DBBT
Digital Broadcasting & Broadband Technologies

Co-funded by the Erasmus+ Programme of the European Union

Procesiranje signala

Profesor dr Miroslav Lutovac

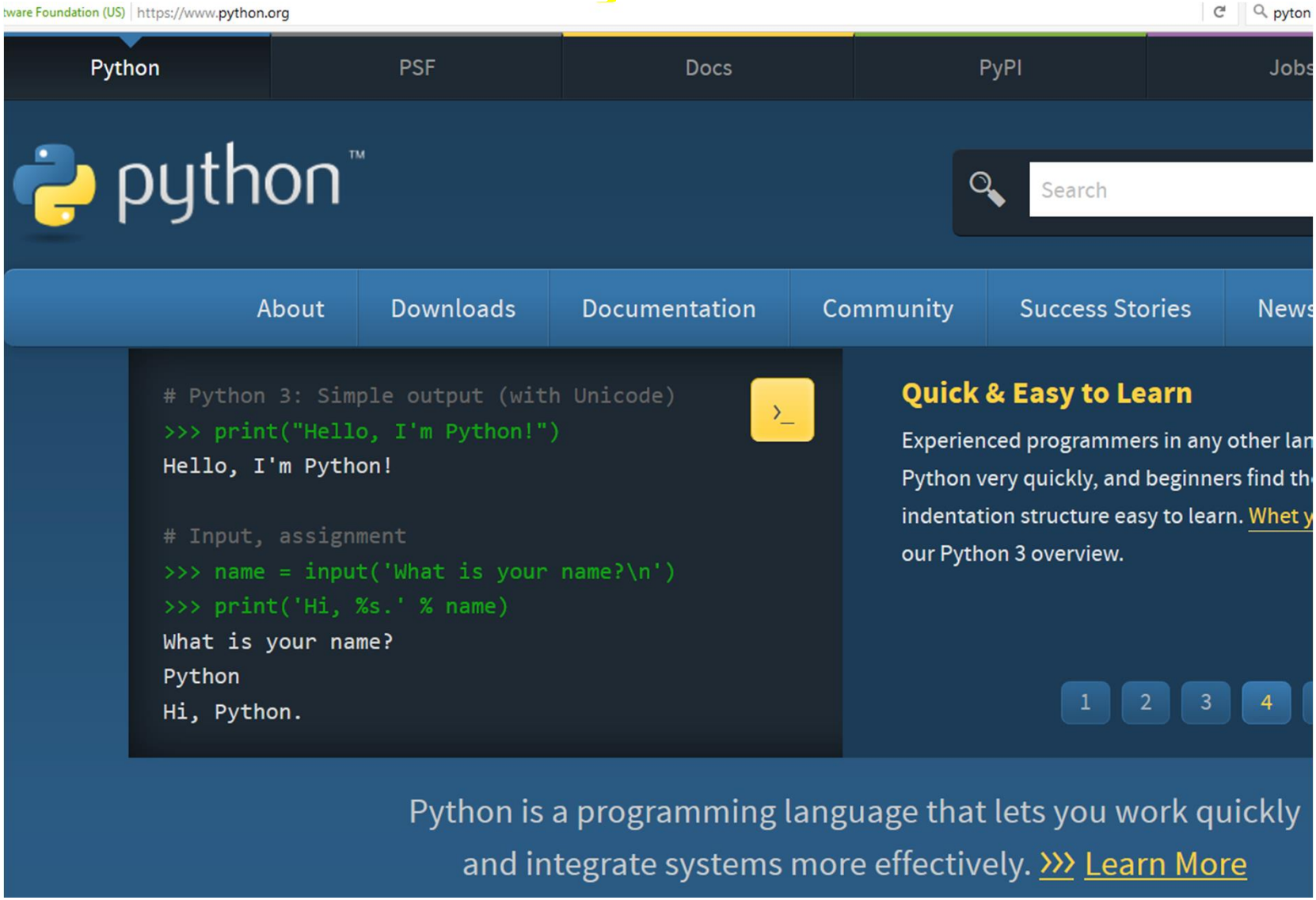
"This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein"

Sadržaj predmeta, Teorijska nastava

1. Uvodno predavanje. Upoznavanje sa planom i programom, ciljevima, ishodom i metodama.
2. Šta je procesiranje signala, istorijski pregled obrade signala, primeri primene.
3. Vizuelizacija signala (Python, Excel).
4. Kompleksni ekponencijalni diskretni signali. Primer sinteze muzičkog signala.
5. Furijeova analiza: Diskretna Furijeova transformacija (DFT) i serija (DFS). Brza Furijeova transformacija (Fast Fourier transform, FFT) i primena za spektralne analizatore i osciloskope.
6. Linearani filtri: konvolucija, idealni i realni filtri, dizajn filtra. Primena konvolucije u GPS sistemima.

…

Python


tware Foundation (US) | https://www.python.org
pyton
Python
PSF
Docs
PyPI
Jobs
python™
Search
About
Downloads
Documentation
Community
Success Stories
News
# Python 3: Simple output (with Unicode)
>>> print("Hello, I'm Python!")
Hello, I'm Python!
# Input, assignment
>>> name = input('What is your name?\n')
>>> print('Hi, %s.' % name)
What is your name?
Python
Hi, Python.
Quick & Easy to Learn
Experienced programmers in any other lan
Python very quickly, and beginners find th
indentation structure easy to learn. Whet y
our Python 3 overview.
1
2
3
4
Python is a programming language that lets you work quickly
and integrate systems more effectively. >>> Learn More

Python

- Easy to learn,
- powerful programming language
- Efficient high-level data structures
- Simple and effective object-oriented
- Dynamic typing, interpreted
- Scripting
- Rapid application development
- On most platforms

Python

- Freely available in source or binary form
- Freely distributed
- Modules programs, tools, documentation
  - search-and-replace over a large number of text files, or rename and rearrange photo files in a complicated way
  - small custom database,
  - specialized GUI application,
  - simple game

Why Python?

- Write/compile/test/re-compile cycle too slow
- Offers much more error checking
- Very-high-level language
- Split program into modules that can be reused in other programs
- Interpreted language
- Save considerable time during program development because no compilation and linking is necessary

Why Python?

- Interpreter used interactively
- Write throw-away programs
- Test functions during bottom-up program development
- Handy desk calculator
- Express complex operations in a single statement
- no variable or argument declarations are necessary

Python

Interactive Help in Python Shell

help()	Invoke interactive help
help(*m*)	Display help for module *m*
help(*f*)	Display help for function *f*
dir(*m*)	Display names in module *m*

Small Operator Precedence Table

func_name(*args, ...*)	Function call
x[*index* : *index*]	Slicing
x[*index*]	Indexing
x.attribute	Attribute reference
**	Exponentiation
*, /, %	Multiply, divide, mod
+, -	Add, subtract
>, <, <=, >=, !=, ==	Comparison
in, not in	Membership tests
not, and, or	Boolean operators NOT, AND, OR

Python

Module Import

```
import module_name
from module_name import name , ...
from module_name import *
```

Common Data Types

Type	Description	Literal Ex
int	32-bit Integer	3, -4
long	Integer > 32 bits	101L
float	Floating point number	3.0, -6.55
complex	Complex number	1.2J
bool	Boolean	True, False
str	Character sequence	“Python”
tuple	Immutable sequence	(2, 4, 7)
list	Mutable sequence	[2, x, 3.1]
dict	Mapping	{ x:2, y:5 }

Python

Common Syntax Structures

Assignment Statement

var = *exp*

Console Input/Output

var = input([*prompt*])
var = raw_input([*prompt*])
print *exp*[,] ...

Selection

if (*boolean_exp*):
    stmt ...
[elif (*boolean_exp*):
    stmt ...] ...
[else:
    stmt ...]

Repetition

while (*boolean_exp*):
    stmt ...

Traversal

for *var* in *traversable_object*:
    stmt ...

Python

Function Definition

```
def function_name( parmameters ):
    stmt ...
```

Function Call

```
function_name( arguments )
```

Class Definition

```
class Class_name [ (super_class) ]:
    [ class variables ]
    def method_name( self,  parameters ):
        stmt
```

Object Instantiation

```
obj_ref = Class_name( arguments )
```

Method Invocation

```
obj_ref.method_name( arguments )
```

Exception Handling

```
try:
    stmt ...
except [exception_type] [, var]:
    stmt ...
```

Python

Common Built-in Functions

Function	Returns
abs(*x*)	Absolute value of *x*
dict()	Empty dictionary, eg: d = dict()
float(*x*)	int or string *x* as float
id(*obj*)	memory addr of *obj*
int (*x*)	float or string *x* as int
len(*s*)	Number of items in sequence *s*
list()	Empty list, eg: m = list()
max(*s*)	Maximum value of items in *s*
min(*s*)	Minimum value of items in *s*
open(*f*)	Open filename *f* for input
ord(*c*)	ASCII code of *c*
pow(*x,y*)	x ** y
range(*x*)	A list of x ints 0 to *x* - 1
round(*x,n*)	float *x* rounded to *n* places
str(*obj*)	str representation of *obj*
sum(*s*)	Sum of numeric sequence *s*
tuple(*items*)	tuple of *items*
type(*obj*)	Data type of *obj*

Python

Common Math Module Functions

Function	Returns (all float)
ceil(*x*)	Smallest whole nbr >= *x*
cos(*x*)	Cosine of *x* radians
degrees(*x*)	*x* radians in degrees
radians(*x*)	*x* degrees in radians
exp(*x*)	e ** *x*
floor(*x*)	Largest whole nbr <= *x*
hypot(*x*, *y*)	sqrt(*x* * *x* + *y* * *y*)
log(*x* [, *base*])	Log of *x* to *base* or natural log if *base* not given
pow(*x*, *y*)	x ** y
sin(*x*)	Sine of *x* radians
sqrt(*x*)	Positive square root of *x*
tan(*x*)	Tangent of *x* radians
pi	Math constant pi to 15 sig figs
e	Math constant e to 15 sig figs

Python

Common String Methods

***S*.method()**	**Returns (str unless noted)**
capitalize	*S* with first char uppercase
center(*w*)	*S* centered in str *w* chars wide
count(*sub*)	int nbr of non-overlapping occurrences of *sub* in *S*
find(*sub*)	int index of first occurrence of *sub* in *S* or -1 if not found
isdigit()	bool True if *S* is all digit chars, False otherwise
islower() isupper()	bool True if *S* is all lower/upper case chars, False otherwise
join(*seq*)	All items in *seq* concatenated into a str, delimited by *S*
lower() upper()	Lower/upper case copy of *S*
lstrip() rstrip()	Copy of *S* with leading/ trailing whitespace removed, or both
split([*sep*])	List of tokens in *S*, delimited by *sep*; if *sep* not given, delimiter is any whitespace

Python

Formatting Numbers as Strings

Syntax: *"format_spec" % numeric_exp*
***format_spec* syntax:** *% width.precision type*

- *width* (optional): align in number of colums specified; negative to left-align, precede with 0 to zero-fill
- *precision* (optional): show specified digits of precision for floats; 6 is default
- *type* (required): d (decimal int), f (float), s (string), e (float – exponential notation)
- Examples for x = 123, y = 456.789
  "%6d" % x -> . . . 123
  "%06d" % x -> 000123
  "%8.2f % y -> . . 456.79
  "8.2e" % y -> 4.57e+02
  "-8s" % "Hello" -> Hello . . .

Python

Common List Methods

L.method()	Result/Returns
append(*obj*)	Append *obj* to end of *L*
count(*obj*)	Returns int nbr of occurrences of *obj* in *L*
index(*obj*)	Returns index of first occurrence of *obj* in *L*; raises ValueError if *obj* not in *L*
pop([*index*])	Returns item at specified *index* or item at end of L if *index* not given; raises IndexError if *L* is empty or *index* is out of range
remove(*obj*)	Removes first occurrence of *obj* from *L*; raises ValueError if *obj* is not in *L*
reverse()	Reverses *L* in place
sort()	Sorts *L* in place

Python

Common Tuple Methods

***T*.method()**	**Returns**
count(*obj*)	Returns nbr of occurrences of *obj* in *T*
index(*obj*)	Returns index of first occurrence of *obj* in *T*; raises ValueError if *obj* is not in *T*

Common Dictionary Methods

***D*.method()**	**Result/Returns**
clear()	Remove all items from *D*
get(*k* [,*val*])	Return *D*[*k*] if *k* in *D*, else *val*
has_key(*k*)	Return True if *k* in *D*, else False
items()	Return list of key-value pairs in *D*; each list item is 2-item tuple
keys()	Return list of *D*'s keys
pop(*k*, [*val*])	Remove key *k*, return mapped value or *val* if *k* not in *D*
values()	Return list of *D*'s values

Python

Common File Methods

***F*.method()**	**Result/Returns**
read([*n*])	Return str of next *n* chars from *F*, or up to EOF if *n* not given
readline([*n*])	Return str up to next newline, or at most *n* chars if specified
readlines()	Return list of all lines in *F*, where each item is a line
write(*s*)	Write str *s* to *F*
writelines(*L*)	Write all str in seq *L* to *F*
close()	Closes the file

Other Syntax

Hold window for user keystroke to close: raw_input("Press <Enter> to quit.")
Prevent execution on import: if __name__ == "__main__": main()

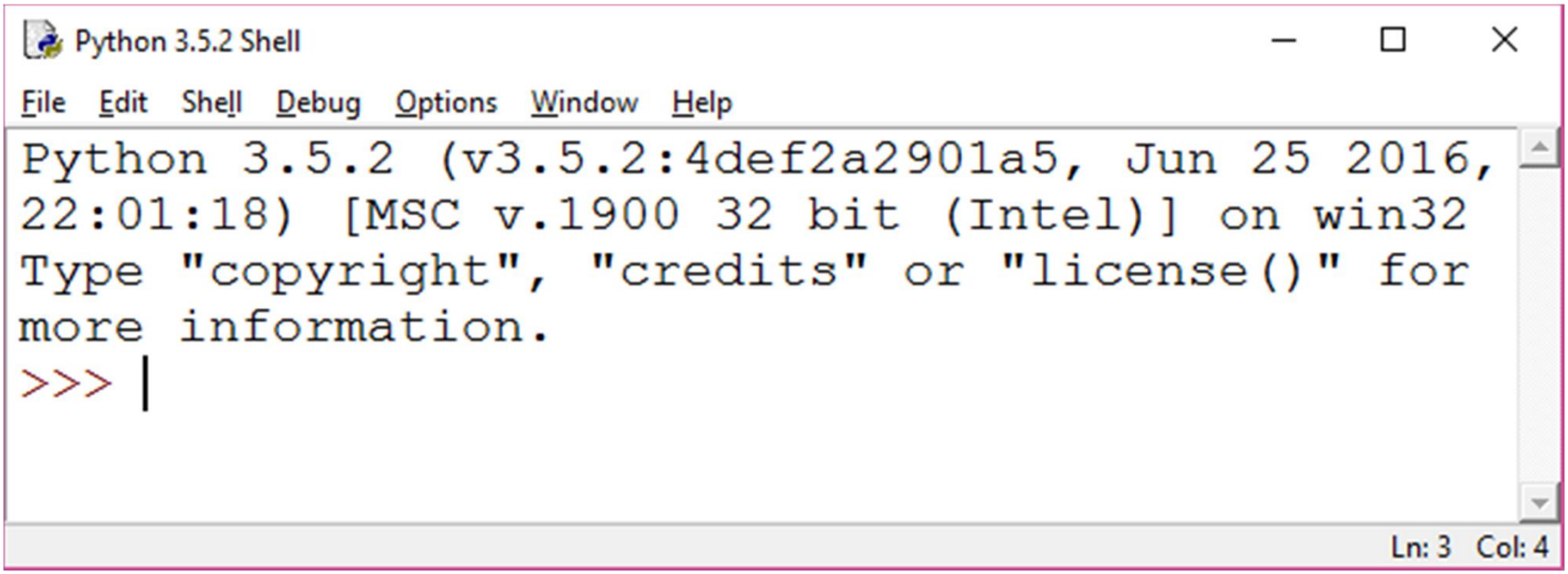
Python

Displayable ASCII Characters

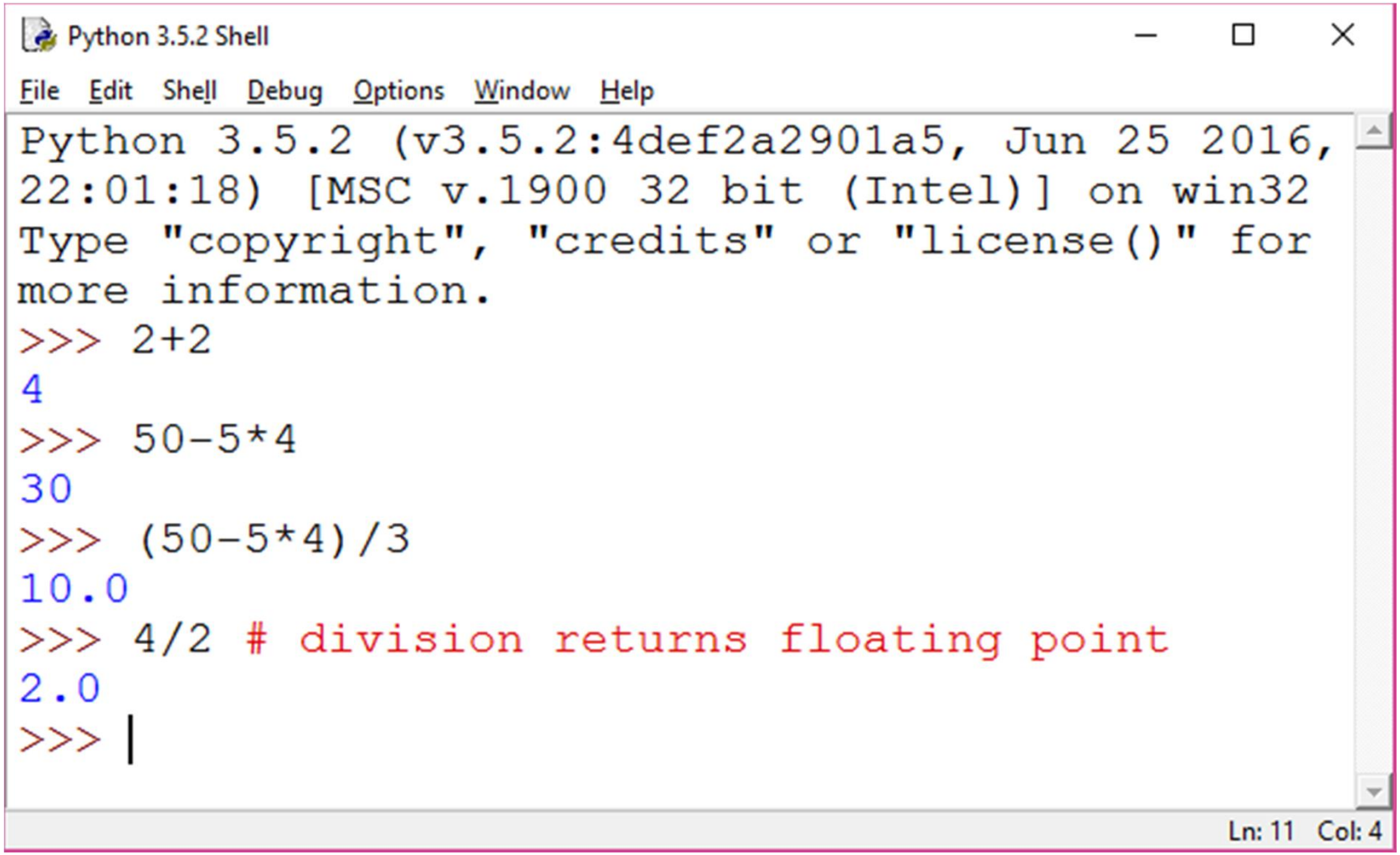
32	SP	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	“	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	‘	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	105	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	DEL

`'\0' = 0, '\t' = 9, '\n' = 10`

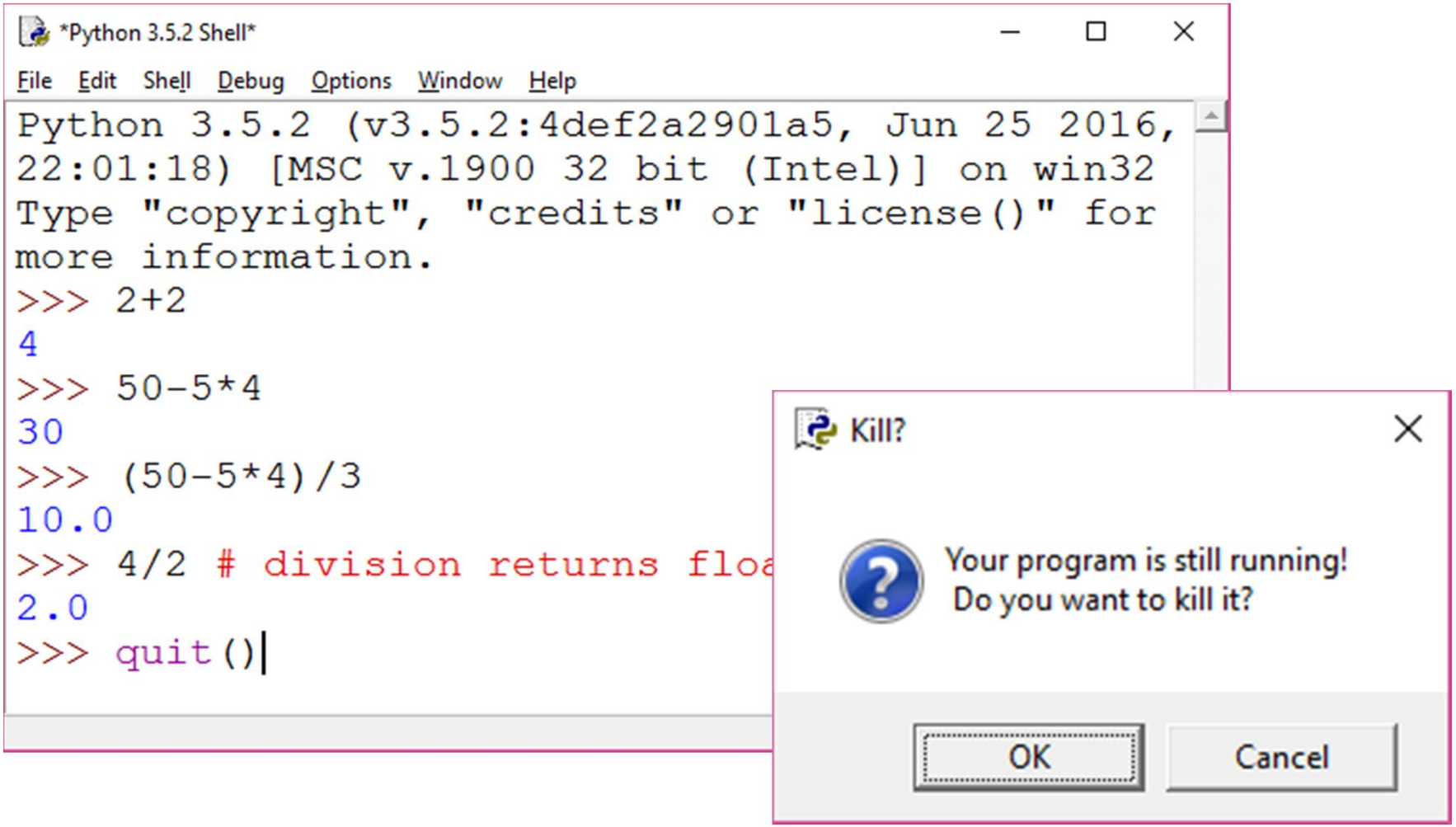
Python


Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
Ln: 3 Col: 4

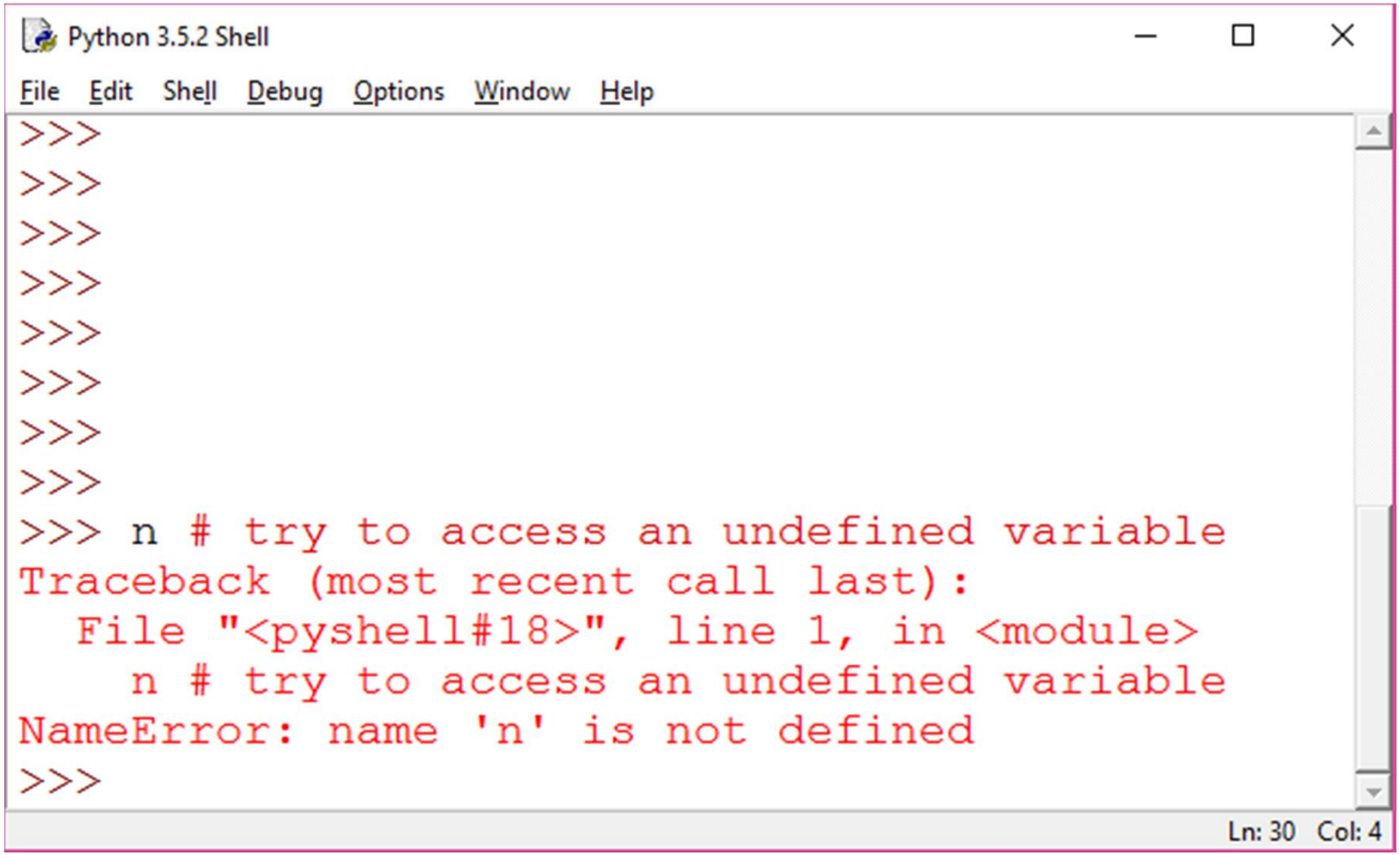
Python


Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>> 50-5*4
30
>>> (50-5*4)/3
10.0
>>> 4/2 # division returns floating point
2.0
>>>
Ln: 11 Col: 4

Python


Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>> 50-5*4
30
>>> (50-5*4)/3
10.0
>>> 4/2 # division returns floa
2.0
>>> quit()
Kill?
Your program is still running!
Do you want to kill it?
OK
Cancel

Python


Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>> n # try to access an undefined variable
Traceback (most recent call last):
File "<pyshell#18>", line 1, in <module>
n # try to access an undefined variable
NameError: name 'n' is not defined
>>>
Ln: 30 Col: 4

the last printed expression is assigned to the variable _

```
>>> 18 / 100
0.18
>>> _
0.18
>>> total = 100 + _
>>> total
100.18
>>>
```

```
>>> tax = 18 / 100
>>> _
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    _
NameError: name '_' is not defined
```

Strings

```
>>> 'ja sam Marko' # single quotes

'ja sam Marko'
```

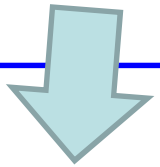
```
>>> 'doesn\'t' # use \' to escape the single quote...
"doesn't"

>>> "doesn't" # ...or use double quotes instead

"doesn't"

>>> '"Yes," he said.'

'"Yes," he said.'

>>> "\"Yes,\" he said."

'"Yes," he said.'

>>> '"Isn\'t," she said.'

'"Isn\'t," she said.'
```

Strings - raw strings

```
>>> print('C:\some\name') # here \n means newline!

C:\some

ame

>>> print(r'C:\some\name') # note the r before the quote
C:\some\name
```

Strings - """ or '''

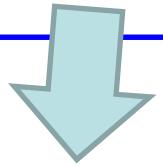
```
print("""\
Usage: thingy [OPTIONS]
       -h              Display this usage message
       -H hostname     Hostname to connect to
""")
```



```
Usage: thingy [OPTIONS]
       -h              Display this usage message
       -H hostname     Hostname to connect to
```

concatenated with the + operator, and repeated with *

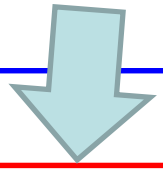
```
>>> # 3 times 'un', followed by 'ium'
>>> 3 * 'un' + 'ium'
```



```
'unununium'
```

Lists

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares
```



```
[1, 4, 9, 16, 25]
```

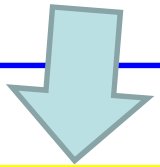
```
>>> squares[-1]
25
```

```
>>> squares[0]
1
```

```
>>>  squares[-3:]
[9, 16, 25]
```

Programming

```
>>> # Fibonacci series:
... # the sum of two elements defines the next
... a, b = 0, 1
>>> while b < 10:
... print(b)
... a, b = b, a+b
...
```

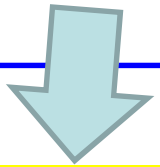


```
1
1
2
3
5
8
```

- ✓ multiple assignment
- ✓ while loop - condition (b < 10) remains true
- ✓ body of the loop is indented
- ✓ print() writes the value

Programming

```
>>> # Fibonacci series:
... # the sum of two elements defines the next
... a, b = 0, 1
>>> while b < 10:
... print(b)
... a, b = b, a+b
...
```



```
1
1
2
3
5
8
```

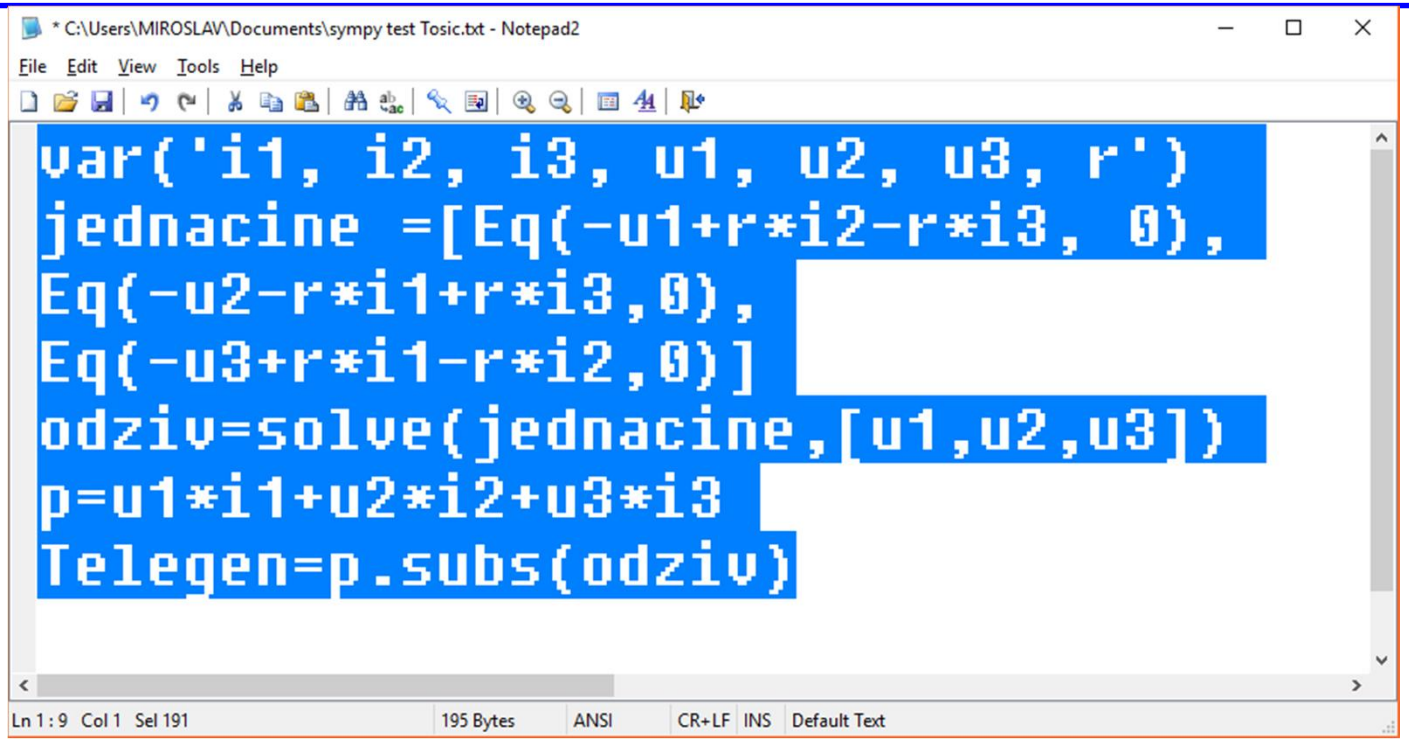
- ✓ multiple assignment
- ✓ while loop - condition (b < 10) remains true
- ✓ body of the loop is indented
- ✓ print() writes the value

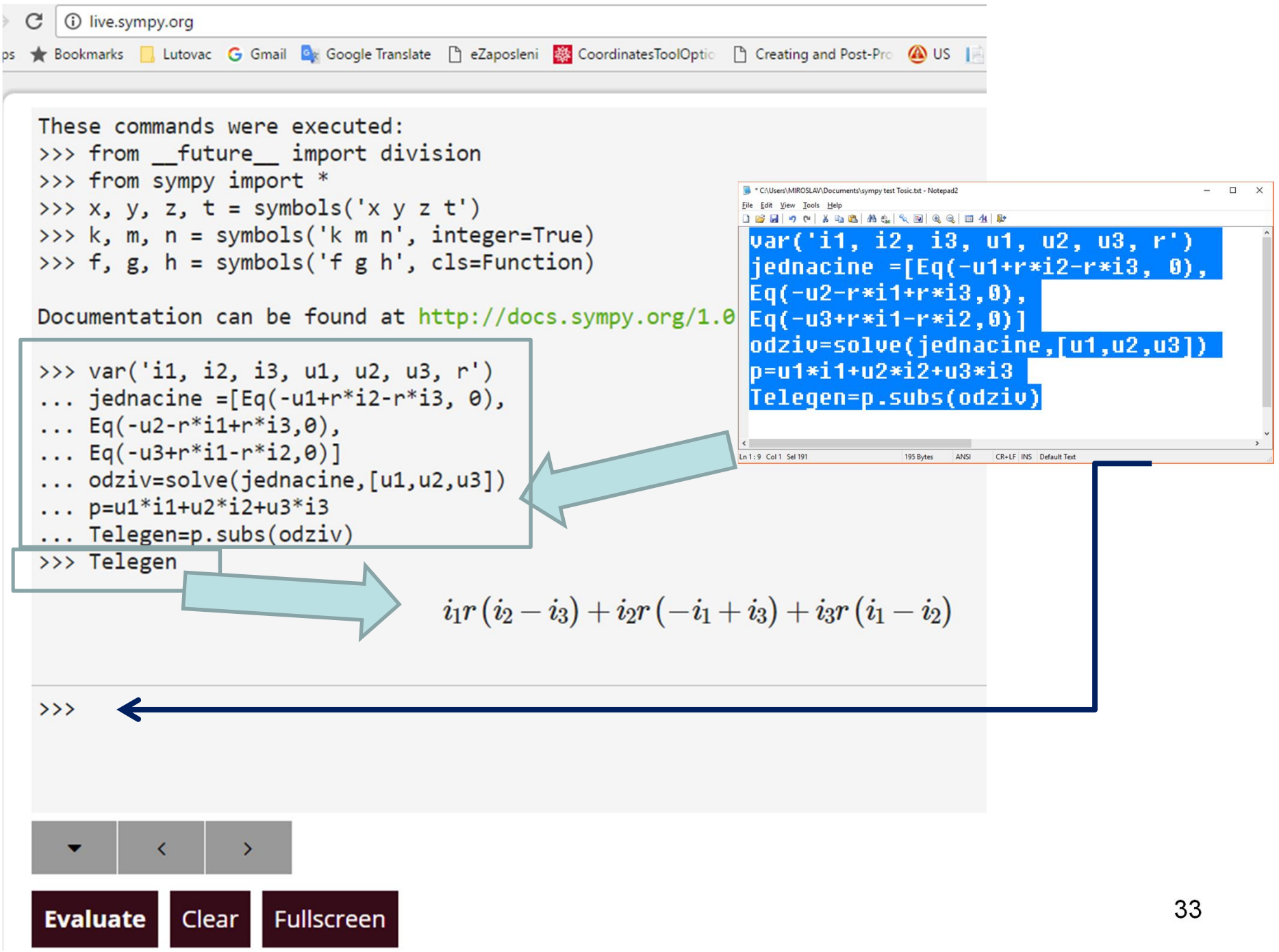
Python console for SymPy 1.0 (Python 2.7.5)

These commands were executed:

```
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
```

Documentation can be found at http://docs.sympy.org/1.0.




live.sympy.org
Bookmarks
Lutovac
Gmail
Google Translate
eZaposleni
CoordinatesToolOptic
Creating and Post-Pro
US
These commands were executed:
>>> from __future__ import division
>>> from sympy import *
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
Documentation can be found at http://docs.sympy.org/1.0
>>> var('i1, i2, i3, u1, u2, u3, r')
... jednacine =[Eq(-u1+r*i2-r*i3, 0),
... Eq(-u2-r*i1+r*i3,0),
... Eq(-u3+r*i1-r*i2,0)]
... odziv=solve(jednacine,[u1,u2,u3])
... p=u1*i1+u2*i2+u3*i3
... Telegen=p.subs(odziv)
>>> Telegen
$i_1 r\left(i_2 - i_3\right) + i_2 r\left(-i_1 + i_3\right) + i_3 r\left(i_1 - i_2\right)$
>>>
* C:\Users\MIROSLAV\Documents\sympy test Tosic.txt - Notepad2
File Edit View Tools Help
var('i1, i2, i3, u1, u2, u3, r')
jednacine =[Eq(-u1+r*i2-r*i3, 0),
Eq(-u2-r*i1+r*i3,0),
Eq(-u3+r*i1-r*i2,0)]
odziv=solve(jednacine,[u1,u2,u3])
p=u1*i1+u2*i2+u3*i3
Telegen=p.subs(odziv)
Ln 1 : 9 Col 1 Sel 191
195 Bytes
ANSI
CR+LF INS Default Text
Evaluate
Clear
Fullscreen

Profesor dr Miroslav Lutovac
mlutovac@viser.edu.rs