INFORMATIČKI KLUB

FUTURA

LIGA PROGRAMIRANJA #4

python

LIGA PROGRAMIRANJA U PYTHONU ZA OSNOVNE ŠKOLE – 4. RADIONICA

Tomo Sjekavica, Mario Miličević *Informatički klub FUTURA*

Dubrovnik, 17. veljače 2018.

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

- **17.02.2018. – 4. radionica programiranja**

- **03.03.2018. – 4. kolo Lige programiranja, podjela nagrada**

- Web stranica Lige programiranja:

www.futura.com.hr/liga-programiranja-u-pythonu-2017-2018

Ponavljanje: programske petlje

- Ponavljanje bloka naredbi određeni broj puta:

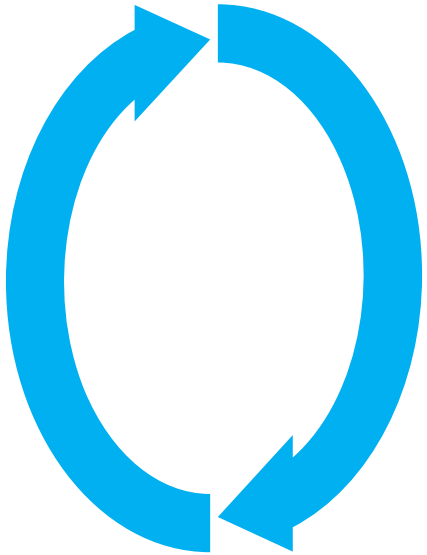
```
...
za i := 1 do n činiti
      naredba_1
      ...
      naredba_z
...
```

- `for` petlja

- Uvjetno ponavljanje bloka naredbi:

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

- `while` petlja



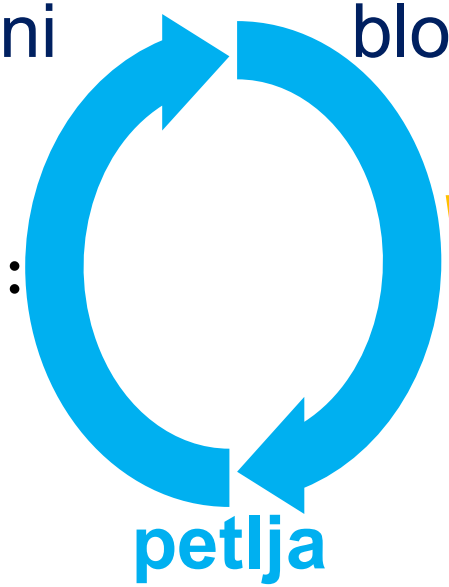
programska petlja

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**.



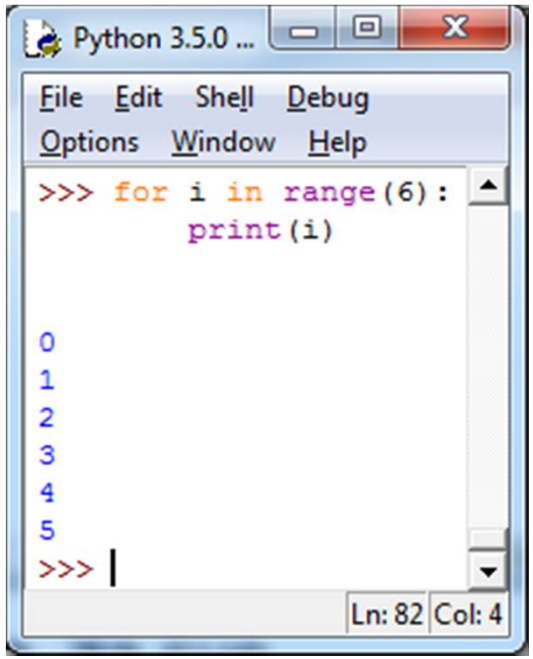
- Uvjetno ponavljanje bloka naredbi:

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

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

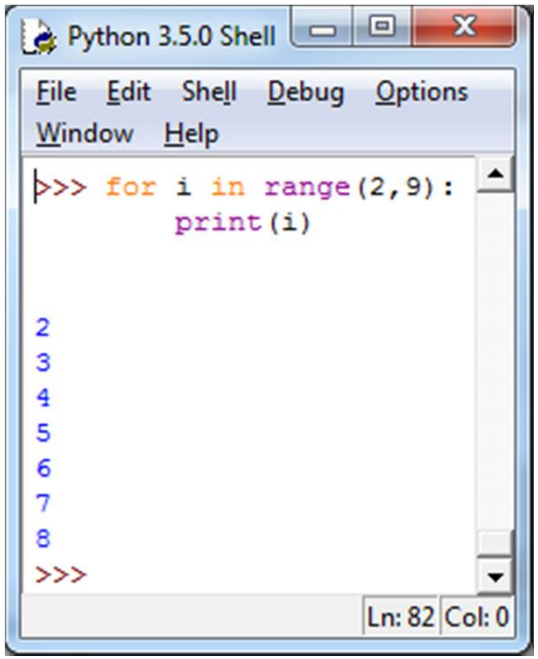
Ponavljanje: vrste **for** petlji

- `range(n)`



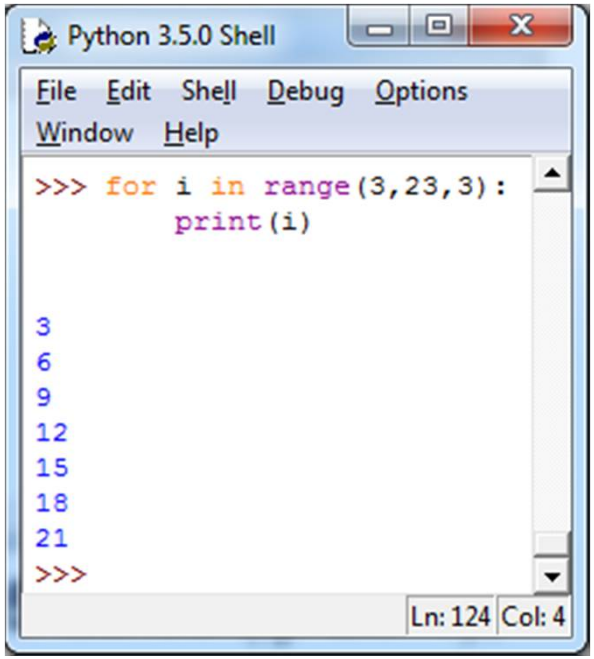
- Blok naredbi se izvodi **6 puta**, za vrijednosti **i** od **0** do **5**.

- `range(n,m)`



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

- `range(n,m,k)`



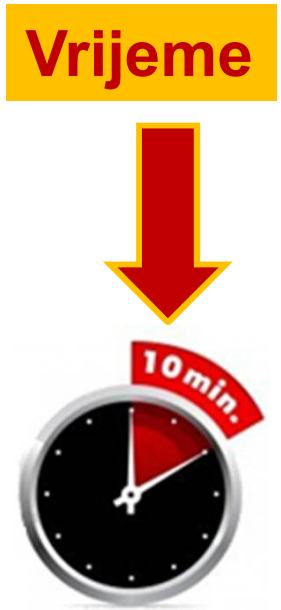
- Varijabla **i** mijenja vrijednost od **3** do **22** (tj. 23-1), s korakom **3**.

Zadatak: Trošenje

- Lukre je radila preko ljeta i uštedjela **3500 kuna**. Odlučila je da će, kad počne škola, za te kune svaki dan nešto sebi kupiti, i to tako da će ovisno o rednom broju dana svaki dan izračunati koliko će potrošiti:
  - ako je redni broj određenog dana neparan, Lukrecija će potrošiti kuna točno onoliko koliki je redni broj tog dana,
  - ako je redni broj određenog dana paran, Lukrecija će potrošiti 10 kuna.
- Koliko dana Lukre može trošiti kune na opisani način? Koliko ostane nepotrošemih kuna?

Zadatak: Trošenje

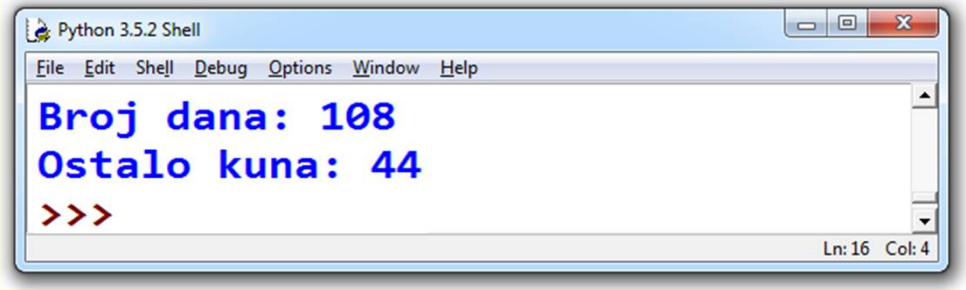
- Primjer:
  - 1. dan - Lukre će potrošiti 1 kn
  - 2. dan - Lukre će potrošiti 10 kn
  - 3. dan - Lukre će potrošiti 3 kn
  - 4. dan - Lukre će potrošiti 10 kn
  - 5. dan - Lukre će potrošiti 5 kn
  - 6. dan - Lukre će potrošiti 10 kn
  - itd. - sve dok neki dan više ne bude kuna za trošiti na opisani način



Zadatak: Trošenje - rješenje

```
sted = 3500
dan = 1
kraj = False
while not kraj:
    if dan % 2 == 0:
        potr = 10
    else:
        potr = dan
    if sted - potr >= 0:
        sted = sted - potr
        dan = dan + 1
    else:
        kraj = True
print('Broj dana:', dan-1)
print('Ostalo kuna:', sted)
```

pomoćna varijabla za izlaz iz petlje



trosenje.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
```

Metoda 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
```

Metoda 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

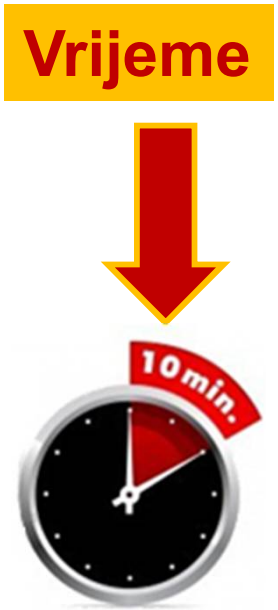
Zadatak: Štoperica

- Luko voli trčati i još više mjeriti koliko je vremena trčao. Međutim, nema pravu štopericu, već zabilježi koliko mu je vremena trebalo za svaki krug, i onda to na kraju zbroji pomoću programa.
- Ulazni podaci:
  - broj istrčanih krugova
  - za svaki krug: vrijeme potrebno za taj krug - u formatu **MM:SS**
- Izlazni podatak:
  - koliko je ukupno Luko trčao, u formatu **HH:MM:SS**

Zadatak: Štoperica

- Testni podaci:

Ulaz	Ulaz	Ulaz
3 12:30 15:10 20:00	4 14:40 20:20 15:55 18:05	7 15:50 20:25 18:20 22:30 25:20 24:50 35:10
Izlaz	**Izlaz**	**Izlaz**
0:47:40	1:9:0	2:42:25



Zadatak: Štoperica - rješenje

```
N = int(input('Broj krugova: '))
uk_sek = 0
for i in range(N):
    vr = input('Vrijeme za krug: ')
    mm, ss = vr.split(':')
    uk_sek = uk_sek + int(mm)*60 + int(ss)

S = uk_sek % 60
Mpom = uk_sek // 60
M = Mpom % 60
H = Mpom // 60

print('Ukupno vrijeme: ', H,':', M, ':', S, sep='')
```

stoperica.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 = 2018
>>> radionica = 'Liga programiranja u Pythonu'
>>> print('Radionice', radionica, godina, 'OŠ')
Radionice Liga programiranja u Pythonu 2018 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 = 'Ivo'
>>> prez = 'Ivić'
>>> ucenik = ime + prez
>>> print(ucenik)
IvoIvić
>>>
>>> ucenik = ime + ' ' + prez
>>> print(ucenik)
Ivo Ivić
>>>
```

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

```
>>> fut = 'Inform.klub FUTURA'
>>> 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

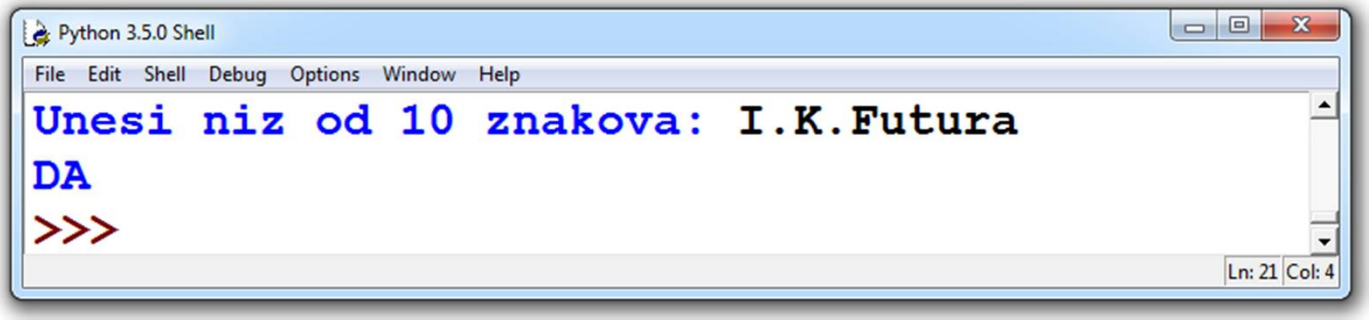
Zadatak: Nađi podniz - rješenje 5./6.

```
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

- Marija voli stvarati nove riječi. To radi tako da Martina pita da joj kaže dvije riječi, a onda napravi novu tako da iz prve riječi izbaci sva slova koja se javljaju u drugoj riječi.
- Primjer:

Zadatak: Nove riječi

- Ulazni podaci:
  - dvije riječi u istom retku odvojene prazninom
- Izlazni podatak:
  - nova riječ
- Testni podaci:

Ulaz	Ulaz
programiranje igra	Dubrovnik Divona
Izlaz	Izlaz
pomnje	ubrk

Vrijeme

Zadatak: Nove riječi - rješenje

```
novo = ''

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

for i in rijec1:
    if i not in rijec2:
        novo = novo + 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: Poruka

- Irena je tipkala poruku za Mateu na računalu u školi bez da je gledala rezultat na monitoru, ali nije znala da je netko zamijenio oznake na tipkama **A** i **S**, odnosno **O** i **P**.
- Npr. "**Program**" je napisala kao "**Orpgrsm**""
- Pomoći Matei da "dešifriranu" poruku ispiše velikim slovima!

Ulaz	**Izlaz**
Poet npvs vrats orpgrsms	**OPET NOVA VRSTA PROGRAMA**
Ulaz	**Izlaz**
Aosai me pd OSVS	**SPASI ME OD PAVA**

Zadatak: Poruka - rješenje

```
poruka = input('Unesi poruku: ')
poruka = poruka.upper()

poruka = poruka.replace('A', 'x')
poruka = poruka.replace('S', 'A')
poruka = poruka.replace('x', 'S')

poruka = poruka.replace('O', 'x')
poruka = poruka.replace('P', 'O')
poruka = poruka.replace('x', 'P')

print(poruka)
```

zašto se ne može 'A' odmah zamijeniti s 'S', nego se mora koristiti pomoćni znak 'x'?

poruka.py

Zadatak: Veliki brojevi

- Damiru se sviđa što u Pythonu cijelim brojevima nije ograničen broj znamenki, pa smišlja igre s velikim brojevima. Tako u jednoj igri prvo napiše veliki prirodni broj, a onda traži koji se od dvoznamenkastih brojeva najviše puta ponavlja u zadanom broju.
- Napisati program koji će pomoći Damiru!

Ulaz	**Ulaz**
456985645785698547485 06	**828976337533876339733933**
Izlaz	**Izlaz**
85	**33**

Zadatak: Veliki brojevi - rješenje

```
poruka = input('Unesi broj: ')
naj_br = 0
naj_dvoz = ''

for i in range(10, 100):
    dvoz = str(i)
    br = veliki.count(dvoz)
    if br > naj_br:
        naj_br = br
        naj_dvoz = dvoz

print(naj_dvoz)
```

brojevi se u ovom zadatku tretiraju kao stringovi, kako bi se mogla koristiti metoda count()!

veliki_brojevi.py

Ne zaboravite!

- Za 15 dana – **u subotu 3.3.2018.** – **4. (finalno) kolo Lige programiranja**
- **5./6.** i **7./8. razredi** - početak **10:00**
- **3** zadatka rješavate **75** minuta
- nemojte kasniti!