pokušaj:

```
>>> brojevi = [1, 2, 3, 4, 5,
6, 7, 8, 9, 10]
>>> print(brojevi)
[1, 2, 3, 4, 5, 6, 7, 8, 9,
10]
```

brojevi (elementi polja).

1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9

indeksi elemenata polja.
Indeksi idu od **0** do **n-1**
(u ovom primjeru do **10-1 = 9**).

Ponavljanje - pristup elem. liste

- Elementima liste se može pristupiti preko indeksa elementa liste

```
>>> brojevi[0]
1
>>> brojevi[1]
2
>>> print(brojevi[4:7])
[5, 6, 7]
>>> brojevi[10]
Traceback (most recent call last):
  File "<pyshell#89>", line 1, in <module>
    brojevi[10]
IndexError: list index out of range
```

GREŠKA: u listi brojevi ne postoji element s indeksom 10, zadnji element liste s 10 brojeva ima indeks 9.

Ponavljanje - operatori za liste

- Za rad s listama koriste se isti operatori kao i za stringove:

Operator	Opis djelovanja
+	spajanje
*****	uvišestručenje - jedan operand je tipa **int**
in	element je sadržan u listi
not in	element nije sadržan u listi

Ponavljanje - ugrađene funkcije za liste

- Za rad s listama postoji više ugrađenih funkcija:

Funkcija	Opis djelovanja
len(a)	vraća duljinu liste **a**
min(a)	vraća najmanju vrijednost elementa liste **a**
max(a)	vraća najveću vrijednost elementa liste **a**
sum(a)	vraća sumu svih elemenata liste **a**
del(a[i])	iz liste **a** uklanja element s indeksom **i**
del(a[i:j])	iz liste **a** uklanja isječak koji započinje indeksom **i**, a završava indeksom **j-1**

Ponavljanje - ugrađene metode za liste

- Postoji i više ugrađenih metoda za liste:

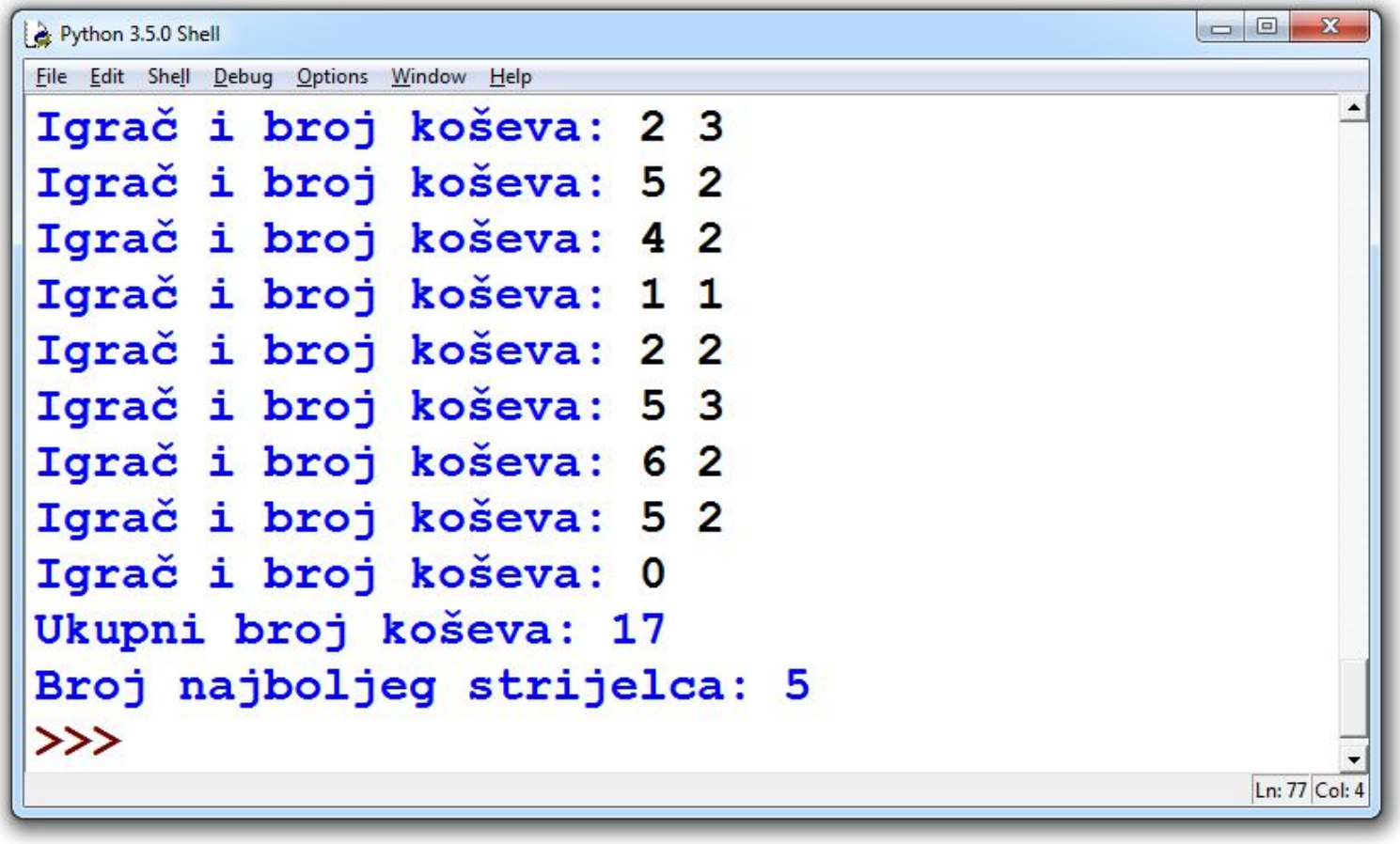
Metoda	Opis djelovanja
L1.append(x)	dodaje element **x** na kraju liste **L1**
L1.insert(i, x)	umeće element **x** prije i-tog elementa liste **L1**
L1.remove(x)	izbacuje element **x** (x je vrijednost elementa)
L1.pop(i)	vraća i izbacuje i-ti element iz liste **L1**. Ako **i** nije naveden vraća i izbacuje zadnji element
L1.reverse()	okreće redoslijed elemenata liste **L1**
L1.sort()	sortira listu **L1** - od najmanje do najveće vrijednosti elementa
L1.index(x)	indeks elementa koji ima vrijednost x. Ako takav element ne postoji vraća pogrešku

Zadatak: Košarka

- Petra bilježi postignute koševe igrača iz košarkaške ekipe njezine škole. U ekipi je 10 igrača koji imaju brojeve od 1 do 10.
- Zabilježiti svaki koš, na način da se u istoj liniji zapiše broj igrača i broj koševa koje je igrač postigao u tom napadu.
- Kraj upisa je kad se upiše '0'.
- Ispisati:
  - koliko je ekipa ukupno postigla koševa,
  - broj igrača koji je postigao najviše koševa.

Zadatak: Košarka

- Primjer izvođenja programa:



Košarka – rješenje

```
kosevi = [0] * 11

ul = input('Igrač i broj koševa: ')
while ul != '0':
    br, kos = ul.split()
    br = int(br)
    kos = int(kos)
    kosevi[br] = kosevi[br] + kos
    ul = input('Igrač i broj koševa: ')

uk = sum(kosevi)
br_naj = kosevi.index(max(kosevi))

print('Ukupni broj koševa:', uk)
print('Broj najboljeg strijelca:', br_naj)
```

kosarka.py

Moduli – zbirke funkcija

- U Pythonu postoji veliki broj ugrađenih funkcija:

```
abs()          dict()       help()       min()        setattr()
all()          dir()        hex()        next()       slice()
any()          divmod()     id()         object()     sorted()
ascii()        enumerate()  input()      oct()        staticmethod()
bin()          eval()       int()        open()       str()
bool()         exec()       isinstance() ord()        sum()
bytearray()    filter()     issubclass() pow()        super()
bytes()        float()      iter()       print()      tuple()
callable()     format()     len()        property()   type()
chr()          frozenset()  list()       range()      vars()
classmethod()  getattr()    locals()     repr()       zip()
compile()      globals()    map()        reversed()   __import__()
complex()      hasattr()    max()        round()
```

Moduli – zbirke funkcija

- Međutim, vidljivo je da primjerice nedostaje fukcija za izračunavanje **drugog korijena**.

$$\sqrt{9} = 3 \quad \rightarrow \quad 3 * 3 = 3^2 = 9$$

- Ta funkcija naravno postoji (`sqrt()`), ali je kao i stotine drugih funkcija pohranjena u module.
- Moduli su zbirke funkcija, u kojima su funkcije grupirane na temelju nekih zajedničkih svojstava. Moduli, odnosno odgovarajuće funkcije, koriste se po potrebi.

Moduli – zbirke funkcija

- Modul se prije korištenja mora uvesti s naredbom **`import naziv_modula`**
- Funkcija **`sqrt`** (korijen) je iz **`math`** modula

```
>>> import math
>>> sqrt(9)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    sqrt(9)
NameError: name 'sqrt' is not defined
>>> math.sqrt(9)
3.0
```

```
>>> math.sqrt(2)
1.4142135623730951
```

Moduli – zbirke funkcija

- Drugi način uvoza funkcija iz modula s naredbom **`from naziv_modula import funkcija1, funkcija2, …`**

```
>>> from math import sqrt, fabs
>>> sqrt(9)
3.0
>>> fabs(-3)
3.0
```

funkcija **fabs** vraća apsolutnu vrijednost broja

$$|3| = 3, \quad |-3| = 3$$

- Ako se žele uvesti sve funkcije iz nekog modula koristi se naredba:

```
>>> from math import *
>>> sqrt(9)
3.0
```

Modul `math`

- Najčešće korištene funkcije modula math:

`sqrt(x)`	korijen broja x
`fabs(x)`	apsolutna vrijednost broja x
`ceil(x)`	zaokruživanje na najmanji cijeli broj veći ili jednak broju x
`floor(x)`	zaokruživanje na najveći cijeli broj manji ili jednak broju x

- https://docs.python.org/3/library/math.html

```
>>> from math import *
>>> ceil(5.2)
6
>>> ceil(6.9)
7
```

```
>>> floor(5.2)
5
>>> floor(6.9)
6
```

Modul `math`

- Izračunati korijen sljedećeg izraza: $\sqrt{\frac{2+10}{3} + 2(3+4) + \frac{11+3}{2}}$

```
>>> from math import *
>>> sqrt((2+10)/3 + 2*(3+4) + (11+3)/2)
5.0
```

- Zaokružiti na veći i na manji cijeli broj sljedeći izraz: $\frac{2+1}{2} + 2(3+1)$

```
>>> ceil((2+1)/2 + 2*(3+1))
10
>>> floor((2+1)/2 + 2*(3+1))
9
```

Zadatak: Pločice

- Meštar postavlja pločice dimenzija 30cm x 30cm na pod pravokutne prostorije. Treba izračunati koliko mu pločica treba.
- Ulazni podaci:

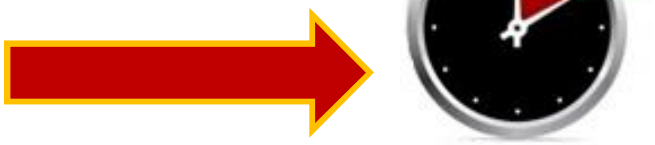
  - duljine stranica prostorije - **a** i **b** se unose u metrima, u istom retku
- Izlazni podatak:

  - broj pločica

- Testni podaci:

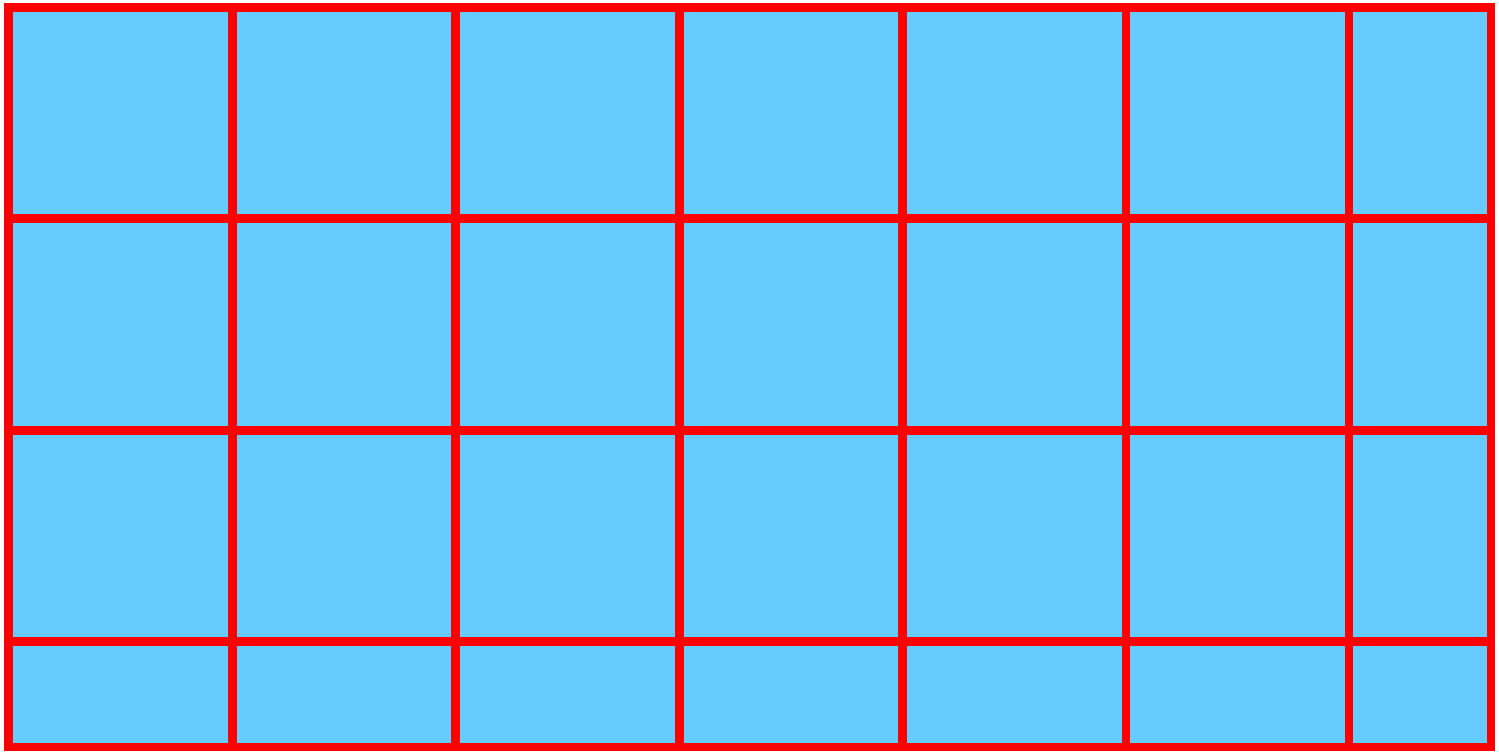
Ulaz	Izlaz
2 4	98
3 5	170
6 9	600

Vrijeme



Zadatak: Pločice

- Primjer: Meštar postavlja pločice dimenzija **30cm x 30cm** na pod pravokutne prostorije **1m x 2m**:



28 pločica

Pločice - rješenje

uvoz funkcije ceil iz modula math

```
from math import ceil
ul = input('Dužina i širina prostorije: ')
a, b = ul.split()
a = int(a)
b = int(b)
br_a = ceil(a * 100 / 30)
br_b = ceil(b * 100 / 30)

uk = br_a * br_b
print('Ukupni broj pločica:', uk)
```

plocice.py

Modul `time`

- Modul **time** sadrži dosta funkcija vezanih za vrijeme, ali nama je trenutno potrebna samo funkcija **time()**:

```
>>> from time import time
>>> print(time())
1455056710.04582
>>> print(time())
1455056717.3822396
```

koliko sekundi je proteklo od početka "epohe" (tj. od 1.1.1970.)

- Funkcija **time()** najviše se koristi za mjerenje proteklog vremena.
- https://docs.python.org/3/library/time.html

Modul `time`

- Primjer: koliko traje zbrajanje svih prirodnih brojeva od 1 do 10,000,000?

```
from time import time
start = time()

suma = 0
for i in range(10000001):
    suma = suma + i
print('Zbroj:', suma)

stop = time()
print('Proteklo vrijeme (s):', stop-start)
```

početak i kraj mjerenja vremena

Zadatak: Najveći djelitelj

- Unijeti broj prirodni broj N i ispisati njegov najveći pravi djelitelj.
- Ispisati i koliko je vremena trajao izračun najvećeg djelitelja.
- Testni podaci:

Ulaz	Izlaz
56	28
1234567	9721
987654321	329218107
1000003	1

Uz najvećeg djelitelja ispisati i proteklo vrijeme (u sekundama)

Vrijeme

Najveći djelitelj - rješenje 1


```
from time import time
djel = 1
N = int(input('Unesi broj: '))
start = time()

for i in range(2, N):
    if N % i == 0:
        djel = i
print('Najveći djelitelj: ', djel)

stop = time()
print('Proteklo vrijeme:', round(stop-start,3))
```

Provjeravaju se svi djelitelji između 1 i N-1

Može li brže?

vrijeme-djelitelj-1.py

Najveći djelitelj - rješenje 2


```
from time import time
djel = 1
N = int(input('Unesi broj: '))
start = time()

for i in range(2, N//2+1):
    if N % i == 0:
        djel = i
print('Najveći djelitelj: ', djel)

stop = time()
print('Proteklo vrijeme:', round(stop-start,3))
```

Dovoljno je provjeriti djelitelje do N/2!

Može li brže?

vrijeme-djelitelj-2.py

Najveći djelitelj - rješenje 3


```
from time import time
from math import sqrt
djel = 1
broj = int(input('Unesi broj: '))
start = time()

for i in range(2, int(sqrt(broj))+1):
    if broj % i == 0:
        djel = broj // i
        break
print('Najveći djelitelj: ', djel)
stop = time()
print('Proteklo vrijeme:', round(stop-start,3))
```

Provjeravaju se djelitelji između 1 i $\sqrt{N}$

Čim se nađe prvi djelitelj >1 odmah je jasno koji je najveći djelitelj!

vrijeme-djelitelj-3.py

Modul `random`

- Funkcije za generiranje slučajnih brojeva:

`randint(a, b)`	vraća slučajni cijeli broj n koji je a <= n <= b
`random()`	vraća slučajni realni broj n koji je 0.0 <= n < 1.0
`uniform(a, b)`	vraća slučajni realni broj n koji je a <= n <= b ako je a <= b ili je b <= n <= a ako je b < a
`sample(N, k)`	vraća listu od k jedinstvenih elemenata iz liste N

- https://docs.python.org/3/library/random.html

```
>>> from random import *
>>> randint(0, 10)
5
>>> randint(0, 10)
9
```

slučajni cijeli broj u intervalu [0, 10]

Modul random

```
>>> from random import *
>>> random()
0.8460300294602602
>>> random()
0.9592937131735048
>>>
>>> uniform(0, 10)
1.594305867774457
>>> uniform(0, 10)
3.394179944212329
>>>
>>> brojevi = [i for i in range(20)]
>>> sample(brojevi, 5)
[14, 7, 16, 11, 17]
```

slučajni realni broj u intervalu [0, 1)

slučajni realni broj u intervalu [0, 10]

definicija liste koja sadrži 20 brojeva od 0 do 19

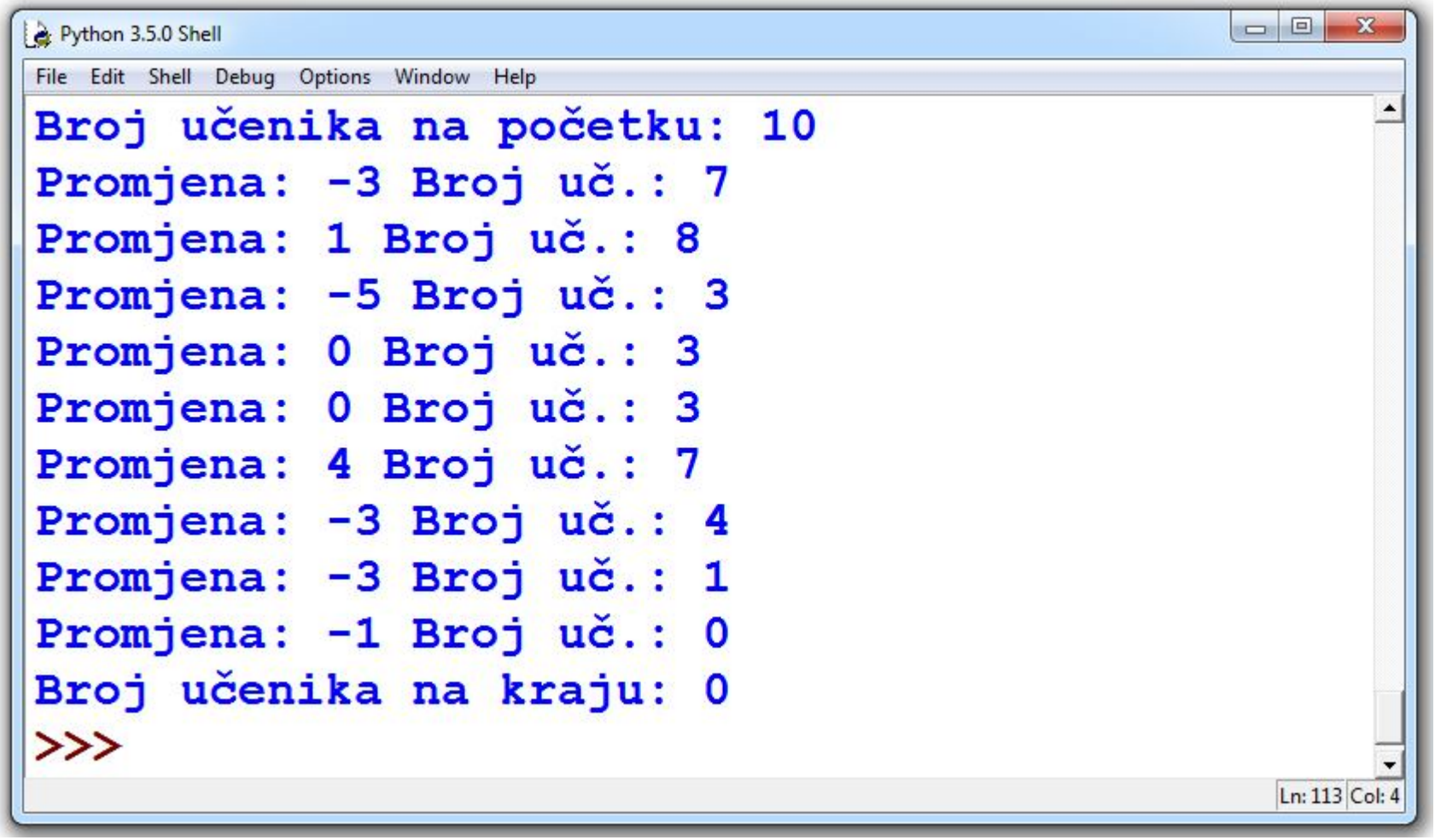
lista od 5 jedinstvenih brojeva iz liste *brojevi*

Zadatak: Učionica

- Futura organizira Dane otvorenih vrata, i cijeli dan dolaze i odlaze učenici. U učionicu može stati 20 učenika, a na početku ih ima 10.
- U nastavku se svakih 10 minuta bilježi za koliko se promijenio broj učenika. Broj učenika se može promijeniti u rasponu od -5 do 5.
- Pomoću generatora slučajnih brojeva simulirati kretanje broja učenika.
- Zaustaviti program kad se učionica napuni (20 učenika) ili sasvim isprazni (0 učenika)!

Zadatak: Učionica

- Primjer pokretanja programa:



Učionica - rješenje


```
from random import randint
br_uc = 10
print('Broj učenika na početku:', br_uc)

while br_uc < 20 and br_uc > 0:
    promj = randint(-5, 5)
    br_uc = br_uc + promj
    if br_uc > 20:
        br_uc = 20
    if br_uc < 0:
        br_uc = 0
    print('Promjena:', promj, 'Broj uč.:', br_uc)

print('Broj učenika na kraju:', br_uc)
```

Za simulaciju kretanja broja učenika koristi se funkcija randint()

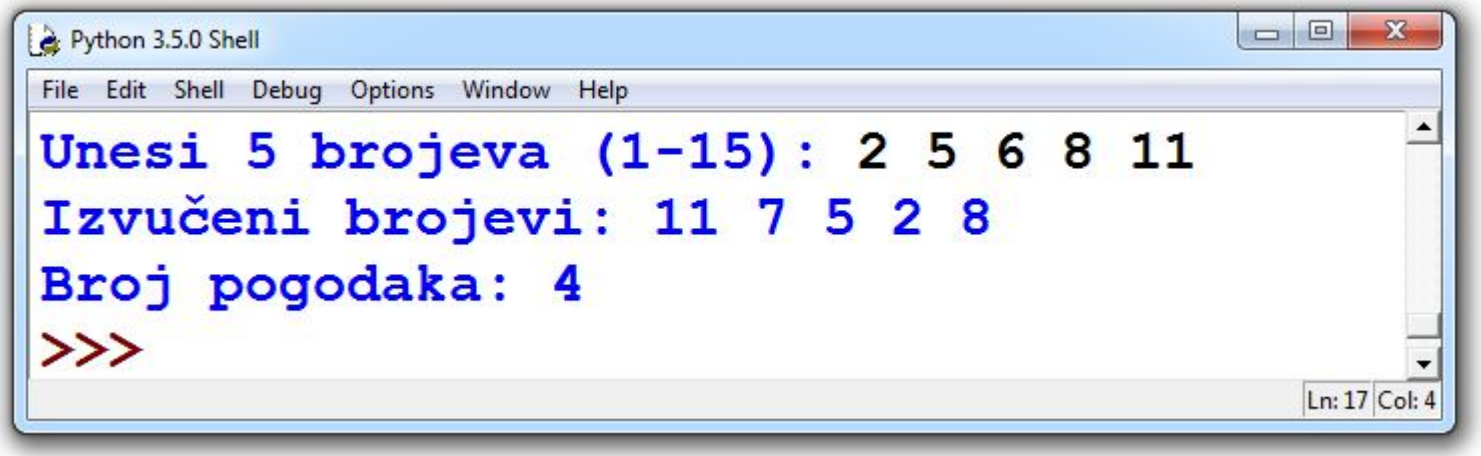
ucionica.py

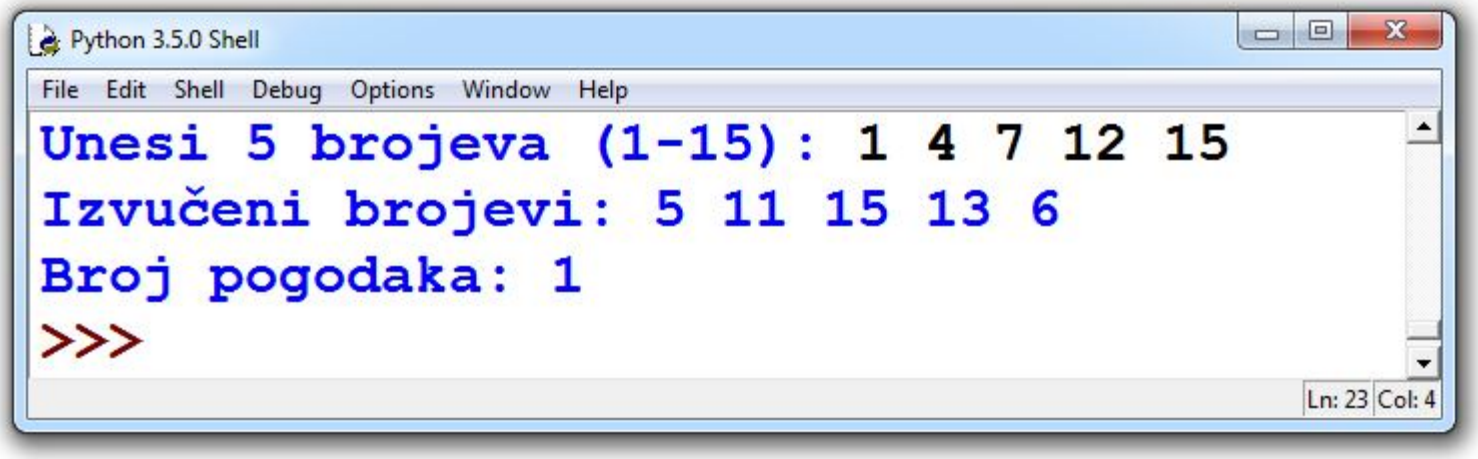
Zadatak: Loto

- Mateo igra Loto 5 od 15. Napisati program koji će simulirati izvlačenje brojeva Lota.
- Ulazni podaci:
  - 5 brojeva koje je Mateo odabrao - u istom retku, odvojeni prazninama
  - nakon toga se generira slučajna Loto kombinacija
- Izlazni podatak:
  - koliko je brojeva Mateo pogodio

Zadatak: Loto

- Primjeri pokretanja programa:





Loto - rješenje


```
from random import sample
brojevi = [str(i) for i in range(1, 16)]
pog = 0

ul = input('Unesi 5 brojeva (1-15): ')
mateo = ul.split()
komb = sample(brojevi, 5)
for i in mateo:
    if i in komb:
        pog += 1

print('Izvučeni brojevi:', end=' ')
for i in komb:
    print(i, end=' ')
print('\nBroj pogodaka:', pog)
```

Funkcija split() puni listu mateo s 5 elemenata

Za simulaciju izvlačenja brojeva Lota koristi se funkcija sample()

loto.py

Ne zaboravite!

- Za 15 dana – **u subotu** **27.02.2016.**
  4. kolo Lige programiranja
- **5./6. razredi** - početak **10:00**
- **7./8. razredi** - početak **10:00**
- **3** zadatka rješavate **75** minuta
- nemojte kasniti!