INFORMATIČKI KLUB

FUTURA

LIGA PROGRAMIRANJA #2

python

LIGA PROGRAMIRANJA U PYTHONU ZA OSNOVNE ŠKOLE – 2. RADIONICA

Mario Miličević, Informatički klub FUTURA

Dubrovnik, 5. prosinca 2015.

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

- 07.11.2015. – 1. radionica
- 21.11.2015. – 1. kolo Lige programiranja
- 05.12.2015. – **2. radionica**
- 19.12.2015. – **2. kolo Lige programiranja**
- termini u 2016. godini će biti naknadno određeni
- Web stranica Lige programiranja:

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

Ponavljanje: programske petlje

- Ponavljanje bloka naredbi određeni broj puta:

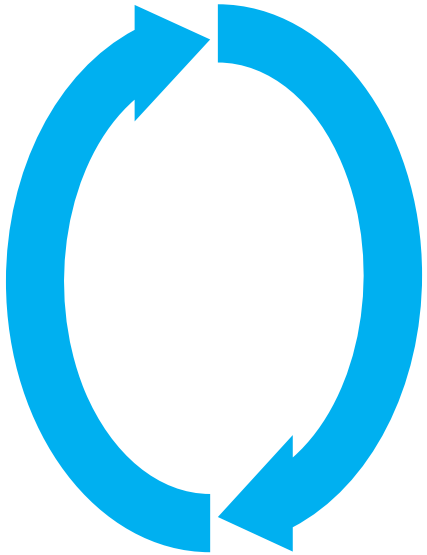
```
...
za i := 1 do n činiti
      naredba_1
      ...
      naredba_z
...
```

- `for` petlja

- Uvjetno ponavljanje bloka naredbi:

```
...
dok je uvjet činiti
      naredba_1
      ...
      naredba_z
...
```

- `while` petlja



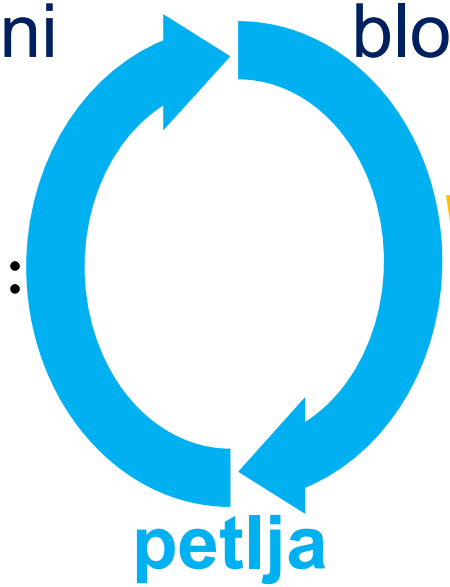
programska petlja

Ponavljanje: programske petlje

- Ponavljanje bloka naredbi određeni broj puta:

```
...
for i in range(n):
      naredba_1
      ...
      naredba_z
...
```

- Blok naredbi će se izvesti **n puta**, za vrijednosti varijable **i** od **0** do **n-1**.



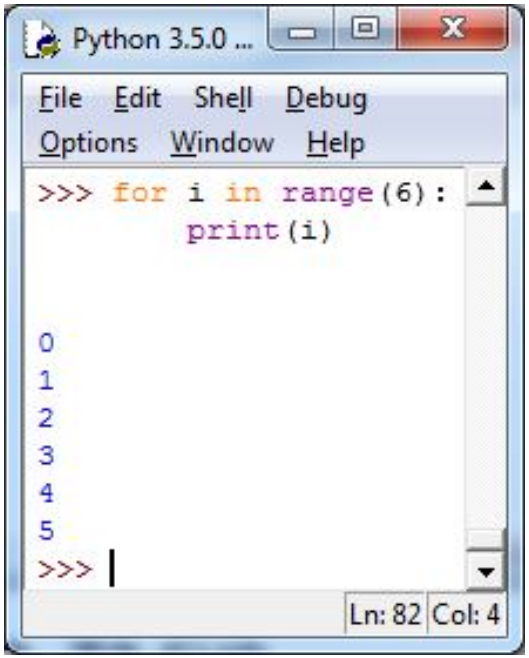
- Uvjetno ponavljanje bloka naredbi:

```
...
while uvjet:
      naredba_1
      ...
      naredba_z
...
```

- Blok naredbi će se izvoditi dok je ***uvjet*** ispunjen (daje vrijednost **True**)

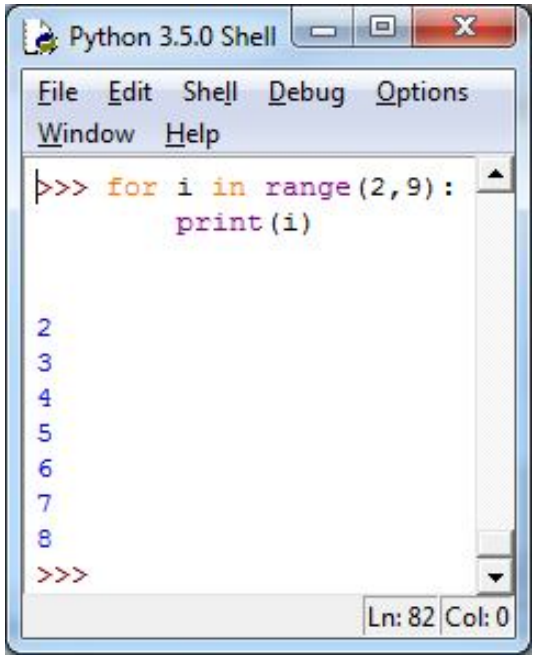
Ponavljanje: vrste **for** petlji

- `range(n)`



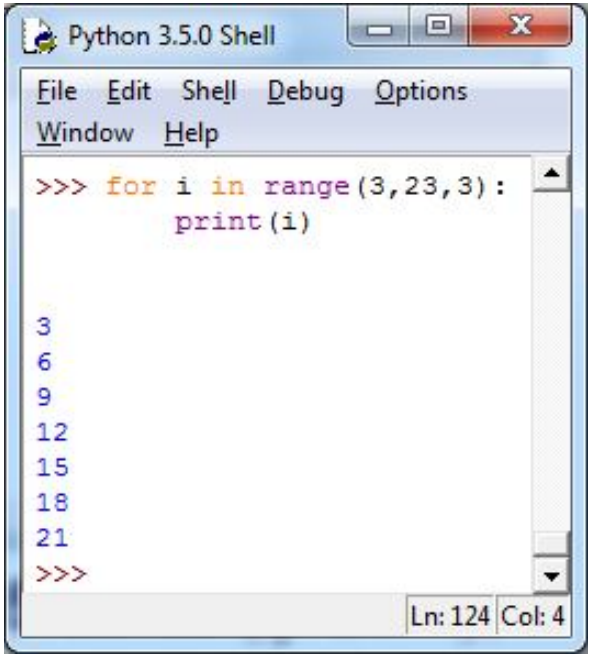
- Blok naredbi se izvodi **6 puta**, za vrijednosti **i** od **0** do **5**.

- `range(n,m)`



- Blok naredbi se izvodi **7 puta**, za vrijednosti **i** od **2** do **8**.

- `range(n,m,k)`



- Varijabla **i** mijenja vrijednost od **3** do **22** (tj. 23-1), s korakom **3**.

Zadatak: Štednja

- Igor je odlučio da će pokušati uštedjeti nešto kuna za N dana (ulazni podatak), i to tako da će ovisno o rednom broju dana svaki dan ili štedjeti ili trošiti kune:
  - ako je redni broj određenog dana neparan, Igor će **ubaciti** u kasicu iznos kuna koji odgovara **dvostrukom** rednom broju dana,
  - ako je redni broj određenog dana paran, Igor će **potrošiti** iz kasice iznos kuna koji odgovara rednom broju dana.
- Koliko je kuna u kasici poslije N dana?

Zadatak: Štednja

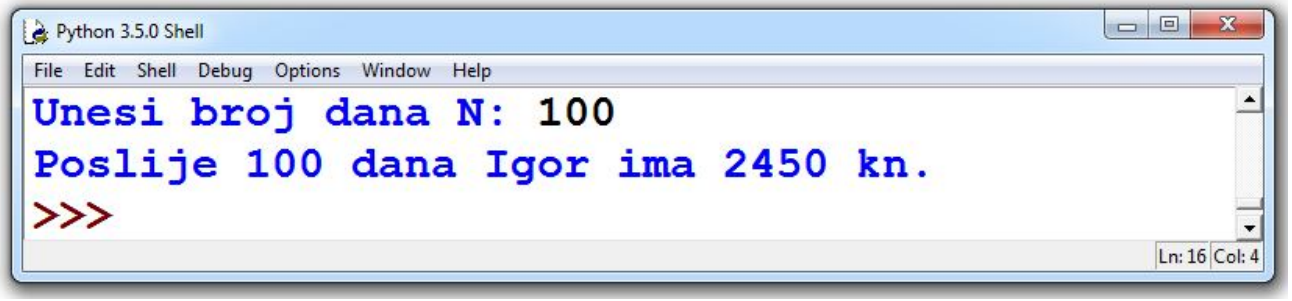
- Primjer: N=**7**, dakle Igor će štedjeti 7 dana:
  - 1. dan - Igor ubaci 2*__1__=2 kn - u kasici su 2 kn
  - 2. dan - Igor potroši **2** kn - u kasici je 0 kn
  - 3. dan - Igor ubaci 2*__3__=6 kn - u kasici je 6 kn
  - 4. dan - Igor potroši **4** kn - u kasici ostaju 2 kn
  - 5. dan - Igor ubaci 2*__5__=10 kn - u kasici je 12 kn
  - 6. dan - Igor potroši **6** kn - u kasici ostaje 6 kn
  - 7. dan - Igor ubaci 2*__7__=14 kn - u kasici je 20 kn
- Poslije N=**7** dana Igor u kasici ima **20** kn.
- Drugi primjer: N=100 => 2450 kn

Zadatak: Štednja - rješenje

```
n = int(input('Unesi broj dana N: '))
stanje = 0

for i in range (1, n+1):
    if i % 2 == 0:
        stanje = stanje - i
    else:
        stanje = stanje + 2 * i
print('Poslije', n, 'dana Igor ima', stanje, 'kn.')
```

stednja.py



Metoda split

- Metoda **`split`** vraća listu riječi iz zadanog niza znakova (standardni razdjelnik je praznina ‘ ‘)

```
>>> tekst = 'Liga programiranja u Pythonu'
>>> tekst.split()
['Liga', 'programiranja', 'u', 'Pythonu']
```

- Korisnik može kod poziva metode **`split`** postaviti razdjelnik po želji

```
>>> vrijeme = '17:30:45'
>>> vrijeme.split(':')
['17', '30', '45']
```

Varijable i višestruko pridruživanje

- Višestruko pridruživanje vrijednosti varijablama

```
>>> broj1, broj2 = 1, 5
>>> print('broj1 =', broj1, 'broj2 =', broj2)
broj1 = 1 broj2 = 5
```

- Ako sve varijable na početku imaju istu vrijednost može se koristiti sljedeći format

```
>>> brojac = suma = 0
>>> print(brojac, suma)
0  0
```

Funkcija split

- Rezultat primjene metode **`split`** može se pohraniti u varijable

```
>>> vrijeme = '17:30:45'
>>> hh, mm, ss = vrijeme.split(':')
>>> print(hh, 'sati', mm, 'minuta', ss,'sekundi')
17 sati 30 minuta 45 sekundi
```

```
>>> izraz = '12+20'
>>> op1, op2 = izraz.split('+')
>>> print('Operandi:', op1, op2)
Operandi: 12 20
```

Funkcija split

- Naravno treba voditi računa da su rezultati primjene metode **split** i dalje znakovni nizovi

```
>>> izraz = '12+20'
>>> op1, op2 = izraz.split('+')
>>> print('Operandi:', op1, op2)
Operandi: 12 20
>>> rez = op1 - op2
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    rez = op1 - op2
TypeError: unsupported operand type(s) for -:
'str' and 'str'
```

pogreška: varijable op1 i op2 moraju biti tipa int ili float

Zadatak: Ruksak

- Mare pomaže mami i u svojem ruksaku nosi kupljenu spenzu doma. U ruksak se može natovariti najviše 10 kg. Težine prozvoda su izražene u gramima (**g**) ili kilogramima (**kg**).
- Napisati program koji će pomoći Mari odrediti kolika je težina proizvoda u ruksaku. Proizvodi se stavljaju u ruksak dok se ne dogodi da neki proizvod ne može stati radi svoje težine.

Zadatak: Ruksak

- Testni podaci:

Ulaz	Ulaz	Ulaz
2 kg 1 kg 3 kg 1 kg 2 kg 2 kg ← ne može stati u ruksak	3 kg 3000 g 1000 g 3 kg 1000 g 2 kg	1000 g 2000 g 1000 g 1 kg 3000 g 5 kg 1000 g
Izlaz 9 kg	**Izlaz** 10 kg	**Izlaz** 8 kg

Zadatak: Ruksak - rješenje

```
ukTez = 0
while True:
    tez = input('Upiši težinu proizvoda: ')
    tezPr, jedMj = tez.split(' ')
    tezPr = int(tezPr)
    if jedMj == 'g':
        tezPr = tezPr / 1000
    if ukTez + tezPr > 10:
        break
    else:
        ukTez = ukTez + tezPr
print(ukTez, 'kg')
```

„beskonačna” petlja

break: izlaz iz petlje

ruksak.py

Osnovni tipovi podataka u Pythonu

- **`int`** – cijeli broj
- **`float`** – broj s pomičnom točkom
- **`str`** – niz znakova (string)
- **`bool`** – logički tip podatka

niz znakova

```
>>> godina = 2015
>>> radionica = 'Liga progamiranja u Pythonu'
>>> print('Radionice', radionica, godina, 'oš')
Radionice Liga programiranja u Pythonu 2015 oš
```

String – niz znakova

- Za rad sa stringovima definirana su 4 binarna operatora:

Operator	Opis djelovanja
+	spajanje
*****	uvišestručenje - jedan operand je tipa **int**
in	je li prvi string sadržan je u drugom stringu
not in	je li prvi string nije sadržan u drugom stringu

String – niz znakova

Spajanje stringova

```
>>> ime = 'Pero'
>>> prez = 'Perić'
>>> ucenik = ime + prez
>>> print(ucenik)
PeroPerić
>>>
>>> ucenik = ime + ' ' + prez
>>> print(ucenik)
Pero Perić
>>>
```

Koristi se standardni operator za zbrajanje: +

Svi operandi su stringovi!

String – niz znakova

- Uvišestručenje niza znakova

```
>>> fut = 'Futura'
>>> fut3 = fut * 3
>>> print(fut3)
FuturaFuturaFutura
>>>
>>> print(fut3*2)
FuturaFuturaFuturaFuturaFuturaFuturaFutura
>>>
```

Koristi se standardni operator za množenje: *
-> Drugi operand je cijeli broj!

String – niz znakova

- Operatori **in** i **not in**

```
>>> niz1 = 'Radionica programiranja'
>>> niz2 = 'program'
>>> niz2 in niz1
True
>>> 'Rad' in niz1
True
>>> 'rad' in niz1
False
>>> 'rad' not in niz1
True
```

String – niz znakova

- Dohvaćanje pojedinih znakova indeksiranjem

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
I	**n**	**f**	**o**	**r**	**m**	**.**	**k**	**l**	**u**	**b**		**F**	**U**	**T**	**U**	**R**	**A**
-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> fut = 'Inform.klub FUTURA'
>>> print(fut[2])
f
>>> print(fut[2:6])
form
>>> print(fut[0] + fut[2:6])
Iform
```

String – niz znakova

- Dohvaćanje pojedinih znakova indeksiranjem

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
I	**n**	**f**	**o**	**r**	**m**	**.**	**k**	**l**	**u**	**b**		**F**	**U**	**T**	**U**	**R**	**A**
-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> print(fut[-1])
A
>>> print(fut[-1:-5])

>>> print(fut[-5:-1])
UTUR
>>> print(fut[-18:-17])
I
```

Negativni indeks: dohvat znakova od kraja niza

Zadatak: Nađi podniz

- Unosi se niz od 10 znakova. U tom nizu potrebno je naći:
  - 7./8.r.: podniz '**Fut**'
  - 5./6.r.: znak '**F**'
- Ako je znak/podniz pronađen ispisati **DA**, inače ispisati **NE**.
- Napomena: ne koristiti posebne funkcije ili metode za stringove!

Ulaz	Ulaz	Ulaz
I.K.Futura	**funkcija25**	**>Futuristi**
Izlaz	**Izlaz**	**Izlaz**
DA	**NE**	**DA**

Vrijeme

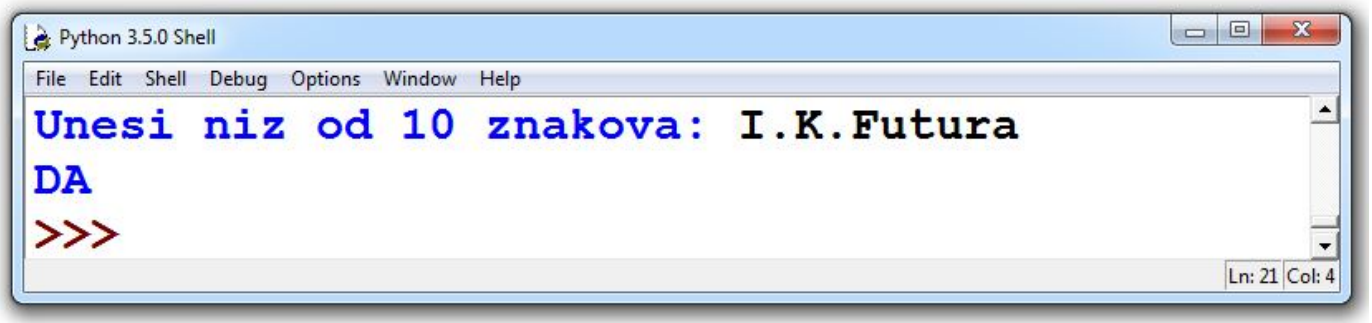
Zadatak: Nađi podniz - rješenje 5./6.

```
niz = input('Unesi niz od 10 znakova: ')
rez = 'NE'

for i in range(0, 10):
    if niz[i] == 'F':
        rez = 'DA'

print(rez)
```

podniz56.py



Zadatak: Nađi podniz - rješenje 7./8.

```
podniz = 'Fut'
k = 0
rez = 'NE'
niz = input('Unesi niz od 10 znakova: ')

for i in range(0, 10):
    if niz[i] == podniz[k]:
        rez = 'DA'
        k += 1
        if k == 3:
            break
    else:
        rez = 'NE'

print(rez)
```

podniz78.py

Ugrađene funkcije za stringove

- Za rad sa stringovima definirano je više funkcija - primjerice:

Funkcija	Opis djelovanja
len(s)	vraća duljinu stringa (broj znakova u stringu)
min(s)	vraća znak iz stringa koji ima najmanju kodnu vrijednost
max(s)	vraća znak iz stringa koji ima najveću kodnu vrijednost
ord(c)	vraća dekadski kod jednog znaka
chr(n)	vraća znak određen zadanim dekadskim kodom **n**
str(n)	vraća znakovni prikaz broja **n**

Ugrađene funkcije za stringove

```
>>> niz1 = 'Futura'
>>> niz2 = 'futura'
>>> len(niz1)
6
>>> min(niz1)
'F'
>>> min(niz2)
'a'
>>> max(niz1)
'u'
>>> max(niz2)
'u'
```

ispis znakova iz niza koji imaju najmanju i najveću kodnu vrijednost - velika slova imaju niže kodne vrijednosti od malih slova

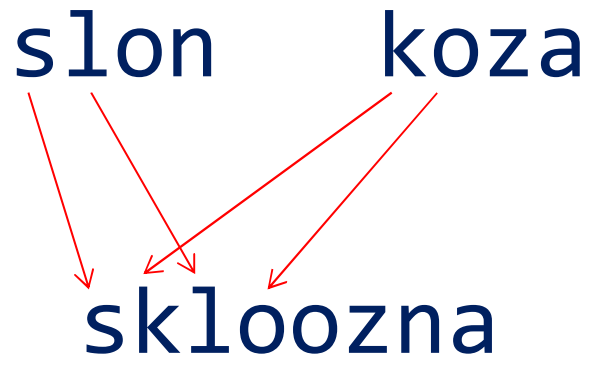
Ugrađene funkcije za stringove

```
>>> ord('F')
70
>>> ord('f')
102
>>> chr(71)
'G'
>>> chr(101)
'e'
>>> str(521)
'521'
```

ASCII (UTF8) kod slova 'F' i slova 'f'

Zadatak: Nove riječi

- Ivan voli stvarati nove riječi. Smislio je igru u kojoj se od dvije riječi iste dužine stvara jedna riječ, na način da se redom uzima po jedno slovo iz svake riječi.
- Primjer:



Zadatak: Nove riječi

- Testni podaci:

Ulaz	Ulaz
ormar zakon	**Ivan Kate**
Izlaz	**Izlaz**
ozramkaorn	**IKvaatne**

- za 7./8.r.: dodatno napraviti verziju programa u kojoj riječi mogu različite dužine!

Zadatak: Nove riječi - rješenje

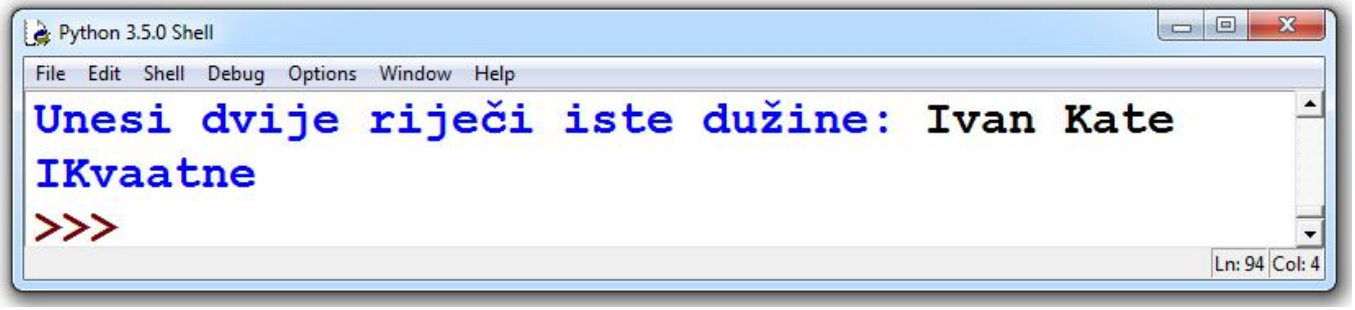
```
novo = ''

rijeci = input('Unesi dvije riječi iste dužine: ')
rijec1, rijec2 = rijeci.split()

for i in range(0, len(rijec1)):
    novo = novo + rijec1[i] + rijec2[i]

print(novo)
```

nove_rijeci.py



Ugrađene metode za stringove

- Za rad sa stringovima definirana je i više metoda - a neke kao što je primjerice **split** već smo koristili:

Metoda	Opis djelovanja
s.lower()	vraća kopiju stringa **s** sa svim malim slovima
s.upper()	vraća kopiju stringa **s** sa svim velikim slovima
s.replace(s1, s2)	vraća kopiju stringa **s** u kojem je svaki podniz **s1** zamijenjem podnizom **s2**
s.find(s1)	vraća poziciju pojavljivanja podniza **s1** u stringu **s**, ili -1 ako podniz nije pronađen
s.count(s1)	vraća broj koliko se puta podniz **s1** pojavljuje u stringu **s**

Ugrađene metode za stringove

```
>>> niz1 = 'Progr.jezik Python'
>>> niz1.upper()
'PROGR.JEZIK PYTHON'
>>> niz1.lower()
'progr.jezik python'
>>> niz1.replace('Py', 'Mara')
'Progr.jezik Marathon'
>>> niz1.find('.')
5
>>> niz1.find('A')
-1
>>> niz1.count('P')
2
```

zamjena jednog podniza drugim

na kojoj se poziciji nalazi znak '.' u stringu niz1

znak 'A' ne postoji u stringu niz1, pa je rezultat -1 (to nije indeks!)

koliko se puta znak 'P' javlja u stringu niz1

Zadatak: Šifriranje 1

- Jasna voli Aniti slati šifrirane poruke tako da:
  - samoglasnike **a**, **e**, **i**, **o**, **u** zamijeni brojevima **1**, **2**, **3**, **4**, **5**;
  - **praznine** zamijeni brojem **0**.

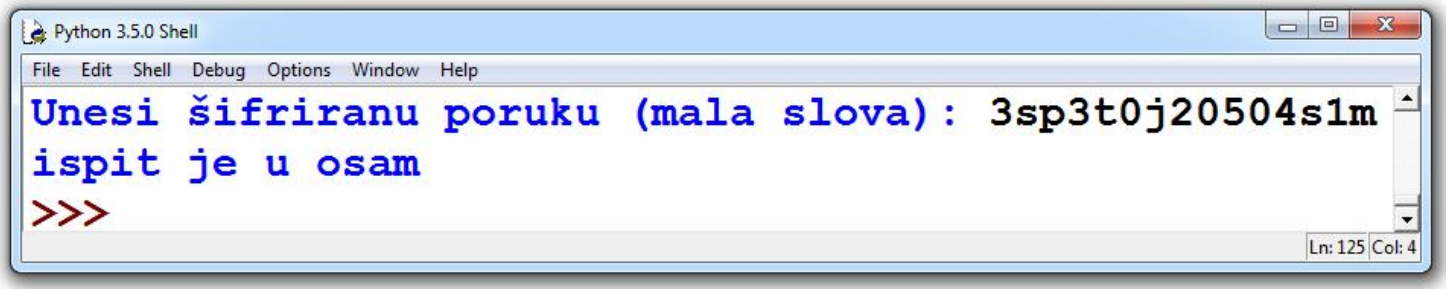
  Poruka je napisana malim slovima!
- Dešifrirati Jasninu poruku!

Ulaz	**Ulaz**
3v40st1ln405č3	**3sp3t0j20504s1m**
Izlaz	**Izlaz**
ivo stalno uči	**ispit je u 8**

Zadatak: Šifriranje 1 - rješenje

```
poruka = input('Unesi šifriranu poruku (mala
                        slova): ')
poruka = poruka.replace('0', ' ')
poruka = poruka.replace('1', 'a')
poruka = poruka.replace('2', 'e')
poruka = poruka.replace('3', 'i')
poruka = poruka.replace('4', 'o')
poruka = poruka.replace('5', 'u')
print(poruka)
```

sifra1a.py



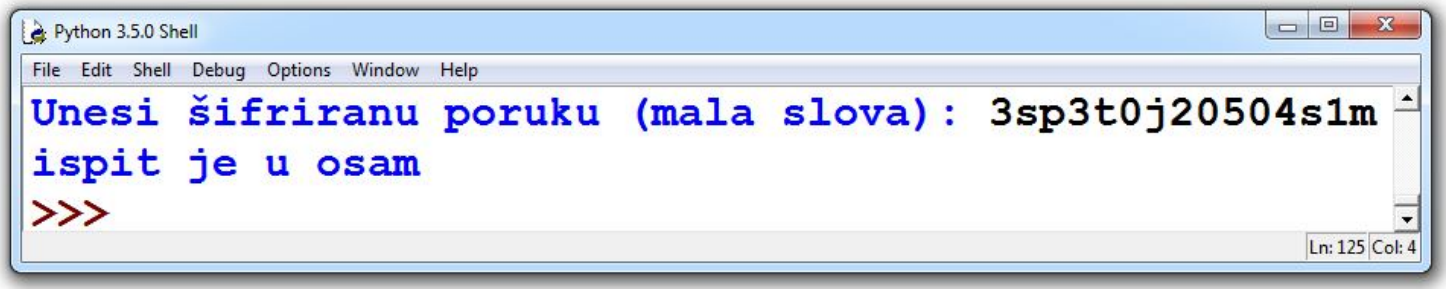
Zadatak: Šifriranje 1 - rješenje

- Ili kraće:

```
poruka = input('Unesi šifriranu poruku (mala
                        slova): ')
zamj = ' aeiou'
for i in range(0, 6):
    poruka = poruka.replace(str(i), zamj[i])

print(poruka)
```

sifra1.py



Zadatak: Šifriranje 2

- Vlaho misli da je smislio bolji način šifriranja. On poruku šifrira tako da svaki znak zamijeni znakom koji ima dekadski kod za N veći (N je jednoznamenkasti prirodni broj).
- Kako bi Aniti olakšao dešifriranje poruke, Vlaho broj N šalje kao prvi znak šifrirane poruke.

- Dešifrirati Vlahovu poruku!

Zadatak: Šifriranje 2

- Primjer:

poruka

```
4T}xlsr
```

dekadski kod svakog znaka pomaknut je za **+4** mjesta

znak '**T**' treba zamijeniti znakom koji ima za **4** niži dekadski kod

Ulaz	**Ulaz**
4T}xlsr	**6Jutkyo&qtpom{'**
Izlaz	**Izlaz**
Python	**Donesi knjigu!**

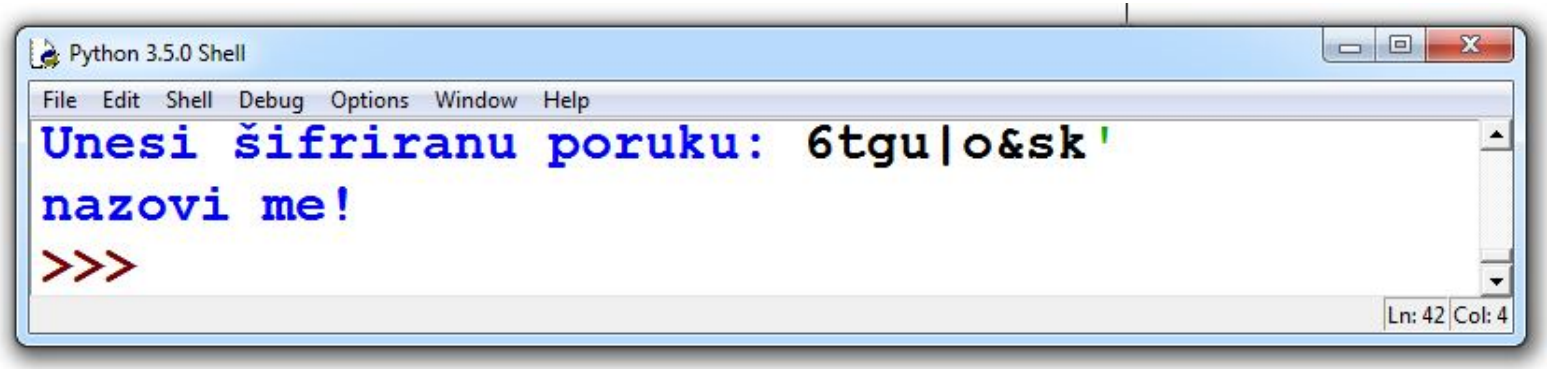
Zadatak: Šifriranje 2 - rješenje

```
poruka = input('Unesi šifriranu poruku: ')
poruka2 = ''
pomak = int(poruka1[0])

for i in range(1, len(poruka1)):
    poruka2 = poruka2 + chr(ord(poruka1[i])-pomak)

print(poruka2)
```

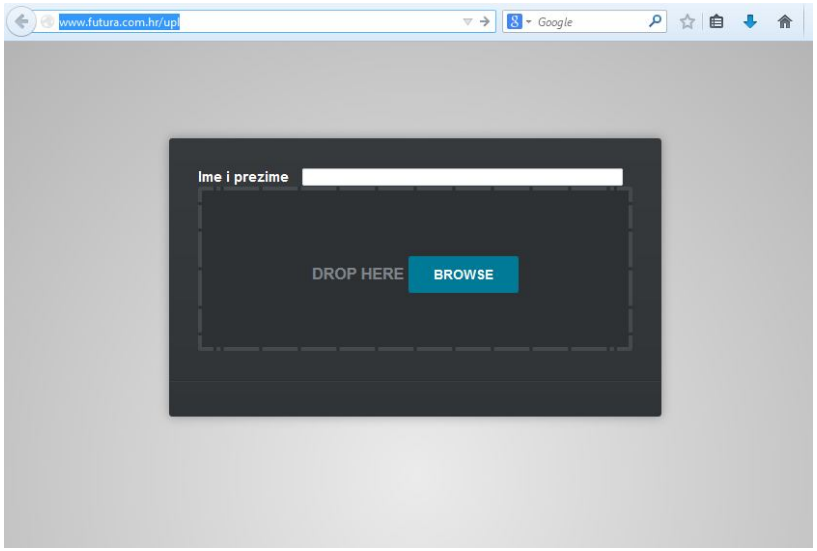
sifra2.py



Slanje programa na natjecanju?

- Kad idući put bude kolo **Lige programiranja**, bit će potrebno poslati (***upload***) programski kod riješenih zadataka.
- Link za slanje programa:

  www.futura.com.hr/upl



Slanje programa na natjecanju?

Ne zaboravite!

- Za 15 dana – **u subotu 19.12.2015.** – **2. kolo Lige programiranja**
- **5./6. razredi** - početak **9:00**
- **7./8. razredi** - početak **10:30**
- **3** zadatka rješavate **75** minuta
- nemojte kasniti!