INFORMATIČKI KLUB

FUTURA

LIGA PROGRAMIRANJA #4

python

LIGA PROGRAMIRANJA U PYTHONU ZA OSNOVNE ŠKOLE – 2. RADIONICA

Tomo Sjekavica, Mario Miličević *Informatički klub FUTURA*

Dubrovnik, 2. prosinca 2017.

Creative Commons

- **slobodno smijete:**
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **remiksirati** — prerađivati djelo
- **pod slijedećim uvjetima:**
  - **imenovanje**. Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno**. Ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima**. Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

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

- 02.12.2017. – **2. radionica**
- 16.12.2017. – **2. kolo Lige programiranja**
- *Božić i Nova godina*
- 20.01.2018. – 3. radionica programiranja

- Web stranica Lige programiranja:

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

Programski jezik Python

- www.python.org
- Open source program
- Besplatni program
- Jednostavna sintaksa
- Autor: Guido van Rossum - kraj 1989. godine
- **Python 1.0** – siječanj 1994. godine
- **Python 2.0** – listopad 2000. godine
- **Python 3.0** – prosinac 2008. godine
- Zadnje verzije: **Python 2.7.14** i **Python 3.6.3**

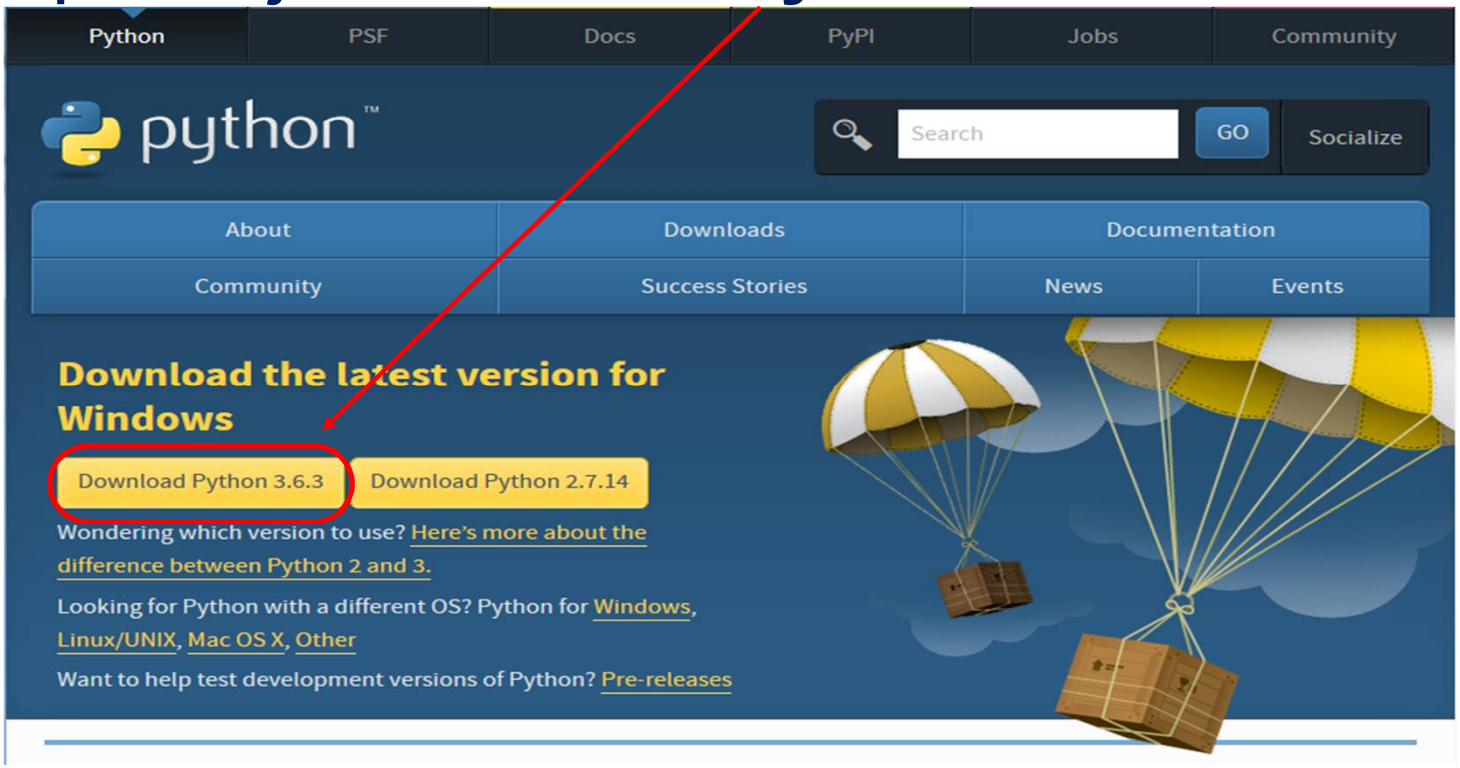
Primjena Pythona

- Web programiranje:
  - Django, Pyramid, Bottle, Tornado, Flask, web2py
- Razvoj samostojeće programske potpore:
  - wxPython, tkInter, PyGtk, PyQt
- Znanost i numeričke simulacije:
  - SciPy, Pandas, Ipython
- Razvoj softvera:
  - Buildbot, Trac, Roundup, Scons, Apache Gump
- Administracija sustava:
  - Ansible, Salt, OpenStack

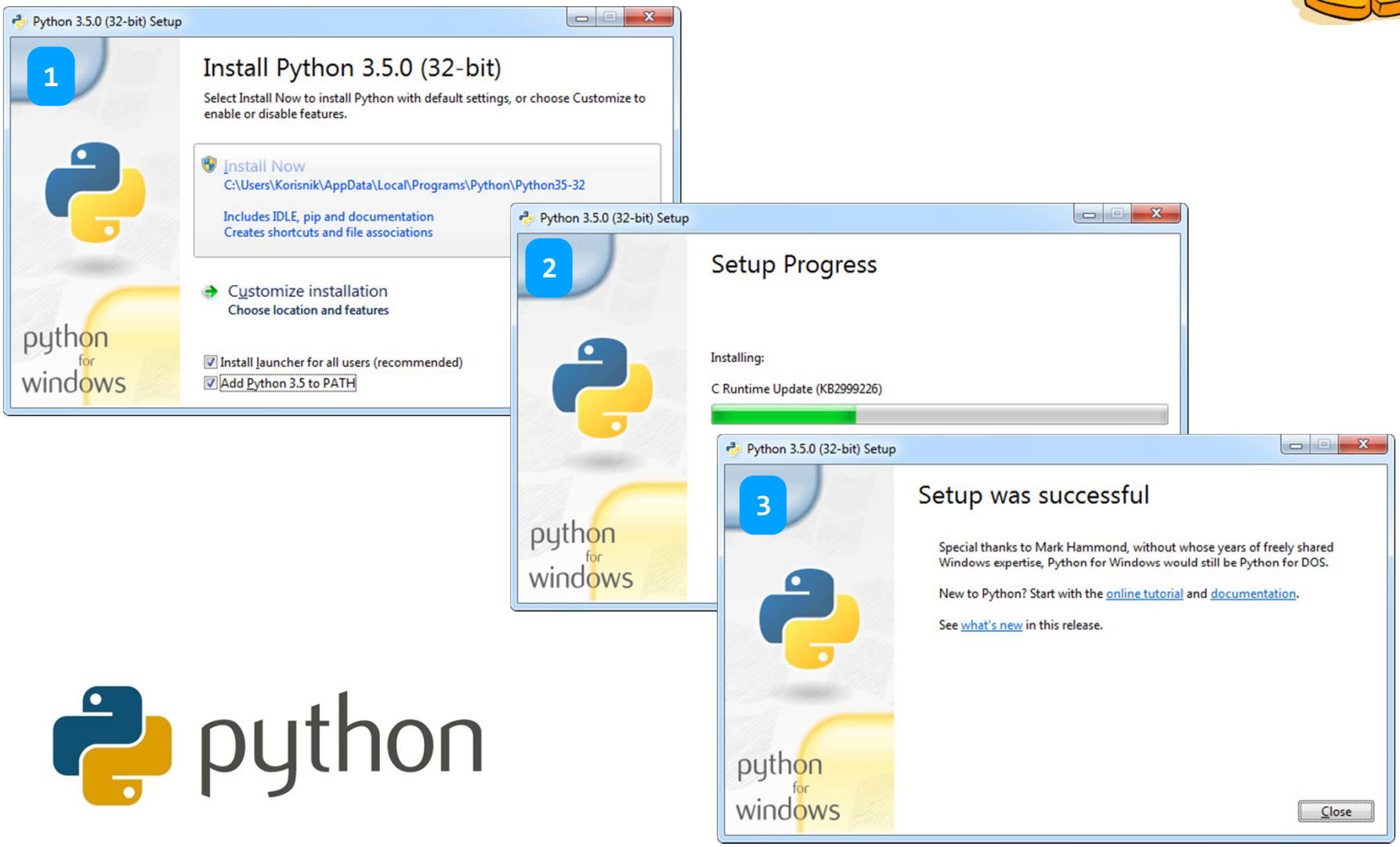
Tko sve koristi Python?

Instalacija Pythona

- **www.python.org/downloads**
- Koristiti ćemo zadnju verziju za Windows operacijski sustav – **Python 3.6.3**



Instalacija Pythona

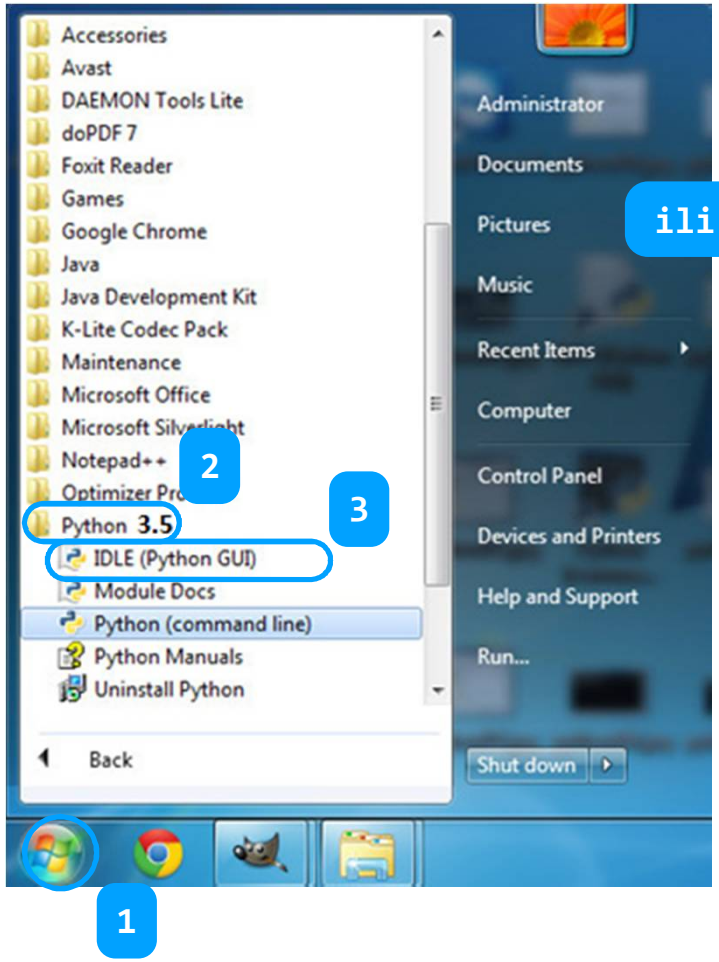


python

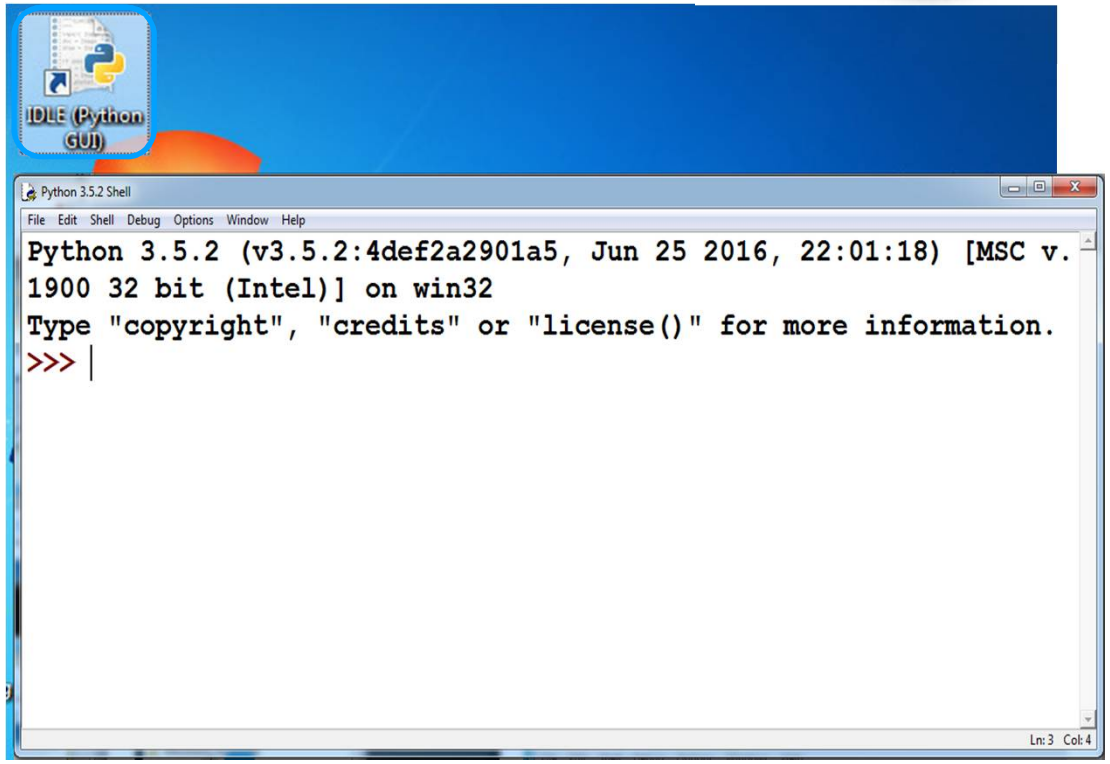
Pokretanje Python IDLE

- Integrated **D**eve**L**opment **E**nviroment

Eric Idle – komičar iz Monty Pythona



ili



Osnovni tipovi podataka u Pythonu

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

Cijeli brojevi

- Primjeri cijelih brojeva

```
>>> 12
12
>>> 10000
10000
```

```
>>> -12
-12
>>> 
SyntaxError: invalid token
```

GREŠKA: kod Pythona ne možemo unositi vodeće nule, crvenom bojom je ispisana greška, a crvenom bojom pozadine je označen dio koda gdje se dogodila greška

Python unos ili rezultat neke naredbe ispisuje fontom plave boje, pa se lako može prepoznati što smo mi unijeli, a što je Python ispisao

- Nije ograničen broj znamenaka cijelog broja

```
>>> 12345678901234567890123456789012345678901234
12345678901234567890123456789012345678901234
```

Brojevi s pomičnom točkom

Primjeri brojeva s pomičnom točkom

```
>>> 3.2
3.2
>>> 0.00032
0.00032
>>> 1e15
1000000000000000.0
```

```
>>> -3.
-3.0
>>> .000032
3.2e-05
>>> 1e16
1e+16
```

```
>>> 0.32
0.32
>>> 1e2
100.0
```

brojevi s pomičnom točkom imaju granicu

Donja granica brojeva s pomičnom točkom

```
>>> 1.2345678901234567e-323
1e-323
```

```
>>> 1.2345678901234567e-324
0.0
```

Gornja granica brojeva s pomičnom točkom

```
>>> 1.2345678901234567e308
1.2345678901234567e+308
```

```
>>> 1.2345678901234567e308
inf
```

Aritmetički operatori

zbrajanje	+
oduzimanje	-
množenje	*
dijeljenje	/
cjelobrojno dijeljenje	//
modulo (ostatak od dijeljenja)	%
potenciranje	**

- Prvenstvo pri izvođenju ima potenciranje, pa nakon toga množenje, dijeljenje, cjelobrojno dijeljenje i modulo, te na kraju zbrajanje i oduzimanje

Aritmetički izrazi

Jednostavni aritmetički izrazi

```
>>> 2 + 3
5
>>> 2 / 3
0.6666666666666666
>>> 2 // 3
0
```

```
>>> 2 - 3
-1
>>> 2 % 3
2
>>> 2+
SyntaxError:invalid syntax
```

```
>>> 2 * 3
6
>>> 2 ** 3
8
```

Složeni aritmetički izrazi

```
>>> 2 + 2 * 3
8
```

```
>>> (2 + 2) * 3 + 2 / 3
 12.666666666666666
```

Zadnja izračunata vrijednost

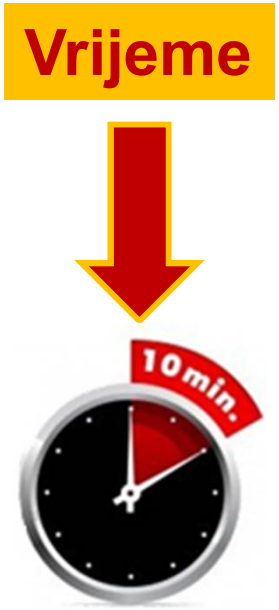
```
>>> 2 / 3
0.6666666666666666
>>> _ * 2
1.3333333333333333
```

GREŠKA: ako ne napišemo dobro aritmetički izraz prikazati će nam se greška

Zadatak: Razlomak

- Napišite jednu naredbu koja će izračunati vrijednost sljedećeg razlomka:

$$\frac{\frac{2}{2+3}+2\left(3+\frac{1}{2}\right)-4.4}{\left(\frac{7}{2}+3.5\right)\frac{1}{5}+0.6}$$



- Rješenje:

```
>>>(2/(2+3) + 2*(3+1/2) - 4.4) / ((7/2+3.5)*1/5 + 0.6)
1.5
```

brojnik razlomka unutar zagrada

nazivnik razlomka unutar zagrada

Znakovni nizovi

- Jednostruki ili dvostruki navodnici

```
>>> 'Python'
'Python'
>>> "Python"
'Python'
```

nizovi znakova su označeni fontom zelene boje

- Ispis dvostrukih navodnika u nizu znakova

```
>>> 'Radionica "Python" za \"osnovne škole\"'
'Radionica "Python" za "osnovne škole"'
```

- Ispis jednostrukih navodnika u nizu znakova

```
>>> "Radionica 'Python' za \'osnovne škole\'"
"Radionica 'Python' za 'osnovne škole'"
```

- Preporuka: korištenje jednostrukih navodnika za znakovne nizove

Znakovni nizovi – funkcija print

- Funkcija je definirani skup naredbi
- Opći oblik funkcije u Pythonu:

```
naziv_funkcije(parametar1, parametar2, … , parametarN)
```

- Funkcija može primiti 0, 1 ili više parametara
- Funkcija **print**

```
>>> print()

>>> print('Radionica', 'Python', 2017)
Radionica Python 2017
```

```
>>> print('Python')
Python
```

funkcije print kao parametre može primati različite tipove podataka

standardne Python funkcije su označene fontom ljubičaste boje

Znakovni nizovi – funkcija print

- Aritmetički izrazi u ispisu

```
>>> print('Zbroj brojeva', 4, 'i', 3, 'je:', 4 + 3)
Zbroj brojeva 4 i 3 je: 7
```

- Ispis lijevo nakošene crte \

```
>>> print('Nakošena crta - \\.')
Nakošena crta - \.
```

- Tabulator - **\t**

```
>>> print('Korištenje\ttabulatora\tu\tPythonu.')
Korištenje	tabulatora	u	Pythonu.
```

- Prelazak u novi red pri ispisu - **\n**

```
>>> print('Prelazak\nu novi red u Pythonu.')
Prelazak
u novi red u Pythonu.
```

Varijable

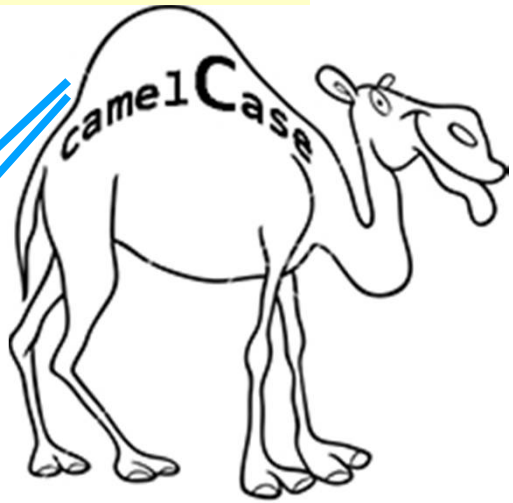
- Varijabla je memorijska lokacija kojoj pristupamo preko njenog naziva, a na njoj je zapisana vrijednost koja se može mijenjati
- Varijabla ima naziv i vrijednost

```
>>> varijabla = 10
>>>
```

vrijednost varijable

naziv varijable

loši nazivi	dobri nazivi
aaaaa	brojac
abcdefgh	ime_prezime
ahauifhasfuhsaiu	godinaRodjenja
hfjhds3u4444	imeNajPrijatelja



Varijable

- Pravila za imenovanje varijabli:
  - Naziv varijable može sadržavati slova, brojeve i podvlake
  - Naziv varijable ne smije počinjati sa znamenkom
  - Naziv varijable ne smiju biti ključne riječi za koje su rezervirani nazivi, kao što **`bool`**, **`True`**, **`False`**, …
  - Naziv varijable smije sadržavati naše znakove (čćžšđČĆŽŠĐ), ali se to **nikako ne preporuča**
  - Python razlikuje velika i mala slova, pa su **`x`** i **`X`** dvije različite varijable

Pridruživanje vrijednosti varijablama

- Znak pridruživanja **=**

```
>>> x = 20
>>> print(x)
20
>>> x = x + 10
>>> print('x =', x)
x = 30
```

```
>>> y = -3.2
>>> print(y)
-3.2
>>> y = y  * 2
>>> print('y =', y)
y = -6.4
```

- U varijable se mogu spremiti i znakovni nizovi

```
>>> python = 'Radionica progr. jezika Python'
>>> print(python)
Radionica progr. jezika Python
```

Zadatak: Izlet

- Učenici su u vrtu škole ubrali 172 kg jabuka. Jabuke je potrebno prenijeti u školu, a na raspolaganju su velike košare u koje stane 30 kg, kutije u koje stane 12 kg i kante u koje stane 3 kg.
- Ako prvo treba koristiti veće posude, koliko će najmanje trebati pojedinih vrsta posuda za prenijeti sve jabuke u školu?

Zadatak: Izlet - rješenje

```
>>> kolJab = 172
>>> brKos = kolJab // 30
>>> ost1 = kolJab % 30
>>> brkut = ost1 // 12
>>> ost2 = ost1 % 12
>>> brKan = ost2 // 3
>>> ost2 % 3
1
>>> brKan = brKan + 1
>>> print('Broj košara:', brKos )
Broj košara: 5
>>> print('Broj kutija:', brKut)
Broj kutija: 1
>>> print('Broj kanti:', brKan)
Broj kanti: 4
```

varijabla za količinu jabuka i pridruživanje vrijednosti 172

izračun broja košara

koliko jabuka ostaje

izračun broja kutija

koliko jabuka ostaje

izračun broja kanti

koliko jabuka ostaje

pošto ostaje još 1kg jabuka treba dodati još jednu kantu

Zadatak: Godine, mjeseci, ...

- Napišite niz naredbi koje će **20.000 sati** pretvoriti u godine, mjesece, dane i sate. Pretpostavka je da svaki mjesec ima 30 dana. Dobivene vrijednosti spremite u varijable. Ispišite dobiveni broj godina, mjeseci i dana jednom `print` funkcijom.

Vrijeme

iz broja sati mogu dobiti dane tako da ih cjelobrojno podijelim s ukupnim brojem sati u jednom danu

ostatak sati (kad se izračuna broj dana) mogu dobiti tako da za ukupni broj sati izračunam ostatak dijeljenja s brojem sati u danu

Zadatak: Godine, ... - rješenje

```
>>> sati = 20000
>>> dani = sati // 24
>>> sati = sati % 24
>>> mjes = dani // 30
>>> dani = dani % 30
>>> god = mjes // 12
>>> mjes = mjes % 12
>>> print('Godina:', god,
      ' Mjeseci:', mjes,
      ' Dana:', dani,
      ' Sati:', sati)
```

deklaracija varijable sati i pridruživanje vrijednosti 20000

izračun broja dana

izračun broja preostalih sati

izračun broja mjeseci

izračun broja preostalih dana

ispis broja godina, mjeseci, dana i sati

```
Godina: 2  Mjeseci: 3  Dana: 23  Sati: 8
```

Korištenje pomoći

Funkcija `help`

```
>>> help()

Welcome to Python 3.6's help utility!

If this is your first time using Python, you should definitely check
ou                                                    on.org/3.6/tutorial/

En                                                    get help on
writing Python programs and using Python modules.  To quit this help
utility and return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics,
type "modules", "keywords", "symbols", or "topics".  Each module
also comes with a one-line summary of what it does; to list the
modules whose name or summary contain a given string such as "spam",
type "modules spam".

help>
```

naredba za izlazak iz pomoći

za popis svih modula, ključnih riječi, simbola i tema treba unijeti riječ modules, keywords, symbols ili topics

unos naziva modula, ključne riječi, simbola ili naziva funkcije za koju nam treba pomoć

Korištenje pomoći

- Popis svih ključnih riječi Pythona

```
help> keywords

Here is a list of the Python keywords.  Enter any keyword to get
more help.

False               def                 if                  raise
None                del                 import              return
True                elif                in                  try
and                 else                is                  while
as                  except              lambda              with
assert              finally             nonlocal            yield
break               for                 not
class               from                or
continue            global              pass

help> quit
```

izlazak iz pomoći

Korištenje pomoći

- Drugi način: **help(naziv_funkcije)**

```
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout,
flush=False)

    Prints the values to a stream, or to s
    Optional keyword arguments:
    file:  a file-like object (s
sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.

>>>
```

pomoć za funkciju print

standardno na kraju je prelazak u novi red, ali korisnik može postaviti neki niz znakova

standardno između dvije vrijednosti je razmak ili korisnik može postaviti neki niz znakova

vrijednosti koje ispisuje print funkcija odvojene zarezom

Funkcija print

- Ispis niza vrijednosti

```
>>> godina = 2017
>>> radionica = 'Python'
>>> print('Radionica', radionica, godina, 'OŠ')
Radionica Python 2017 OŠ
```

- Promjena standardnog separatora

```
>>> print('Radionica','Python','osnovne škole', sep='#')
Radionica#Python#osnovne škole
```

- Promjena standardnog kraja ispisa

```
>>> print('Radionica','Python','osnovne škole', end='#')
Radionica Python osnovne škole#
```

Unos s tipkovnice

- Funkcija **input**
- Proučite pomoć za funkciju **input**

```
>>> help(input)
Help on built-in function input in module builtins:

input(...)
    input([prompt]) -> string

    Read a string from standard input.  The trailing newline is
stripped.
    If the user hits EOF (Unix: Ctl-D, Windows: Ctl-Z+Return), raise
EOFError.
    On Unix, GNU readline is used if enabled.  The prompt string, if
given,
    is printed without a trailing newline before reading.

>>>
```

funkcija input sve što se unese s tipkovnice sprema kao znakovni niz

Unos s tipkovnice

- Pomoću funkcije **input** unesite vaše ime s tipkovnice, spremite ga u varijablu **ime**, te nakon toga ispišite vrijednost varijable **ime**.

```
>>> ime = input('Unesi svoje ime: ')
Unesite vaše ime: Futurist
>>> print('Uneseno ime je: ', ime)
Uneseno ime je:  Futurist
```

Unos s tipkovnice

Primjer funkcije `input` s cijelim brojem

```
>>> broj = input('Unesi cijeli broj: ')
Unesi cijeli broj: 10
>>> broj + 10
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    broj+10
TypeError: Can't convert 'int' object to str implicitly
```

GREŠKA: broj 10 unesen s tipkovnice je spremljen kao niz znakova

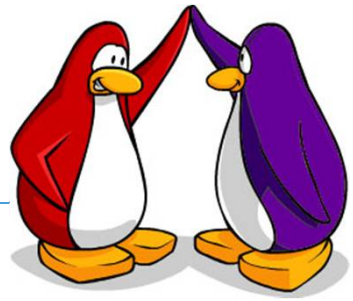
Funkcija `int` – pretvara u cijeli broj

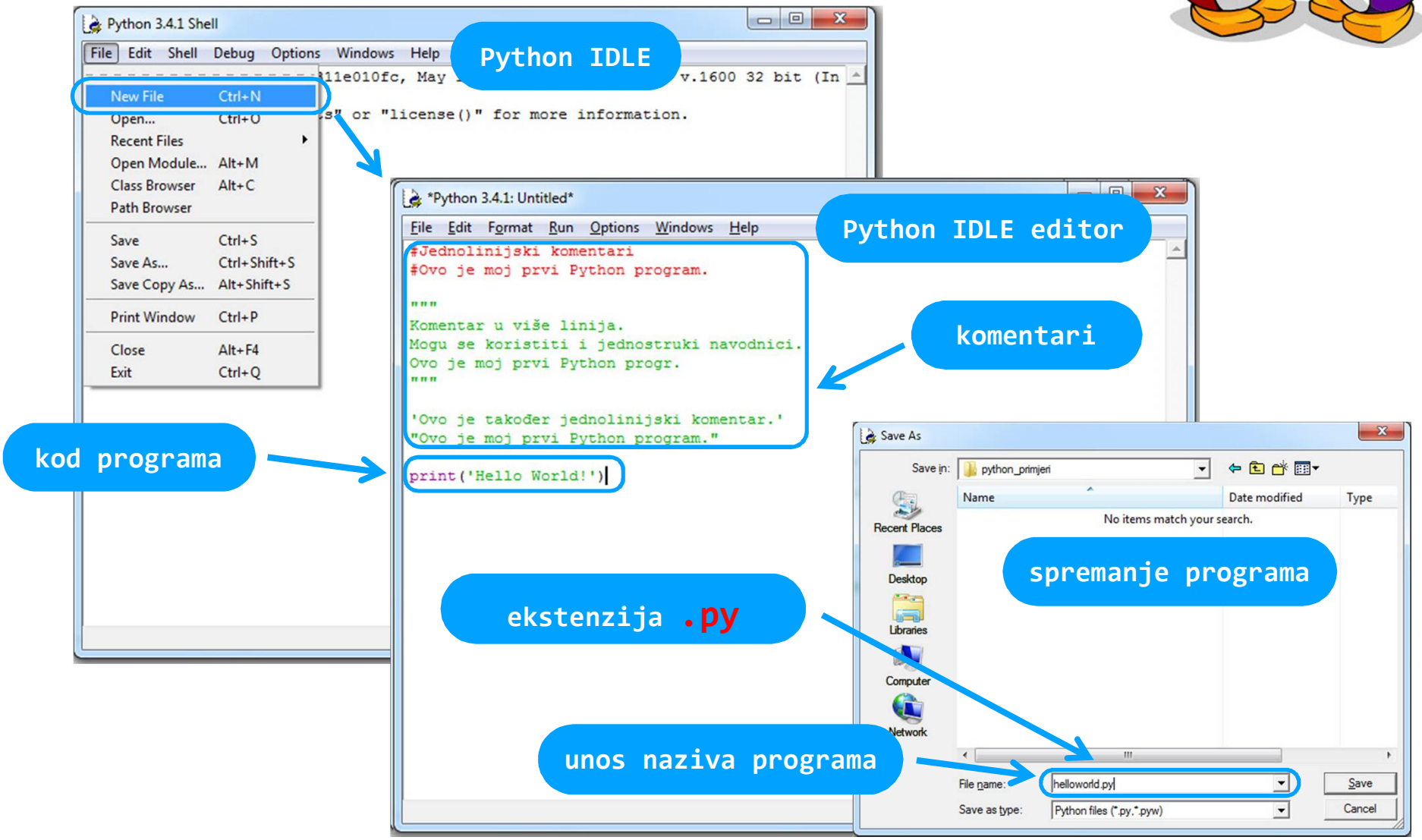
```
>>> broj = input('Unesi cijeli broj: ')
Unesi cijeli broj: 10
>>> broj = int(broj)
>>> broj + 10
20
```

Program

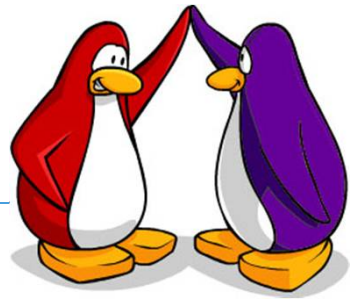
- Naredbe smo dosad unosili i odmah pokretali u Python IDLE-u
- Što će se dogoditi ako zatvorimo Python IDLE?
- Izgubili smo sve naredbe koje smo unosili
- Program je skup naredbi čijim se izvršenjem obavlja neki posao
- Naredbe možemo spremiti kao poseban program, pa taj program možemo naknadno ažurirati i pokretati

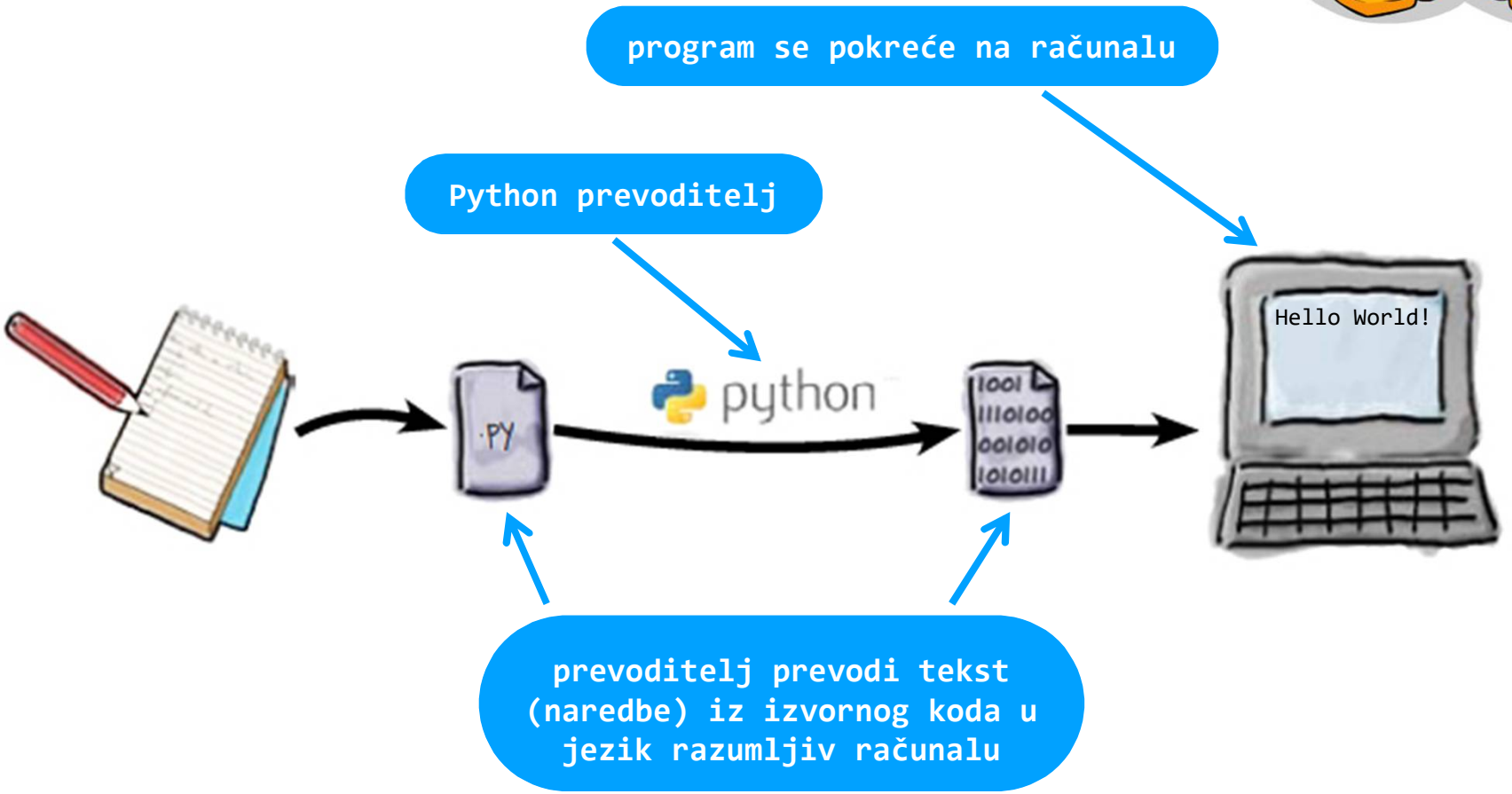
Prvi Python program





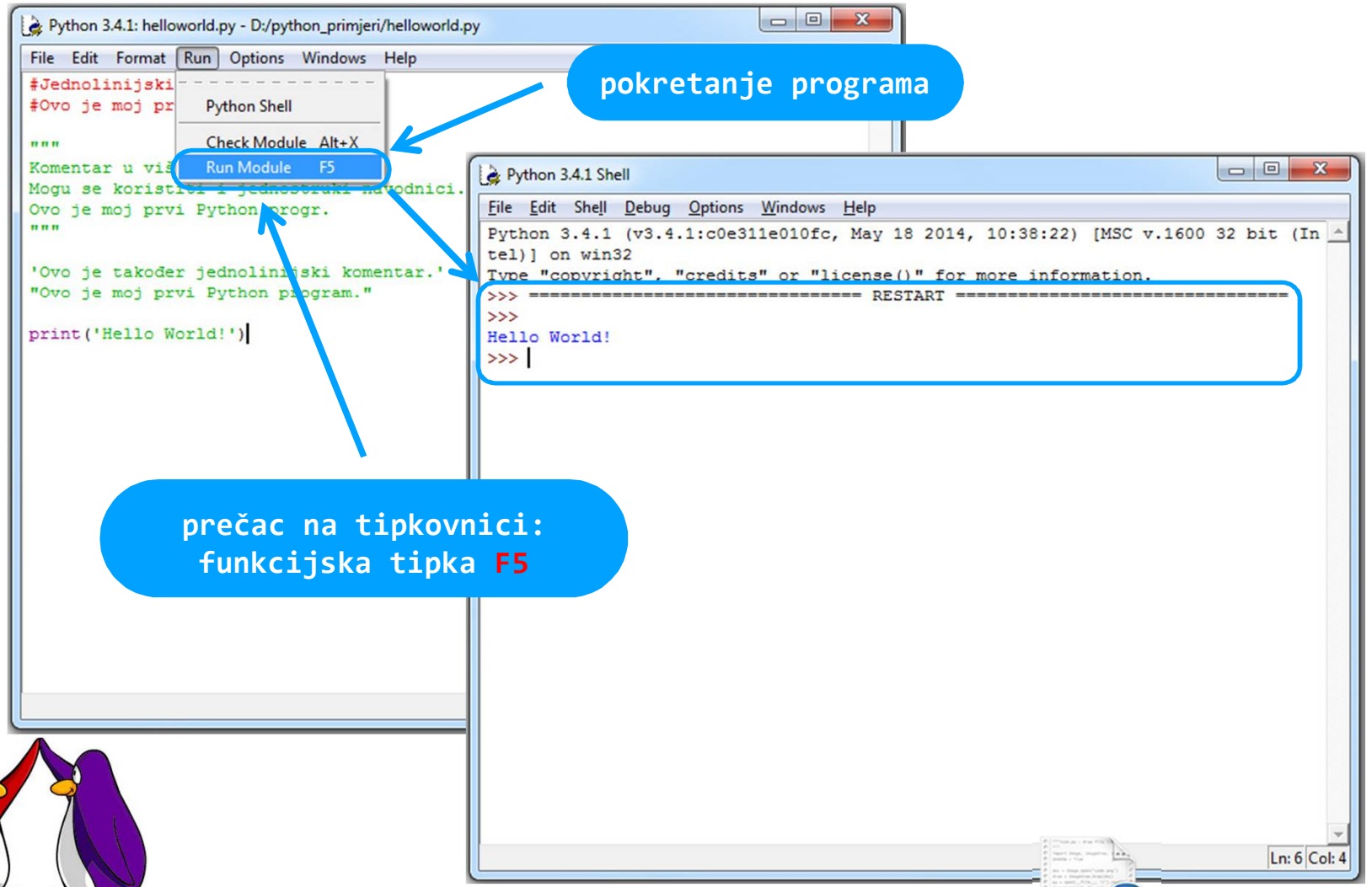
Prevođenje programa

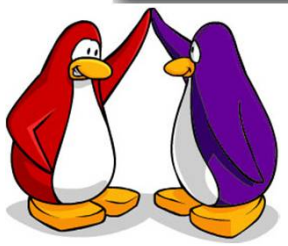




Prilagođeno iz knjige: P. Barry & D. Griffiths, Head First Programming, O'Reilly, 2009

Pokretanje programa u Python IDLE-u



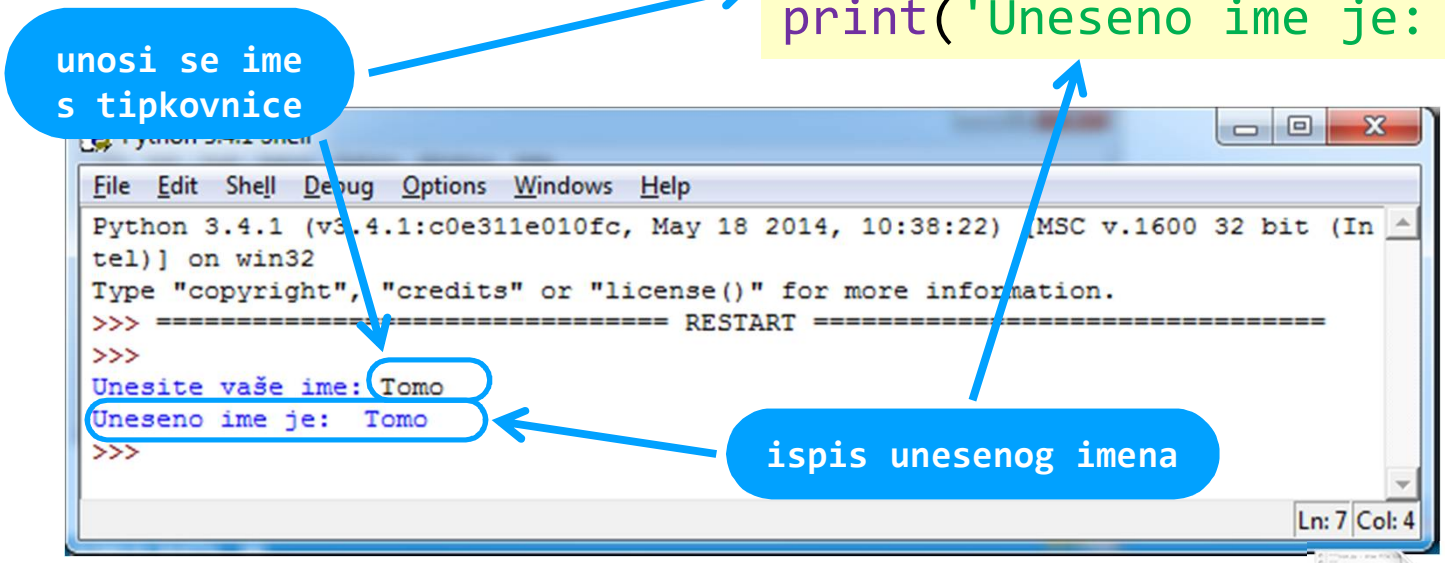


helloworld.py

Program: Unos imena

- Naredbe za unos imena i ispis imena na ekran iz primjera spremite u program **`unos_imena.py`**, te pokrenite program u Python IDLE-u.

```
ime = input('Unesite vaše ime: ')
print('Uneseno ime je: ', ime)
```

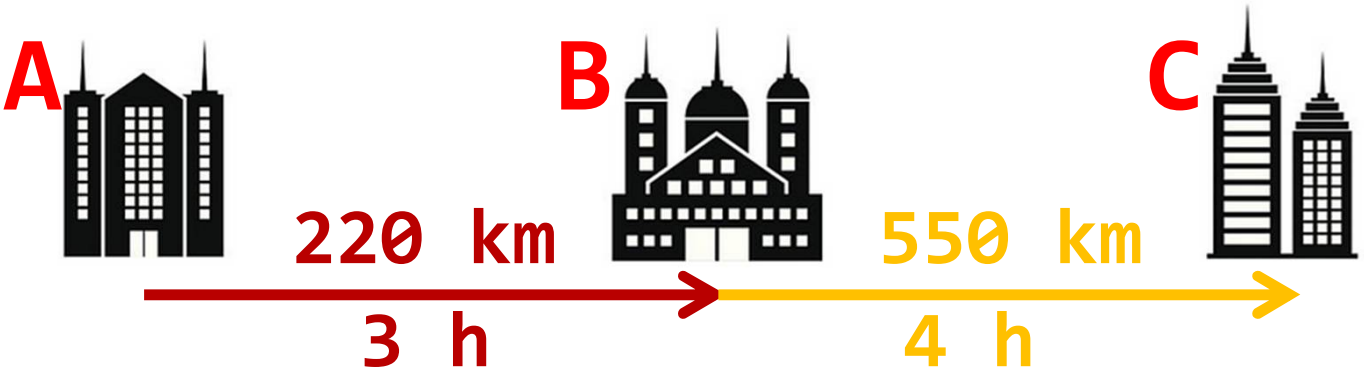


unos_imena.py

Zadatak: Prosječna brzina

- Napisati program kojim se unose:
  - udaljenost između gradova A i B (u **km**),
  - udaljenost između gradova B i C (u **km**),
  - vrijeme vožnje od grada A do grada B (u **h**)
  - vrijeme vožnje od grada B do grada C (u **h**)
- Treba izračunati prosječnu brzinu na putu iz grada A u grad C (preko grada B).

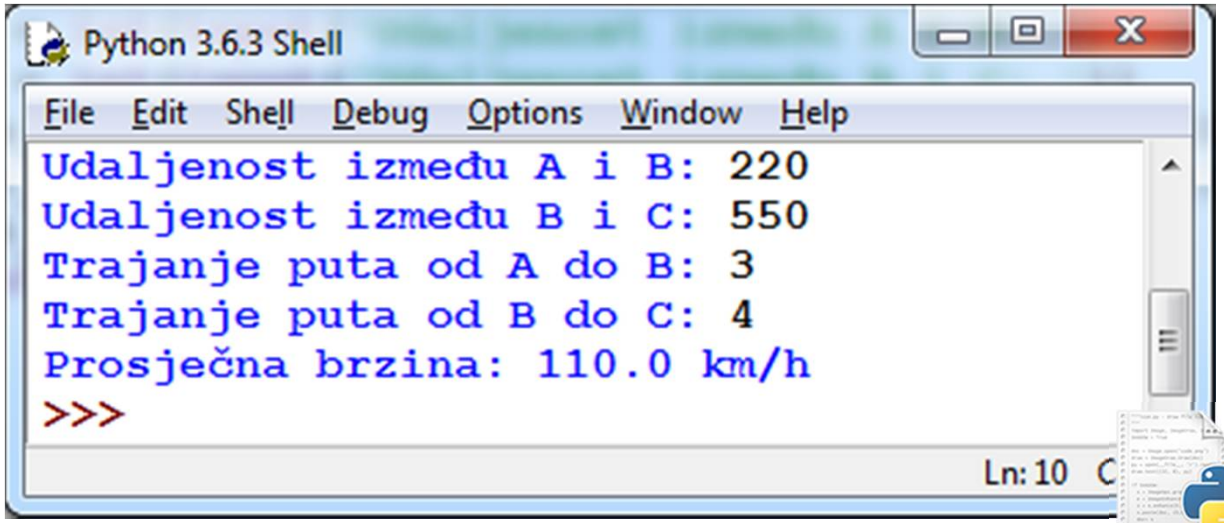
Vrijeme



PRIMJER

Zadatak: Prosječna brzina -rješenje

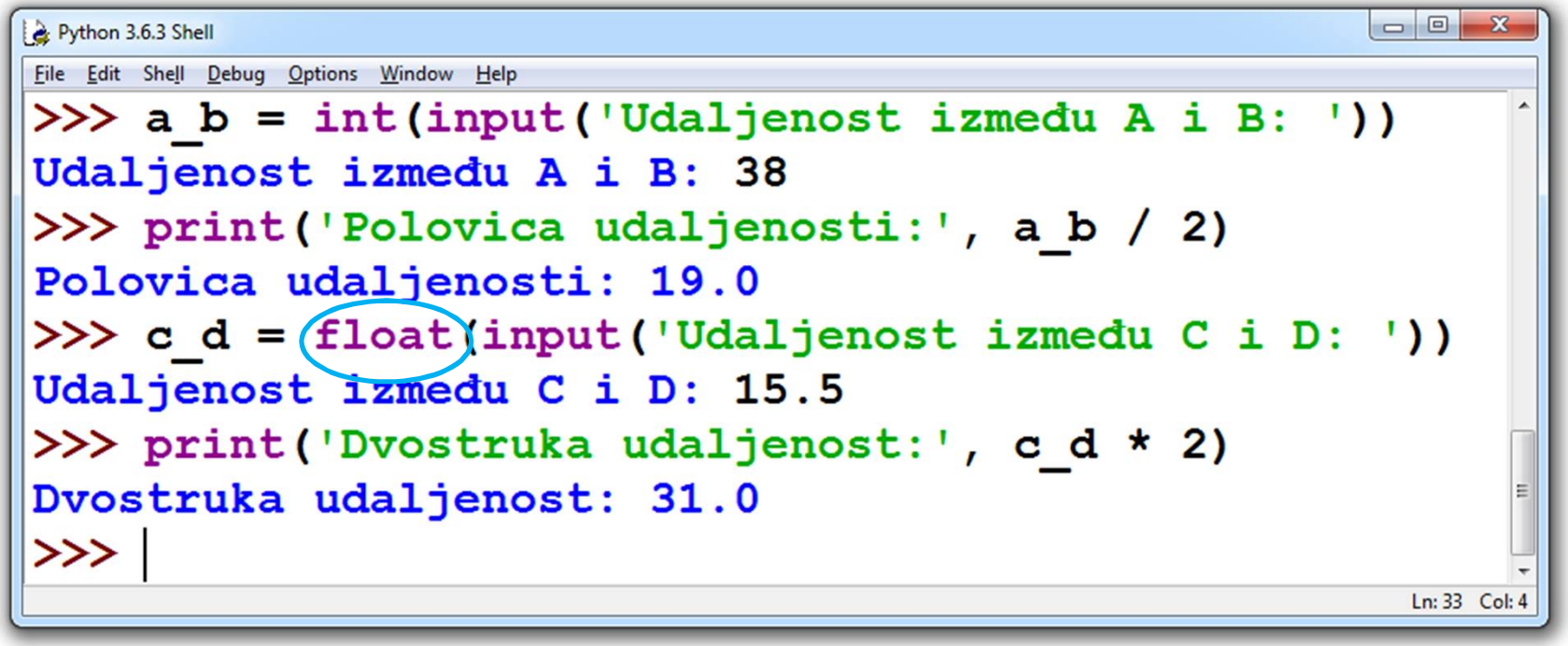
```
a_b = int(input('Udaljenost između A i B: '))
b_c = int(input('Udaljenost između B i C: '))
vr_a_b = int(input('Trajanje puta od A do B: '))
vr_b_c = int(input('Trajanje puta od B do C: '))
pros_brz = (a_b + b_c) / (vr_a_b + vr_b_c)
print('Prosječna brzina:', pros_brz, 'km/h')
```



prosjecnaBrzina.py

Unos decimalnih brojeva

- Što da udaljenost nije bila cijeli broj, već decimalni?



- Funkcija **float** pretvara znakovni niz u decimalni broj.

Relacijski operatori

veće od	**>**
manje od	**<**
veće od ili jednako	**>=**
manje od ili jednako	**<=**
jednako	**==**
nije jednako	**!=**

- Relacijski operatori uspoređuju dva operanda. Rezultat usporedbe ima vrijednosti ***True*** ili ***False*** (*Istina* ili *Laž*).

Relacijski operatori

- Provjeriti kako operatori djeluju:

```
>>> 3 > 2
True
>>> 3 < 2
False
>>> 3 >= 2
True
>>> 3 <= 2
False
>>> 3 == 2
False
>>> 3 != 2
True
```

```
>>> a = 2
>>> b = 7
>>> b > a
True
>>> b+1 == a*4
True
>>> b/a != b//a
True
>>> (a+b)**2 <=
10*a
False
```

```
>>> r1 = 'tri'
>>> r2 = 'pet'
>>> r1 > r2
True
>>> m = 'tri'
>>> n = 'Tri'
>>> m == n
False
>>> m > n
True
```

Nizovi znakova se uspoređuju slovo po slovo (uzimajući u obzir ASCII kod)

Relacijski operatori

- Redoslijed znakova (ASCII tablica):

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Zato je:
't' > 'T'

Usput: kako saznati ASCII kod nekog slova?

```
>>> ord('T')
84
>>> ord('t')
116
```

Donošenje odluka u programima

- Kako izvesti programsku naredbu (ili više njih) na temelju ispitivanja nekog uvjeta:

```
...
ako je uvjet onda
        naredba1_1
        ...
        naredba1_n
...
```

Donošenje odluka u programima

- U Pythonu:

```
...
if uvjet:
    naredba1_1
    ...
    naredba1_n
...
```

Izvršit će se ako je *uvjet* zadovoljen (logički izraz je istinit)

Svaki redak koji će se izvesti ako je *uvjet* zadovoljen mora biti uvučen. Najbolje je koristiti tipku TAB

- Primjer:

```
if a > b:
    print ('A je veće od B')
```

Zadatak: Paran broj

- Napisati program koji će za uneseni cijeli broj ispisati je li taj broj paran. Na kraju programa ispisati poruku ‘Kraj programa’.

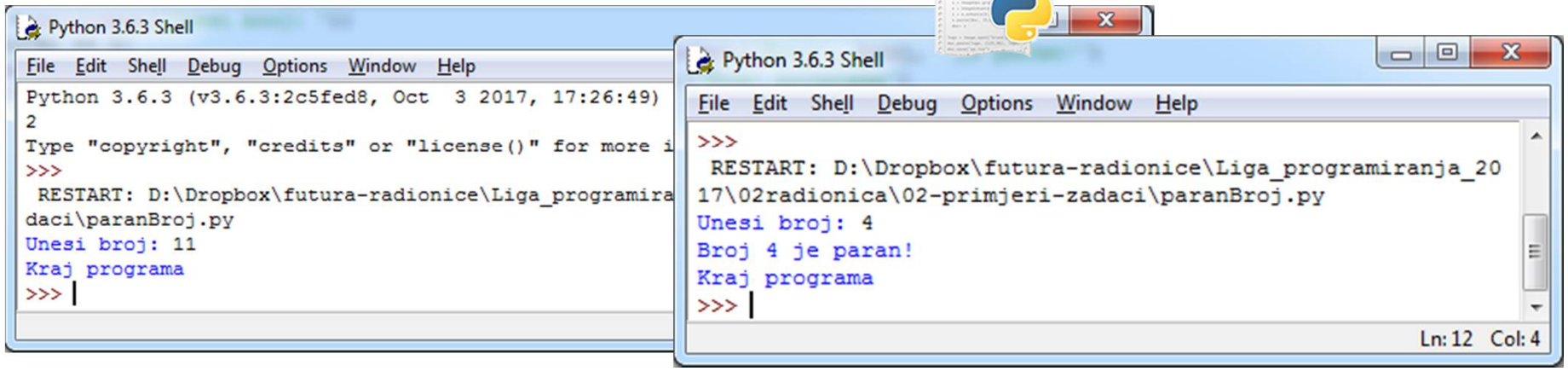
- Sjećate li se kako se piše i pokreće Python program?

Zadatak: Paran broj

- Napisati program koji će za uneseni cijeli broj ispisati da li je taj broj paran. Na kraju programa ispisati poruku ‘Kraj programa’.
- Rješenje:

```
broj = int(input('Unesi broj: '))
if broj%2 == 0:
    print('Broj', broj, 'je paran!')
print('Kraj programa')
```

paranBroj.py



Zadatak: Kolači

- **Futura** organizira cjelodnevno natjecanje u programiranju. Učenici kod dolaska dobivaju kolače i čokolade kako bi programi bili bolji.
- Nabavljen je veći broj kolača i čokolada. Svaki učenik kad stigne dobije **3 kolača** ili **2 čokolade**. Pretpostavlja se da ima dovoljno kolača i čokolada za sve učenike, ali i da nema dovoljno kolača da svi dobiju kolače.
- Kako je rok trajanja kolača kraći, prvo je potrebno podijeliti kolače.
- Učenik dobije ili samo kolače ili samo čokolade.

Zadatak: Kolači

- Ulazni podaci:
  - prirodni broj K - broj kupljenih kolača
  - prirodni broj C - broj kupljenih čokolada
  - prirodni broj N - broj učenika
- Ispis rezultata:
  - ispisati koliko je učenika dobilo kolače
  - ispisati koliko učenika je dobilo čokolade
  - ispisati ako je ostalo kolača

Zadatak: Kolači

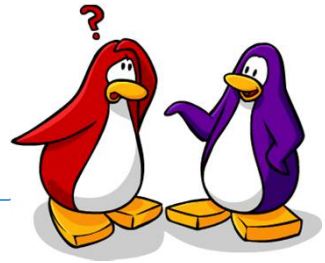
- Primjeri testnih podataka:

ULAZ	ULAZ	ULAZ
Broj kolača: **11**	Broj kolača: **15**	Broj kolača: **20**
Broj čokolada: **4**	Broj čokolada: **10**	Broj čokolada: **20**
Broj učenika: **4**	Broj učenika: **5**	Broj učenika: **16**
IZLAZ	**IZLAZ**	**IZLAZ**
Dobili kolače: **3**	Dobili kolače: **5**	Dobili kolače: **6**
Dobili čokoladu: **1**	Dobili čokoladu: **0**	Dobili čokoladu: **10**
Ostalo kolača!	Nije ostalo kolača!	Ostalo kolača!

** svaki učenik kad stigne dobije
3 kolača ili **2 čokolade**

Vrijeme

Kakav je ovo zadatak!?!



- Zadaci na natjecanjima najčešće su zadani u obliku priče koja bi sudionicima natjecanja trebala biti zabavna, zanimljiva i poticajna
- Priča opisuje problemsko područje i postavlja okvir za rješavanja zadatka
- Naglašeni su specifični uvjeti i ograničenja
- Detaljno je opisan oblik ulaznih podataka i izlaznih rezultata
- Obavezno je navedeno nekoliko ulaznih podataka i očekivanih izlaza
  - **OPREZ!** Navedeni podaci za testiranje često ne pokrivaju SVE moguće situacije

Pristup rješenju

1. Pročitaj zadatak i shvati ga
   - **Ne idi dalje dok nisi shvatio zadatak!**
2. Skiciraj rješenje
   - Blok dijagramom,
   - Pseudokodom,
   - Slobodnim tekstom,…
3. Izaberi alat (programski jezik)
4. Programiraj u **malim koracima**
   1. Isprogramiraj mali komadić koda
   2. Istestiraj napisani komadić koda
   3. Ako je do tada napisani kod u redu, dodaj novi

Zadatak: Kolači

- Primjeri testnih podataka:

ULAZ	ULAZ	ULAZ
Broj kolača: **11**	Broj kolača: **15**	Broj kolača: **20**
Broj čokolada: **4**	Broj čokolada: **10**	Broj čokolada: **20**
Broj učenika: **4**	Broj učenika: **5**	Broj učenika: **16**
IZLAZ	**IZLAZ**	**IZLAZ**
Dobili kolače: **3**	Dobili kolače: **5**	Dobili kolače: **6**
Dobili čokoladu: **1**	Dobili čokoladu: **0**	Dobili čokoladu: **10**
Ostalo kolača!		Ostalo kolača!

** svaki učenik kad stigne dobije
3 kolača ili **2 čokolade**

Vrijeme

Zadatak: Kolači - rješenje

```
k = int(input('Broj kupljenih kolača: '))
c = int(input('Broj kupljenih čokolada: '))
n = int(input('Broj učenika: '))
br_uc_k = k // 3                # dobili kolač
br_uc_c = n - br_uc_k           # dobili čokoladu
ost_k = k - (br_uc_k * 3)       # ostalo kolača
print('Broj uč. koji su dobili kolač:', br_uc_k)
print('Broj uč. koji su dobili čokoladu:', br_uc_c)
if ost_k > 0:
    print('Ostalo kolača!')
```

kolaci.py

Donošenje odluka u programima

- Često je u programima potrebno odabrati jednu od dvije mogućnosti:

```
...
ako je uvjet onda
        naredba1_1
        ...
        naredba1_n
inače
        naredba2_1
        ...
        naredba2_m
...
```

Izvršit će se samo JEDAN od ova dva bloka naredbi!

Usput: ovakav način opisa programa zove se "pseudojezik" ili "pseudokod"

Donošenje odluka u programima

- U Pythonu:

```
...
if uvjet:
        naredba1_1
        ...
        naredba1_n
else:
        naredba2_1
        ...
        naredba2_m
...
```

Izvršit će se samo JEDAN od ova dva bloka naredbi!

Donošenje odluka u programima

- U Pythonu:

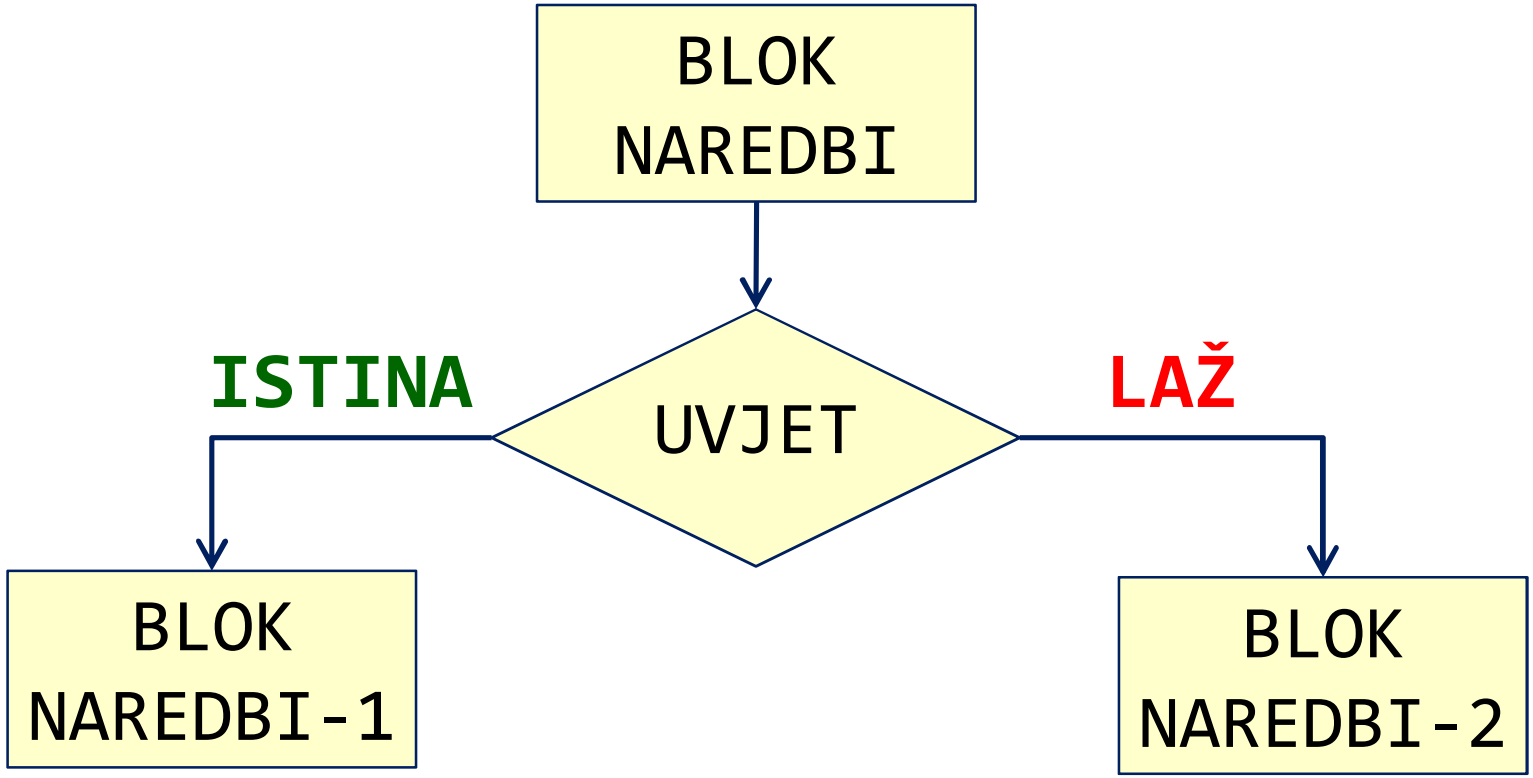
```
...
if uvjet:
        naredba1_1
        ...
        naredba1_n
else:
        naredba2_1
        ...
        naredba2_m
...
```

Izvršit će se ako je *uvjet* zadovoljen (logički izraz je istinit)

Izvršit će se ako *uvjet* nije zadovoljen (logički izraz je lažan)

Donošenje odluka u programima

- Dakle, na temelju ispitivanja *uvjeta* imamo grananje programa:



Donošenje odluka u programima

- U Pythonu je važno paziti na uvlačenje redaka naredbi koje su dio bloka :

```
...
if uvjet:
    naredba1_1
    ...
    naredba1_n
else:
    naredba2_1
    ...
    naredba2_m
...
```

Izvršit će se samo JEDAN od ova dva bloka naredbi!

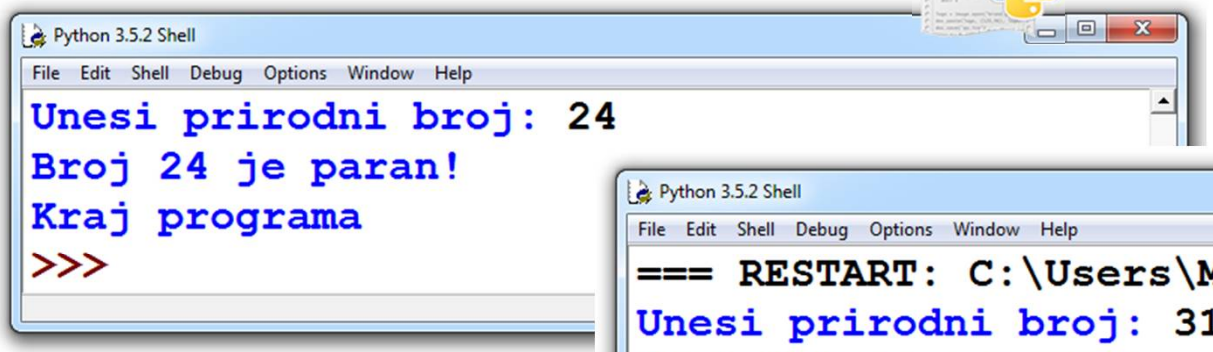
Najbolje je koristiti tipku TAB

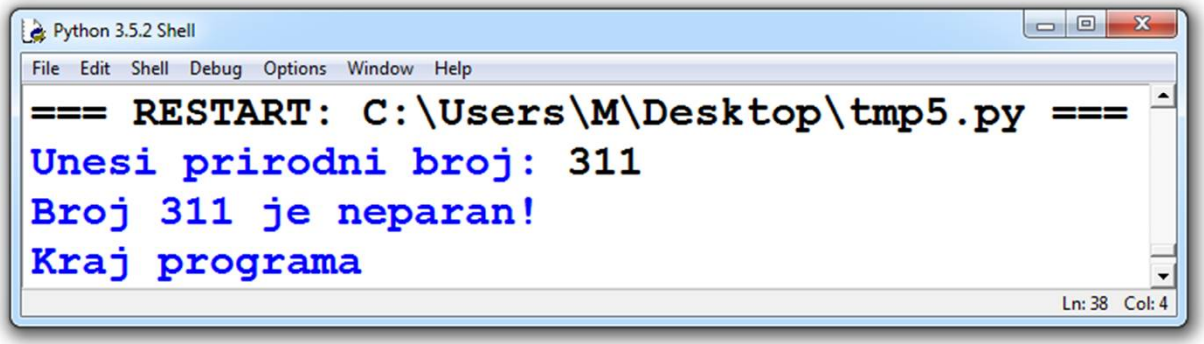
Primjer: parni i neparni brojevi

```
broj = int(input('Unesi prirodni broj: '))
if broj%2 == 0:
    print('Broj', broj, 'je paran!')
else:
    print('Broj', broj, 'je neparan!')
print('Kraj programa')
```

Izvršit će se samo JEDAN od ova dva bloka naredbi!

parNepar.py

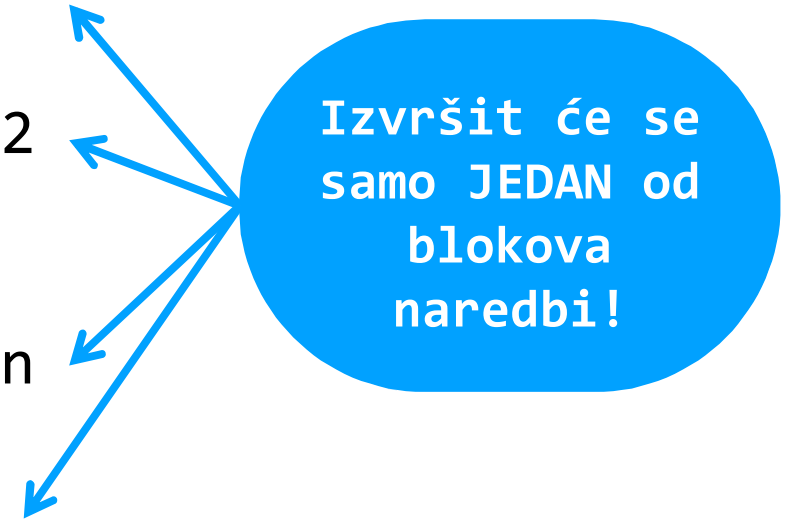




Donošenje odluka u programima

- U slučaju višestrukog izbora:

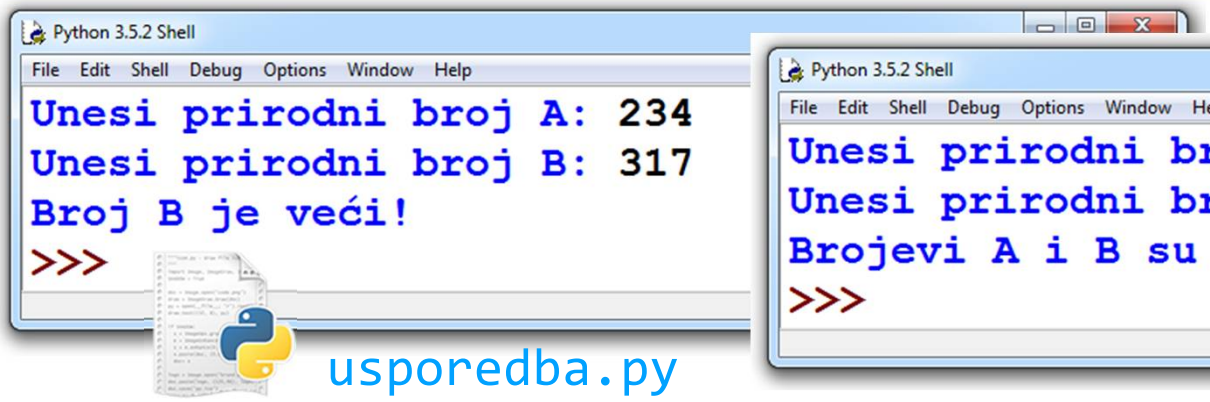
```
...
if uvjet_1:
        blok naredbi_1
elif uvjet_2:
        blok  naredbi_2
...
elif uvjet_n:
        blok  naredbi_n
else:
        blok naredbi
 ...
```



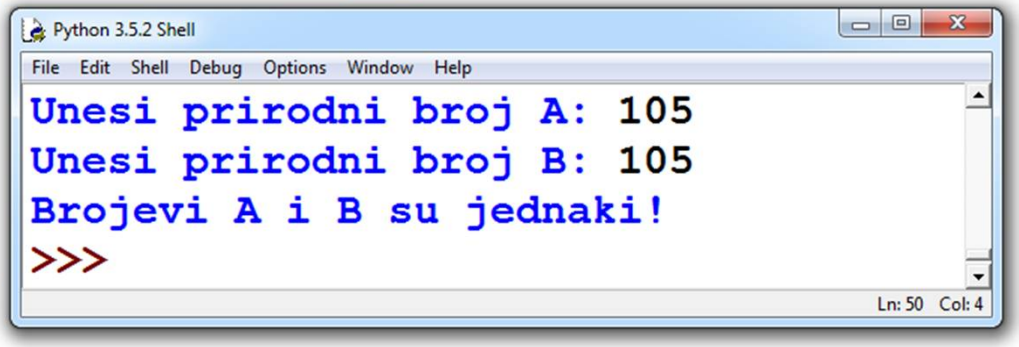
Primjer: usporedba brojeva

```
A = int(input('Unesi prirodni broj A: '))
B = int(input('Unesi prirodni broj B: '))
if A > B:
    print('Broj A je veći!')
elif B > A:
    print('Broj B je veći!')
else:
    print('Brojevi A i B su jednaki!')
```

Izvršit će se samo JEDAN od ova tri bloka naredbi!



usporedba.py



Logički operatori i logički izrazi

- Što ako je uvjet na temelju kojeg treba donijeti odluku složen?
- Logički operatori:

logička I operacija	**and**
logička ILI operacija	**or**
NE operacija (negacija)	**not**

- Redoslijed izvođenja logičkih operacija:

1.	**not**
2.	**and**
3.	**or**

Logički operatori i logički izrazi

- Primjeri logičkih operacija:

```
>>> a = 2
>>> b = 3
>>> c = 10
>>> a > b
False
>>> c > b
True
>>> a > b and c > b
False
>>> a > b or c > b
True
```

Za logičku **AND** operaciju rezultat će biti **True** (istina) samo ako su oba izraza **True** (istinita)

Za logičku **OR** operaciju rezultat će biti **True** (istina) već ako je jedan od izraza **True** (istinit)

Logički operatori i logički izrazi

- Kako se podsjetiti ishoda logičkih operacija?

and (i)

```
>>> False and False
False
>>> False and True
False
>>> True and False
False
>>> True and True
True
```

or (ili)

```
>>> False or False
False
>>> False or True
True
>>> True or False
True
>>> True or True
True
```

not (ne)

```
>>> not False
True
>>> not True
False
```

- Je li isto `True` i `true`? Ili `False` i `false`?
- Što je rezultat izraza: `True and true`

Logički operatori i logički izrazi

- Primjeri logičkih operacija:

```
>>> a = 2
>>> b = 3
>>> c = 10
>>> a+2*3>=c or not(a>b) and a*b-2==c%6
```

1.	aritmetički
2.	relacijski
3.	logički

Ako imamo kombinirane aritmetičke, relacijske i logičke operatore, onda je ovo redoslijed izvođenja operacija.

Logički operatori i logički izrazi

- Primjeri logičkih operacija:

```
>>> a = 2
>>> b = 3
>>> c = 10
>>> a+2*3>=c or not(a>b) and a*b-2==c%6
True
>>>
>>> (a+2*3>=c) or (not(a>b) and (a*b-2==c%6))
True
>>>
```

Ako ipak nismo posve sigurni u redoslijed operacija onda je najbolje koristiti zagrade!

Zadatak: Rođendan 1

- Lucija je sportašica i želi sportski proslaviti rođendan. Pozvala je dosta prijatelja, ali ne zna točno koliko ih će doći.
- Zato je napravila plan:
  - ako dođe od 1 do 3 prijatelja igrat će stolni tenis
  - ako dođe od 4 do 10 prijatelja igrat će odbojku
  - ako dođe više od 10 prijatelja igrat će nogomet
- Ulazni podatak: koliko prijatelja dolazi na rođendan
- Izlazni podatak: naziv sporta kojeg će igrati

Primjeri testnih podataka

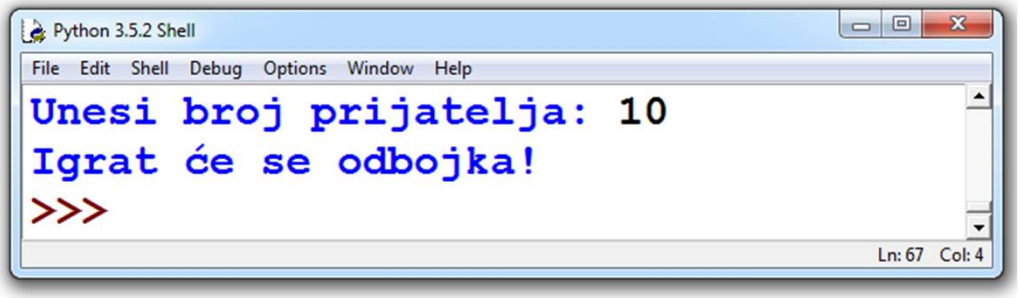
ULAZ	ULAZ
3	15
IZLAZ	**IZLAZ**
Igrat će se stolni tenis.	Igrat će se nogomet.

Zadatak: Rođendan 1 - rješenje


```
N = int(input('Unesi broj prijatelja: '))
if N <= 3:
    print('Igrat će se stolni tenis!')
elif N >= 4 and N <= 10:
    print('Igrat će se odbojka!')
else:
    print('Igrat će se nogomet!')
```

Testirati program s dovoljno vrijednosti – npr.:

1 3 4 7 10 11 15 150



rodjendan1.py

Zadatak: Rođendan 2

- Emilio je bio na Lucijinom rođendanu, ali su u kući igrali "Čovječe ne ljuti se" - jer je padala kiša!
- Zato on hoće napraviti plan i za kišovito i za sunčano vrijeme.

- Ako je sunčano vrijeme:
  - ako dođe od 1 do 3 prijatelja igrat će stolni tenis
  - ako dođe od 4 do 10 prijatelja igrat će odbojku
  - ako dođe više od 10 prijatelja igrat će nogomet

- Ako je kišovito vrijeme:
  - ako dođe od 1 do 3 prijatelja igrat će šah
  - ako dođe više od 3 prijatelja igrat će karata

Zadatak: Rođendan 2

- Ako je sunčano vrijeme:
  - ako dođe od 1 do 3 prijatelja igrat će stolni tenis
  - ako dođe od 4 do 10 prijatelja igrat će odbojku
  - ako dođe više od 10 prijatelja igrat će nogomet
- Ako je kišovito vrijeme:
  - ako dođe od 1 do 3 prijatelja igrat će šah
  - ako dođe više od 3 prijatelja igrat će karata

- Ulazni podaci:
  - koliko prijatelja dolazi na rođendan
  - kakvo je vrijeme ("sunčano" ili "kišovito")
- Izlazni podatak: naziv sporta kojeg će igrati

Zad.: Rođendan 2 - rješenje a)


```
N = int(input('Unesi broj prijatelja: '))
vr = input('Kakvo je vrijeme: ')
if vr == 'sunčano' and N <= 3:
    print('Igrat će se stolni tenis!')
elif vr == 'sunčano' and N >= 4 and N <= 10:
    print('Igrat će se odbojka!')
elif vr == 'sunčano' and N > 10:
    print('Igrat će se nogomet!')
elif vr == 'kišovito' and N <= 3:
    print('Igrat će se šah!')
else:
    print('Igrat će se karata!')
```

rodjendan2a.py

Zad.: Rođendan 2 - rješenje b)

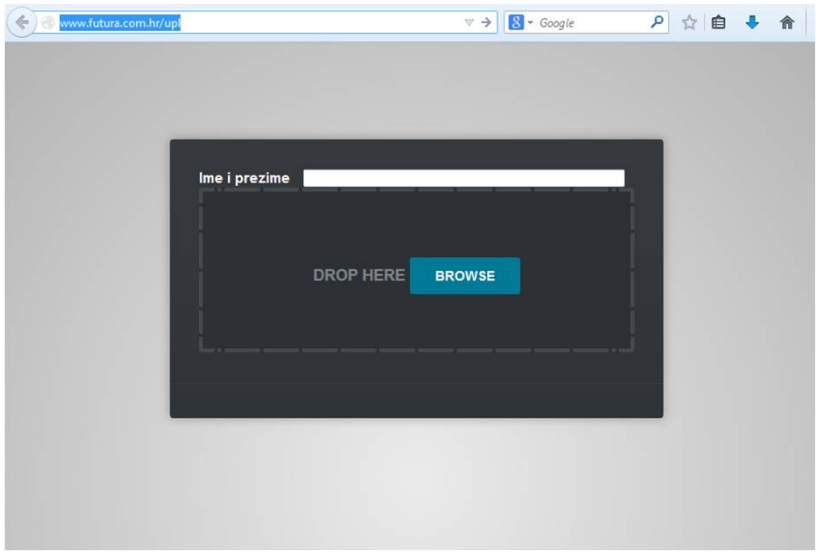

```
N = int(input('Unesi broj prijatelja: '))
vr = input('Kakvo je vrijeme: ')
if vr == 'sunčano':
    if N <= 3:
        print('Igrat će se stolni tenis!')
    elif N >= 4 and N <= 10:
        print('Igrat će se odbojka!')
    else:
        print('Igrat će se nogomet!')
else:
    if N <= 3:
        print('Igrat će se šah!')
    else:
        print('Igrat će se karata!')
```

rodjendan2b.py

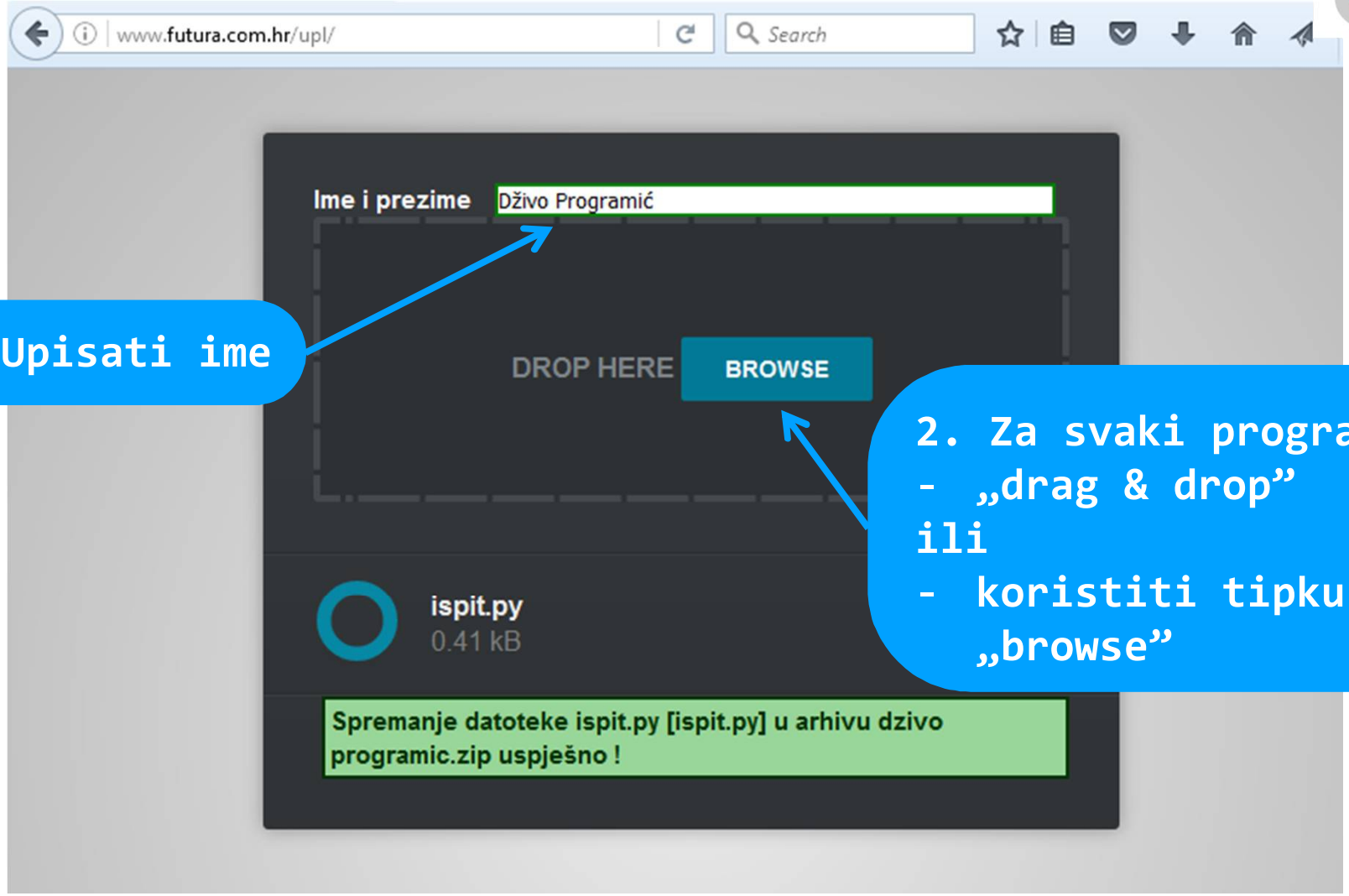
Slanje programa na natjecanju

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

www.futura.com.hr/upl



Slanje programa na natjecanju



Ne zaboravite!

- Za 15 dana – **u subotu 16.12.2017.** – **2. kolo Lige programiranja**
- **5./6. razredi** - početak **10:00**
- **7./8. razredi** - početak **10:00**
- **3** zadatka rješavate **75** minuta
- nemojte kasniti!

Za kraj – Natjecanje iz Informatike

https://informatika.azoo.hr/

O Natjecanju iz informatike 1/4

- 3 razine natjecanja

- Školsko natjecanje: **18. siječnja 2018.**
- Županijsko natjecanje: 9. veljače 2018.
- Državno natjecanje: 13.-16. ožujka 2018.

O Natjecanju iz informatike 2/4

- Vrste natjecanja
  - Osnove informatike (teorija)
  - **Primjena algoritama OŠ**
    (to je ono što učimo na radionicama)
  - Razvoj softvera
- Školska razina natjecanja je **18.01.2018.**
- Propozicije Natjecanje iz informatike 2018:
  https://informatika.azoo.hr/Content/Downloads/Propozicije-Informatika-2018.pdf

O Natjecanju iz informatike 3/4

- Formalni mentor učenicima na natjecanju mora biti nastavnik iz iste škole
  - Trebate se svojim nastavnicima informatike javiti što prije jer se škole trebaju prijaviti do 29. prosinca 2018.
  - Ako još niste sigurni – prijavite se do 16. siječnja 2018. do 15:00 (uvijek stignete prespavati ili odustati ☺)
  - Znači šifra je "**Primjena algoritama OŠ**"
- Web stranica natjecanja: informatika.azoo.hr
  - Detaljne informacije, propozicije natjecanja i prijava
  - Zadaci, rješenja i testni primjeri s prethodnih natjecanja

O Natjecanju iz informatike 4/4

- Za natjecanje u kategoriji **razvoj softvera** je za ovu godinu vjerojatno prekasno…
  - … osim ako već nemate gotovo isprogramirano rješenje!
  - Za iduću godinu (2018./2019. ima dovoljno vremena za kvalitetnu pripremu)
- Ako je netko zainteresiran za sudjelovanje u ovoj vrsti natjecanja neka nam se javi osobno na kraju radionice, ili naknadno e-mailom na **futura.dubrovnik@gmail.com**
  - Kod ove vrste natjecanja imate **potpunu slobodu** izbora aplikacije koju želite napraviti te programskog jezika i razvojnih alata koje želite koristiti

Dodatne pripreme za natjecanje

1. Ako ima zainteresiranih možemo vam poslati zadatke za samostalnu vježbu na vašu e-mail adresu, pa da nam se javite s rješenjima na futura.dubrovnik@gmail.com
2. Ako ima veći broj zainteresiranih možemo dogovoriti dodatni termin za radionicu/e preko zimskih školskih praznika?