Įvadinė paskaita

2008-09-08

Dėstytojas

Gintautas Miliauskas

gintas@akl.lt
+370 685 13369

Planas

1. Kursas
   - Įvadas
   - Darbo aplinka
   - Ištekliai
2. Python duomenų tipai
   - Apžvalga
   - Skaitiniai tipai, bool
   - Simbolių sekos, sąrašai, žodynai
3. Sakiniai

Outline

1. Kursas
   - Įvadas
   - Darbo aplinka
   - Ištekliai
2. Python duomenų tipai
   - Apžvalga
   - Skaitiniai tipai, bool
   - Simbolių sekos, sąrašai, žodynai
3. Sakiniai

Kas yra Python?

- Programavimo kalba
- Paprasta
- Interpretuojama kalba
  (nereikia kompiliuoti, galima dirbti interaktyviai)
- Objektinė kalba

Kas yra Python?

- Programavimo kalba
- Paprasta
- Interpretuojama kalba
  (nereikia kompiliuoti, galima dirbti interaktyviai)
- Objektinė kalba

Kas yra Python?

- Programavimo kalba
- Paprasta
- Interpretuojama kalba
  (nereikia kompiliuoti, galima dirbti interaktyviai)
- Objektinė kalba

Kas yra Python?

- Programavimo kalba
- Paprasta
- Interpretuojama kalba
  (nereikia kompiliuoti, galima dirbti interaktyviai)
- Objektinė kalba

Kaip atrodo Python kodas?

```
def fib(n):
    """Calculate the nth Fibonacci number."""
    # Non-recursive
    a, b = 1, 1
    for i in range(n-2):
        a, b = b, a+b
    return b

def fibsum(n):
    """Sum the first n Fibonacci numbers."""
    return sum([fib(i) for i in range(1, n+1)])

for i in range(3, 15):
   print fib(i+2), fibsum(i)
```

Kodėl Python?

- Gyva kalba (ne mokomoji)
- Daug bibliotekų
- Naudojamas industrijoje.
  Firmos, naudojančios Python:
  Google, NASA, Nokia, YouTube, Ubuntu, ...
- Naudojamas mokymui.
  Python kursai skaitomi universitetuose:
  MIT, Oxford, Toronto, Waterloo, California, ...
- Tinka mažoms programoms (priešingai negu Java, C++, etc.)
- Tinka didelėms programoms (priešingai negu Perl, Logo, Pascal)

Kodėl Python?

- Gyva kalba (ne mokomoji)
- Daug bibliotekų
- Naudojamas industrijoje.
  Firmos, naudojančios Python:
  Google, NASA, Nokia, YouTube, Ubuntu, ...
- Naudojamas mokymui.
  Python kursai skaitomi universitetuose:
  MIT, Oxford, Toronto, Waterloo, California, ...
- Tinka mažoms programoms (priešingai negu Java, C++, etc.)
- Tinka didelėms programoms (priešingai negu Perl, Logo, Pascal)

Kodėl Python?

- Gyva kalba (ne mokomoji)
- Daug bibliotekų
- Naudojamas industrijoje.
  Firmos, naudojančios Python:
  Google, NASA, Nokia, YouTube, Ubuntu, ...
- Naudojamas mokymui.
  Python kursai skaitomi universitetuose:
  MIT, Oxford, Toronto, Waterloo, California, ...
- Tinka mažoms programoms (priešingai negu Java, C++, etc.)
- Tinka didelėms programoms (priešingai negu Perl, Logo, Pascal)

Kodėl Python?

- Gyva kalba (ne mokomoji)
- Daug bibliotekų
- Naudojamas industrijoje.
  Firmos, naudojančios Python:
  Google, NASA, Nokia, YouTube, Ubuntu, ...
- Naudojamas mokymui.
  Python kursai skaitomi universitetuose:
  MIT, Oxford, Toronto, Waterloo, California, ...
- Tinka mažoms programoms (priešingai negu Java, C++, etc.)
- Tinka didelėms programoms (priešingai negu Perl, Logo, Pascal)

Kodėl Python?

- Gyva kalba (ne mokomoji)
- Daug bibliotekų
- Naudojamas industrijoje.
  Firmos, naudojančios Python:
  Google, NASA, Nokia, YouTube, Ubuntu, ...
- Naudojamas mokymui.
  Python kursai skaitomi universitetuose:
  MIT, Oxford, Toronto, Waterloo, California, ...
- Tinka mažoms programoms (priešingai negu Java, C++, etc.)
- Tinka didelėms programoms (priešingai negu Perl, Logo, Pascal)

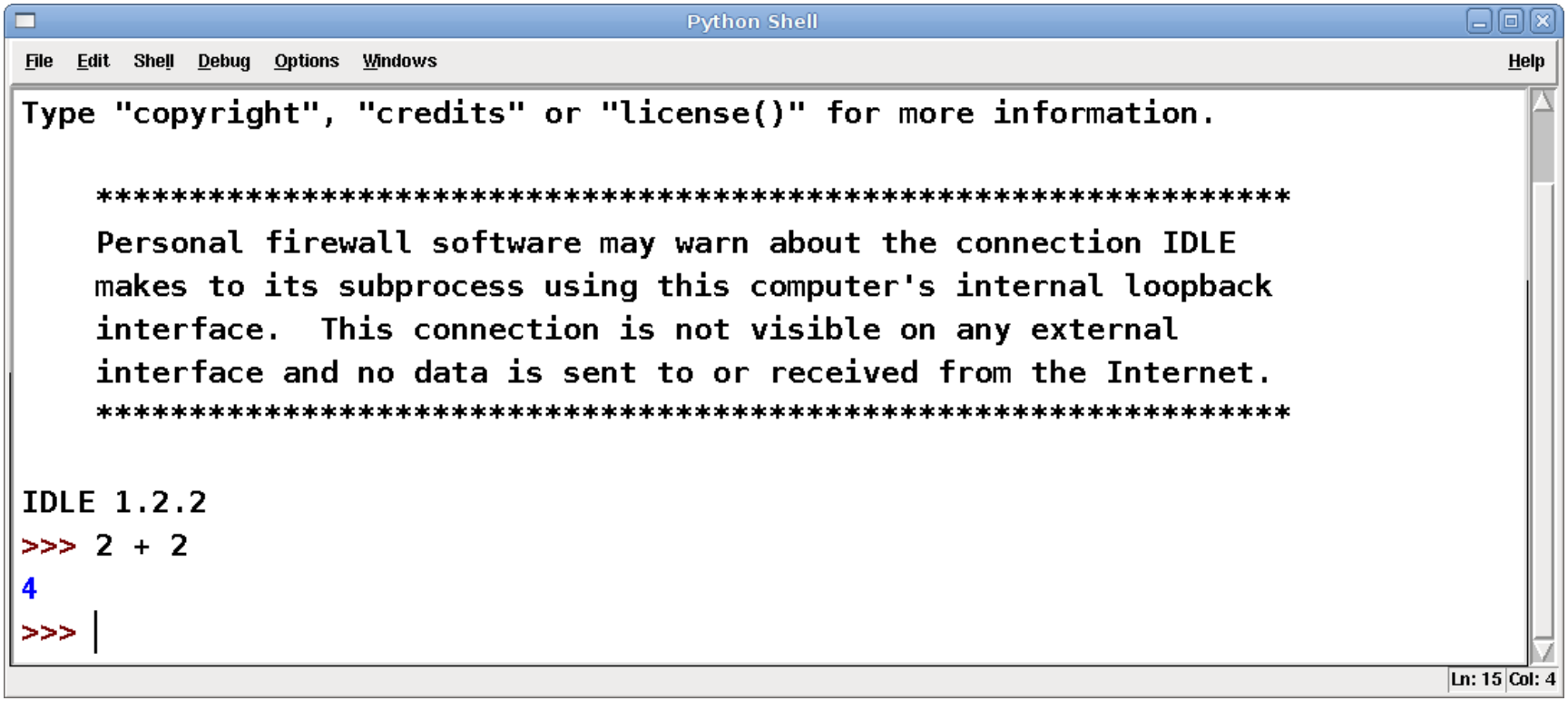
Kodėl Python?

- Gyva kalba (ne mokomoji)
- Daug bibliotekų
- Naudojamas industrijoje.
  Firmos, naudojančios Python:
  Google, NASA, Nokia, YouTube, Ubuntu, ...
- Naudojamas mokymui.
  Python kursai skaitomi universitetuose:
  MIT, Oxford, Toronto, Waterloo, California, ...
- Tinka mažoms programoms (priešingai negu Java, C++, etc.)
- Tinka didelėms programoms (priešingai negu Perl, Logo, Pascal)

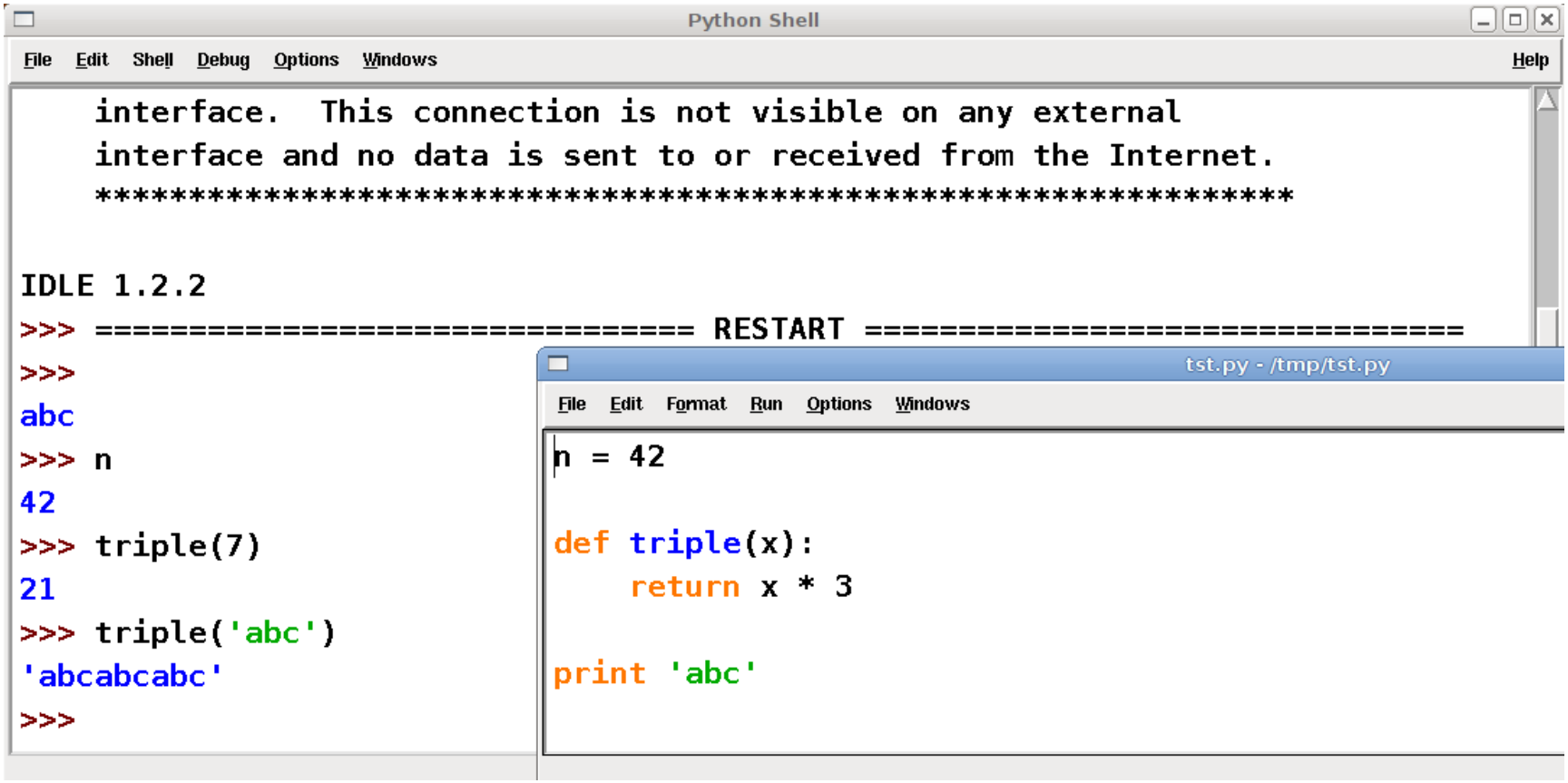
Outline

1. Kursas
   - Įvadas
   - Darbo aplinka
   - Ištekliai
2. Python duomenų tipai
   - Apžvalga
   - Skaitiniai tipai, bool
   - Simbolių sekos, sąrašai, žodynai
3. Sakiniai

IDLE – interaktyvus langas



IDLE – darbas su failais

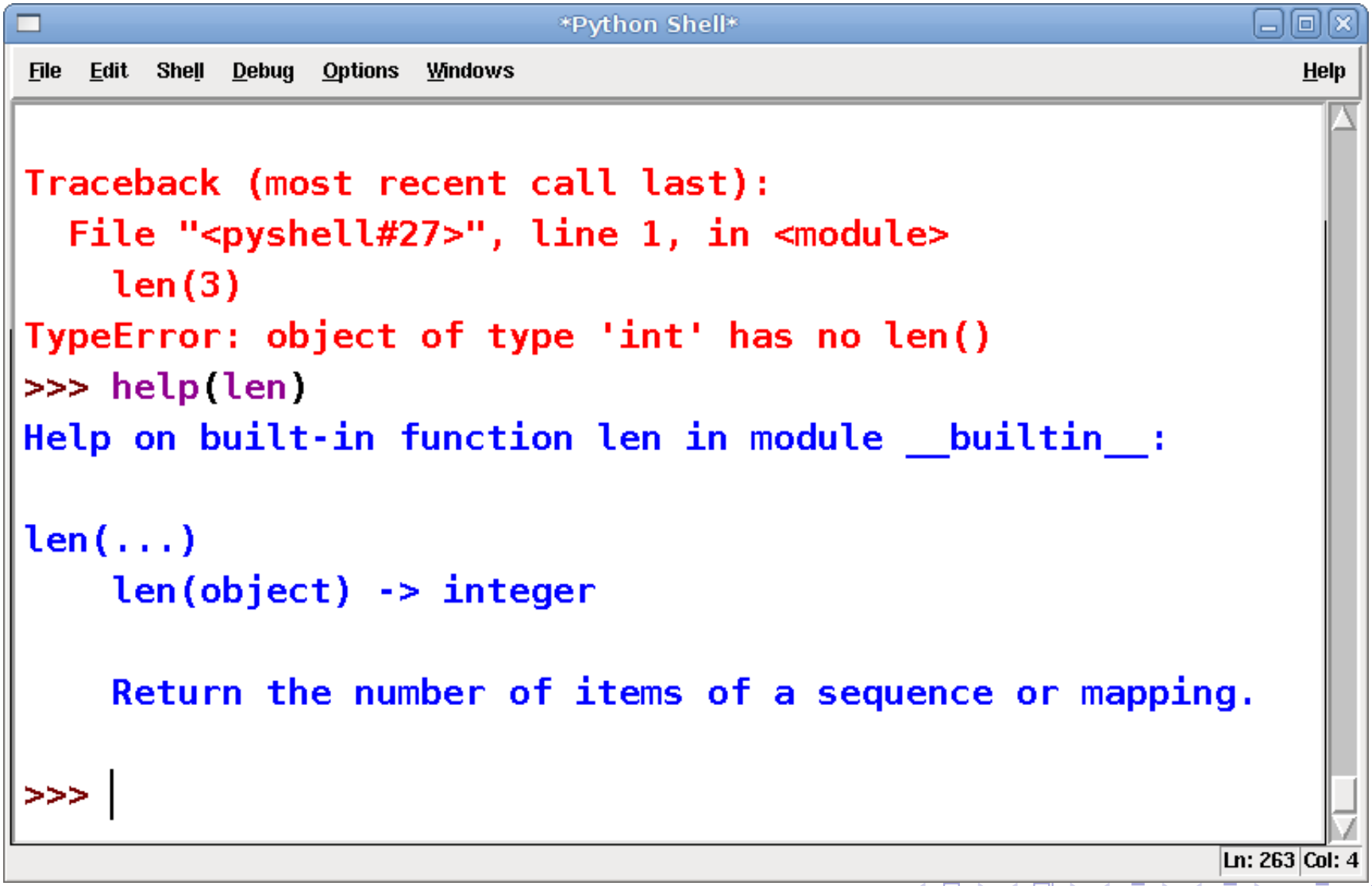


(F5 paleidžia programą)

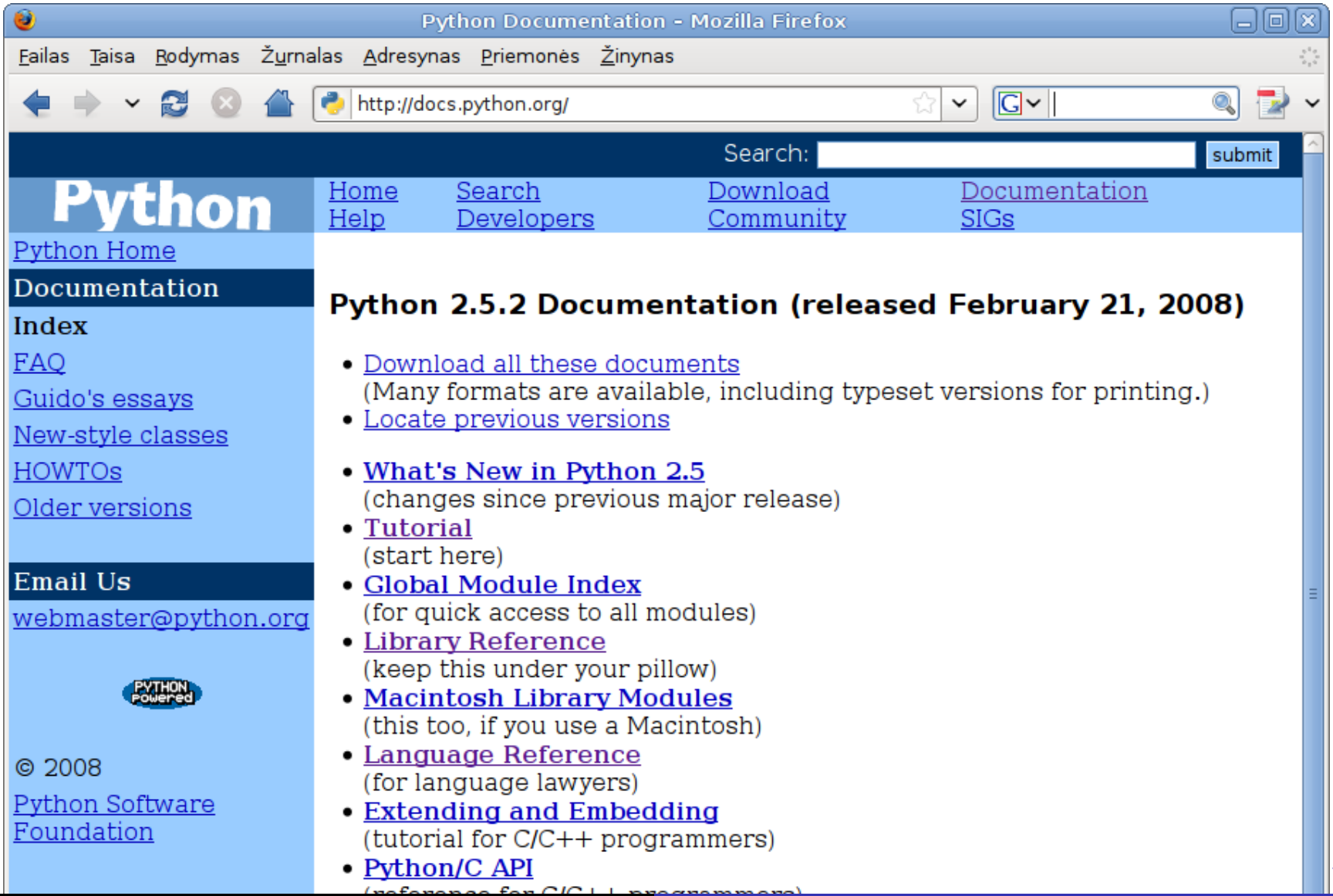
Outline

1. Kursas
   - Įvadas
   - Darbo aplinka
   - Ištekliai
2. Python duomenų tipai
   - Apžvalga
   - Skaitiniai tipai, bool
   - Simbolių sekos, sąrašai, žodynai
3. Sakiniai

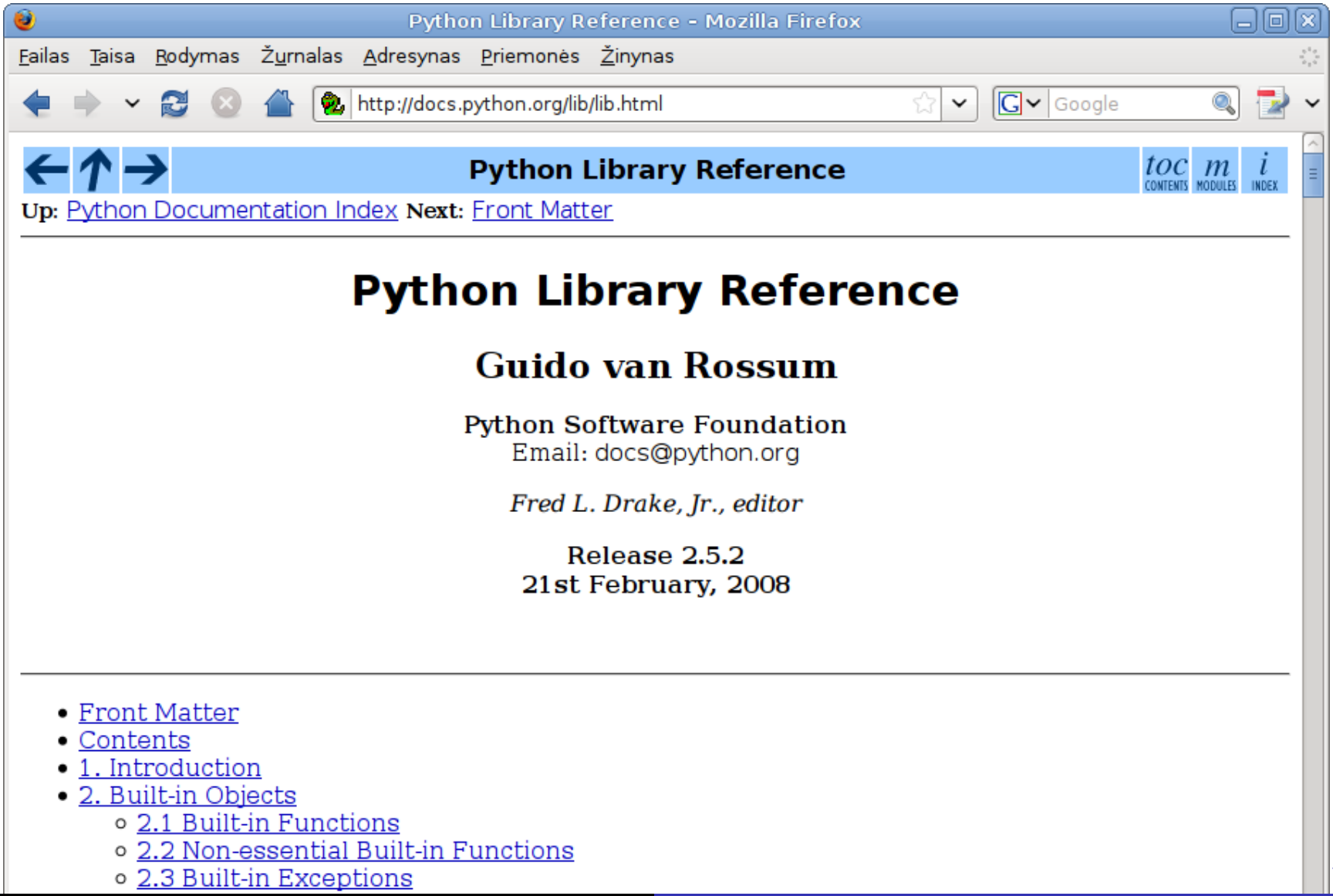
Vidinė Python dokumentacija



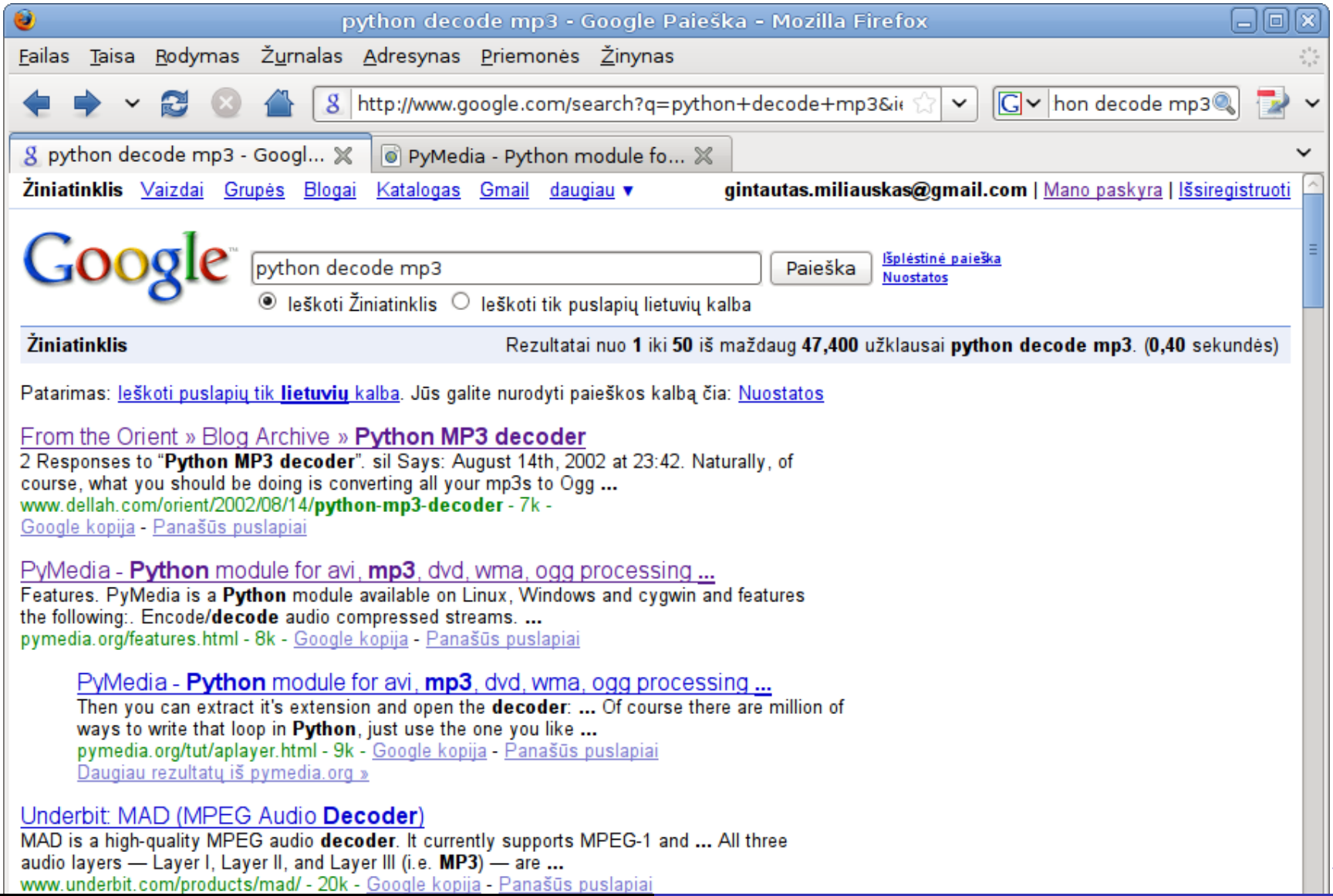
Python dokumentacija internete



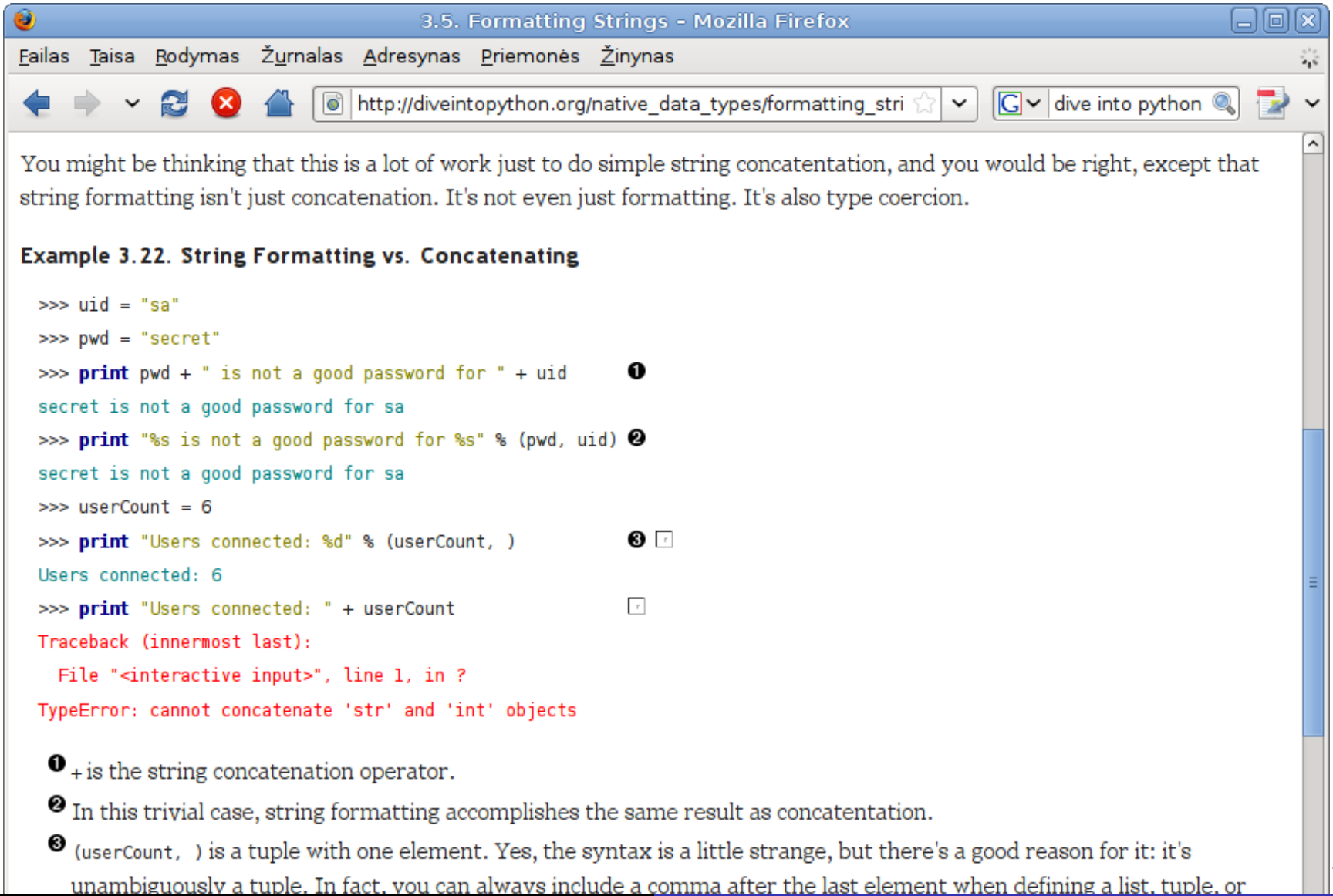
Standartinės bibliotekos aprašymas



Google


python decode mp3 - Google Paieška - Mozilla Firefox
Failas Taisa Rodymas Žurnalas Adresynas Priemonės Žinynas
http://www.google.com/search?q=python+decode+mp3&ie
hon decode mp3
python decode mp3 - Googl...
PyMedia - Python module fo...
Žiniatinklis Vaizdai Grupės Blogai Katalogas Gmail daugiau ▾
gintautas.miliauskas@gmail.com | Mano paskyra | Išsiregistruoti
Google
python decode mp3
Paieška
Išplėstinė paieška
Nuostatos
Ieškoti Žiniatinklis
Ieškoti tik puslapių lietuvių kalba
Žiniatinklis
Rezultatai nuo 1 iki 50 iš maždaug 47,400 užklausai python decode mp3. (0,40 sekundės)
Patarimas: Ieškoti puslapių tik lietuvių kalba. Jūs galite nurodyti paieškos kalbą čia: Nuostatos
From the Orient » Blog Archive » Python MP3 decoder
2 Responses to "Python MP3 decoder". sil Says: August 14th, 2002 at 23:42. Naturally, of course, what you should be doing is converting all your mp3s to Ogg ...
www.dellah.com/orient/2002/08/14/python-mp3-decoder - 7k -
Google kopija - Panašūs puslapiai
PyMedia - Python module for avi, mp3, dvd, wma, ogg processing ...
Features. PyMedia is a Python module available on Linux, Windows and cygwin and features the following:. Encode/decode audio compressed streams. ...
pymedia.org/features.html - 8k - Google kopija - Panašūs puslapiai
PyMedia - Python module for avi, mp3, dvd, wma, ogg processing ...
Then you can extract it's extension and open the decoder: ... Of course there are million of ways to write that loop in Python, just use the one you like ...
pymedia.org/tut/aplayer.html - 9k - Google kopija - Panašūs puslapiai
Daugiau rezultatų iš pymedia.org »
Underbit: MAD (MPEG Audio Decoder)
MAD is a high-quality MPEG audio decoder. It currently supports MPEG-1 and ... All three audio layers — Layer I, Layer II, and Layer III (i.e. MP3) — are ...
www.underbit.com/products/mad/ - 20k - Google kopija - Panašūs puslapiai

Knyga: Dive Into Python



You might be thinking that this is a lot of work just to do simple string concatentation, and you would be right, except that string formatting isn't just concatenation. It's not even just formatting. It's also type coercion.

Example 3.22. String Formatting vs. Concatenating

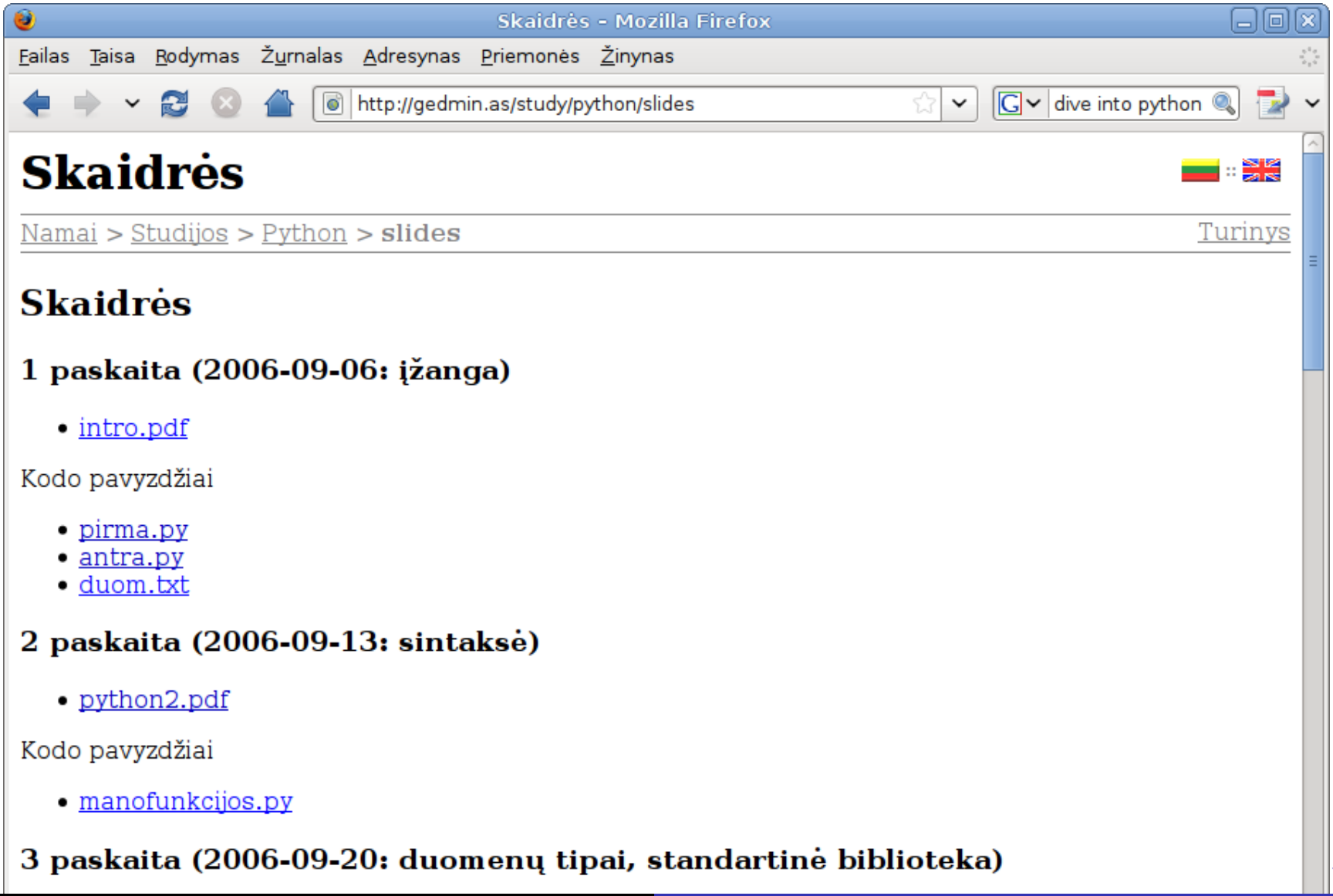
```
>>> uid = "sa"
>>> pwd = "secret"
>>> print pwd + " is not a good password for " + uid      ❶
secret is not a good password for sa
>>> print "%s is not a good password for %s" % (pwd, uid) ❷
secret is not a good password for sa
>>> userCount = 6
>>> print "Users connected: %d" % (userCount, )           ❸
Users connected: 6
>>> print "Users connected: " + userCount
Traceback (innermost last):
  File "<interactive input>", line 1, in ?
TypeError: cannot concatenate 'str' and 'int' objects
```

❶ + is the string concatenation operator.

❷ In this trivial case, string formatting accomplishes the same result as concatentation.

❸ (userCount,) is a tuple with one element. Yes, the syntax is a little strange, but there's a good reason for it: it's unambiguously a tuple. In fact, you can always include a comma after the last element when defining a list, tuple, or

Mariaus Gedmino paskaitų konspektai



Nuorodos

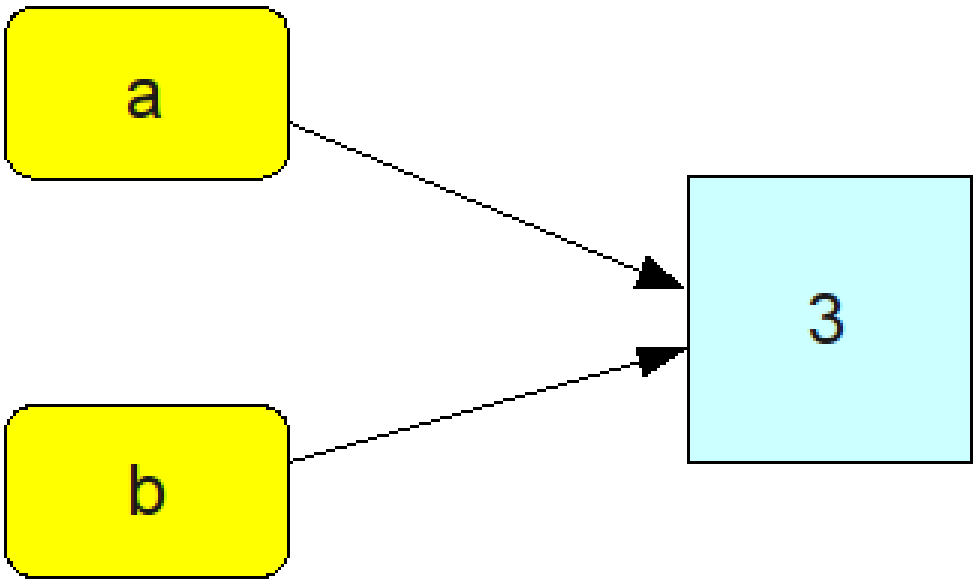
- Python bibliotekos dokumentacija (angl. *library reference*)
  http://docs.python.org/lib/lib.html
- Python įvadas (*tutorial*)
  http://docs.python.org/tut/tut.html
- Marko Pilgrimo knyga „Dive Into Python“ (nemokama)
  http://diveintopython.org/toc/index.html
- Mariaus Gedmino Python kurso puslapis:
  lietuviškos skaidrės, užduotys, pavyzdžiai
  http://gedmin.as/study/python/

Outline

1. Kursas
   - Įvadas
   - Darbo aplinka
   - Ištekliai
2. Python duomenų tipai
   - Apžvalga
   - Skaitiniai tipai, bool
   - Simbolių sekos, sąrašai, žodynai
3. Sakiniai

Objektai ir nuorodos

```
3          # objektas "trejetas"
a = 3      # dabar "a" yra nuoroda i objekta 3
b = a      # "b" rodo i "a" rodoma objekta (3)
```



Metodai ir funkcijos

```
>>> len('abc')
3
>>> 'abc'.upper()
'ABC'
>>> 'abaca'.count('a')
3
```

Svarbiausi Python tipai

```
>>> type(123)
<type 'int'>

>>> type(123.45)
<type 'float'>

>>> type("Labas, pasauli")
<type 'str'>

>>> type([1, "abc", 3.45])
<type 'list'>

>>> type({"vienas": "one", "du": "two"})
<type 'dict'>
```

Outline

1. Kursas
   - Įvadas
   - Darbo aplinka
   - Ištekliai
2. Python duomenų tipai
   - Apžvalga
   - Skaitiniai tipai, bool
   - Simbolių sekos, sąrašai, žodynai
3. Sakiniai

Operacijos su *int*

sudėtis	456 + 789	1245
atimtis	987 - 654	333
daugyba	7 * 9	63
dalyba	11 / 3	3
sveikųjų skaičių dalyba	11 // 3	3
dalybos liekana	11 % 3	2
absoliuti reikšmė	abs(-5)	5
kėlimas laipsniu	2 ** 10	1024

long

- long – didelių sveikųjų skaičių tipas

```
>>> 2 ** 1000
10715086071862673209484250490600018105614048117055336074437503883703510511249361224931983788156958581275946729175531468251871452856923140435984577574698574803934567774824230985421074605062371141877954182153046474983581941267398767559165543946077062914571196477686542167660429831652624386837205668069376L
>>> type(2 ** 1000)
<type 'long'>
```

- *long* elgiasi panašiai kaip *int*, su juo galima atlikti tokias pačias operacijas.

long

- long – didelių sveikųjų skaičių tipas

```
>>> 2 ** 1000
10715086071862673209484250490600018105614048117055336074437503883703510511249361224931983788156958581275946729175531468251871452856923140435984577574698574803934567774824230985421074605062371141877954182153046474983581941267398767559165543946077062914571196477686542167660429831652624386837205668069376L
>>> type(2 ** 1000)
<type 'long'>
```

- *long* elgiasi panašiai kaip *int*, su juo galima atlikti tokias pačias operacijas.

float

- *float* – trupmeninių skaičių tipas

```
>>> type(2.5)
<type 'float'>
```

- Galima atlikti operacijas kaip ir su *int* (išskyrus sveikų skaičių dalybą, liekanos skaičiavimą)
- Apvalinama su funkcija **round(x)**
- *float* skaičiai iš prigimties apytiksliai!

```
>>> 2.3
2.2999999999999998
>>> 100 * 11.0 - 100 * 2.2 * 5
-2.2737367544323206e-13
```

- *float* skaičių negalima tiesiog lyginti su nuliu ar vieno su kitu – reikia vertinti jų skirtumą.

float

- *float* – trupmeninių skaičių tipas

```
>>> type(2.5)
<type 'float'>
```

- Galima atlikti operacijas kaip ir su *int* (išskyrus sveikų skaičių dalybą, liekanos skaičiavimą)
- Apvalinama su funkcija **round(x)**
- *float* skaičiai iš prigimties apytiksliai!

```
>>> 2.3
2.2999999999999998
>>> 100 * 11.0 - 100 * 2.2 * 5
-2.2737367544323206e-13
```

- *float* skaičių negalima tiesiog lyginti su nuliu ar vieno su kitu – reikia vertinti jų skirtumą.

float

- *float* – trupmeninių skaičių tipas

```
>>> type(2.5)
<type 'float'>
```

- Galima atlikti operacijas kaip ir su *int* (išskyrus sveikų skaičių dalybą, liekanos skaičiavimą)
- Apvalinama su funkcija **round(x)**
- *float* skaičiai iš prigimties apytiksliai!

```
>>> 2.3
2.2999999999999998
>>> 100 * 11.0 - 100 * 2.2 * 5
-2.2737367544323206e-13
```

- *float* skaičių negalima tiesiog lyginti su nuliu ar vieno su kitu – reikia vertinti jų skirtumą.

float

- *float* – trupmeninių skaičių tipas

```
>>> type(2.5)
<type 'float'>
```

- Galima atlikti operacijas kaip ir su *int* (išskyrus sveikų skaičių dalybą, liekanos skaičiavimą)
- Apvalinama su funkcija **round(x)**
- *float* skaičiai iš prigimties apytiksliai!

```
>>> 2.3
2.2999999999999998
>>> 100 * 11.0 - 100 * 2.2 * 5
-2.2737367544323206e-13
```

- *float* skaičių negalima tiesiog lyginti su nuliu ar vieno su kitu – reikia vertinti jų skirtumą.

float

- *float* – trupmeninių skaičių tipas

```
>>> type(2.5)
<type 'float'>
```

- Galima atlikti operacijas kaip ir su *int* (išskyrus sveikų skaičių dalybą, liekanos skaičiavimą)
- Apvalinama su funkcija **round(x)**
- *float* skaičiai iš prigimties apytiksliai!

```
>>> 2.3
2.2999999999999998
>>> 100 * 11.0 - 100 * 2.2 * 5
-2.2737367544323206e-13
```

- *float* skaičių negalima tiesiog lyginti su nuliu ar vieno su kitu – reikia vertinti jų skirtumą.

Palyginimai

Reikšmes galima lyginti.

```
>>> 2 * 2 == 4
True
>>> 0.0 == 0
True
>>> 2 < 3
True
>>> 7 >= 8
False
```

None tipas

- Ypatingas tipas – tik viena galima reikšmė None

```
>>> type(None)
<type 'NoneType'>
```

- Reikšmė: „nėra duomenų“
- Tradiciškai lyginamas ne su ==, bet su operatoriumi is

```
>>> if a is None:
...     print "ND"
```

None tipas

- Ypatingas tipas – tik viena galima reikšmė None

```
>>> type(None)
<type 'NoneType'>
```

- Reikšmė: „nėra duomenų“
- Tradiciškai lyginamas ne su ==, bet su operatoriumi is

```
>>> if a is None:
...     print "ND"
```

None tipas

- Ypatingas tipas – tik viena galima reikšmė None

```
>>> type(None)
<type 'NoneType'>
```

- Reikšmė: „nėra duomenų“
- Tradiciškai lyginamas ne su ==, bet su operatoriumi is

```
>>> if a is None:
...     print "ND"
```

bool tipas

- Dvi reikšmės: True ir False
- Operacijos: and, or, not, ==, !=

```
>>> True and False
False
>>> True or False
True
>>> not False
True
>>> True == True
True
>>> True != True
False
```

bool tipas

- Dvi reikšmės: True ir False
- Operacijos: and, or, not, ==, !=

```
>>> True and False
False
>>> True or False
True
>>> not False
True
>>> True == True
True
>>> True != True
False
```

Outline

1. Kursas
   - Įvadas
   - Darbo aplinka
   - Ištekliai
2. Python duomenų tipai
   - Apžvalga
   - Skaitiniai tipai, bool
   - Simbolių sekos, sąrašai, žodynai
3. Sakiniai

Simbolių sekos (angl. *strings*)

- Užrašomos viengubose arba dvigubose kabutėse: "abc", ’de’.
- Naujos eilutės simbolis – „\n“: "Vienas\ndu".
- Operacijos

sudūrimas	"abc" + "de"	"abcde"
dauginimas	"abc" * 3	"abcabcabc"
ilgis	len("abcd")	4
šablonai	"vienas %s trys" % "du"	"vienas du trys"

- Galima skaityti iš klaviatūros su funkcija *raw_input()*

```
>>> a = raw_input()
>>> b = raw_input()
>>> print "%s ir %s" % (a, b)
```

Simbolių sekos (angl. *strings*)

- Užrašomos viengubose arba dvigubose kabutėse: "abc", ’de’.
- Naujos eilutės simbolis – „\n“: "Vienas\ndu".
- Operacijos

sudūrimas	"abc" + "de"	"abcde"
dauginimas	"abc" * 3	"abcabcabc"
ilgis	len("abcd")	4
šablonai	"vienas %s trys" % "du"	"vienas du trys"

- Galima skaityti iš klaviatūros su funkcija *raw_input()*

```
>>> a = raw_input()
>>> b = raw_input()
>>> print "%s ir %s" % (a, b)
```

Simbolių sekos (angl. *strings*)

- Užrašomos viengubose arba dvigubose kabutėse: "abc", ’de’.
- Naujos eilutės simbolis – „\n“: "Vienas\ndu".
- Operacijos

sudūrimas	"abc" + "de"	"abcde"
dauginimas	"abc" * 3	"abcabcabc"
ilgis	len("abcd")	4
šablonai	"vienas %s trys" % "du"	"vienas du trys"

- Galima skaityti iš klaviatūros su funkcija *raw_input()*

```
>>> a = raw_input()
>>> b = raw_input()
>>> print "%s ir %s" % (a, b)
```

Simbolių sekos (angl. *strings*)

- Užrašomos viengubose arba dvigubose kabutėse: "abc", ’de’.
- Naujos eilutės simbolis – „\n“: "Vienas\ndu".
- Operacijos

sudūrimas	"abc" + "de"	"abcde"
dauginimas	"abc" * 3	"abcabcabc"
ilgis	len("abcd")	4
šablonai	"vienas %s trys" % "du"	"vienas du trys"

- Galima skaityti iš klaviatūros su funkcija *raw_input()*

```
>>> a = raw_input()
>>> b = raw_input()
>>> print "%s ir %s" % (a, b)
```

Sąrašai

- Reikšmių sąrašas (angl. *list*). Užrašomas laužtiniuose skliaustuose:
  [3, 4, 5, 'abc', None, 0.0]
- Paprasčiausios operacijos

sudūrimas	[1, 2] + [3, 4]	[1, 2, 3, 4]
dauginimas	[1, 2] * 3	[1, 2, 1, 2, 1, 2]
indeksavimas	['a', 'b', 'c'][2]	'c'
ilgis	len([2, 5, 7])	3

```
>>> l = [1, 2, 3]
>>> l[2] = 4
>>> l[2]
4
```

Sąrašai

- Reikšmių sąrašas (angl. *list*). Užrašomas laužtiniuose skliaustuose:
  [3, 4, 5, 'abc', None, 0.0]
- Paprasčiausios operacijos

sudūrimas	[1, 2] + [3, 4]	[1, 2, 3, 4]
dauginimas	[1, 2] * 3	[1, 2, 1, 2, 1, 2]
indeksavimas	['a', 'b', 'c'][2]	'c'
ilgis	len([2, 5, 7])	3

```
>>> l = [1, 2, 3]
>>> l[2] = 4
>>> l[2]
4
```

Sąrašai

- Reikšmių sąrašas (angl. *list*). Užrašomas laužtiniuose skliaustuose:
  [3, 4, 5, 'abc', None, 0.0]
- Paprasčiausios operacijos

sudūrimas	[1, 2] + [3, 4]	[1, 2, 3, 4]
dauginimas	[1, 2] * 3	[1, 2, 1, 2, 1, 2]
indeksavimas	['a', 'b', 'c'][2]	'c'
ilgis	len([2, 5, 7])	3

```
>>> l = [1, 2, 3]
>>> l[2] = 4
>>> l[2]
4
```

Žodynai

Žodynas (angl. *dict*) saugo reikšmių poras.
Užrašomas riestiniuose skliaustuose.
{1: 'vienas', 2: 'du', None: 'nieko'}

```
>>> d = {'vienas': 1, 'du': 2, 'nieko': None}
>>> d['du']
2
>>> d['keturi'] = 4
>>> d['keturi']
4
```

Kaip pakeisti reikšmės tipą?

Galima keisti reiškmės tipą, pavyzdžiui, iš skaičiaus padarant simbolių seką arba atvirkščiai.

```
>>> 123
123
>>> str(123)
'123'
>>> 123 + 456
579
>>> int(str(123) + str(456))
123456
>>> list("abc")
['a', 'b', 'c']
```

Bool reikšmė

- Bet kuri reikšmė gali būti paversta bool (True/False) reikšme.
- Tiesiogiai tai galima padaryti išraiška *bool(x)*
- Dauguma reikšmių keičiamos į True, išskyrus keletą: 0, 0.0, "", False, None, []

```
>>> bool(0)
False
>>> bool(1)
True
>>> bool('')
False
>>> bool('0')
True
```

Bool reikšmė

- Bet kuri reikšmė gali būti paversta bool (True/False) reikšme.
- Tiesiogiai tai galima padaryti išraiška *bool(x)*
- Dauguma reikšmių keičiamos į True, išskyrus keletą: 0, 0.0, "", False, None, []

```
>>> bool(0)
False
>>> bool(1)
True
>>> bool('')
False
>>> bool('0')
True
```

Bool reikšmė

- Bet kuri reikšmė gali būti paversta bool (True/False) reikšme.
- Tiesiogiai tai galima padaryti išraiška *bool(x)*
- Dauguma reikšmių keičiamos į True, išskyrus keletą: 0, 0.0, "", False, None, []

```
>>> bool(0)
False
>>> bool(1)
True
>>> bool('')
False
>>> bool('0')
True
```

Loginiai operatoriai

Loginiai operatoriai *and* ir *or* iš tiesų grąžina ne bool reikšmę, bet vieną iš argumentų. Į kitą argumentą gali būti net nežiūrima.

```
>>> 1 and ''
''
>>> 1 and 2
2
>>> '' and 2
''
>>> 2 or 1 / 0
2
```

Išvedimo sakinys

```
>>> print 123
123
>>> print "abc"
abc
>>> print "abc", 123, "de"
abc 123 de
>>> print "abc\\nde"
abc
de
```

Sąlyginis sakinys

- Sąlyginio sakinio pavyzdys:

```
if a == 2:
    b = 3
    c = 4
else:
    b = 5
    c = 6
```

- Iškart po „if“ užrašoma sąlyga, kuri vertinama kaip *bool* reikšmė.
- *else* dalies gali nebūti.
- Nereikia begin/end ar {}.
- Sakiniai if’o viduje turi būti pastumti per 4 tarpelius.

Sąlyginis sakinys

- Sąlyginio sakinio pavyzdys:

```
if a == 2:
    b = 3
    c = 4
else:
    b = 5
    c = 6
```

- Iškart po „if“ užrašoma sąlyga, kuri vertinama kaip *bool* reikšmė.
- *else* dalies gali nebūti.
- Nereikia begin/end ar {}.
- Sakiniai if’o viduje turi būti pastumti per 4 tarpelius.

Sąlyginis sakinys

- Sąlyginio sakinio pavyzdys:

```
if a == 2:
    b = 3
    c = 4
else:
    b = 5
    c = 6
```

- Iškart po „if“ užrašoma sąlyga, kuri vertinama kaip *bool* reikšmė.
- *else* dalies gali nebūti.
- Nereikia begin/end ar {}.
- Sakiniai if’o viduje turi būti pastumti per 4 tarpelius.

Sąlyginis sakinys

- Sąlyginio sakinio pavyzdys:

```
if a == 2:
    b = 3
    c = 4
else:
    b = 5
    c = 6
```

- Iškart po „if“ užrašoma sąlyga, kuri vertinama kaip *bool* reikšmė.
- *else* dalies gali nebūti.
- Nereikia begin/end ar {}.
- Sakiniai if’o viduje turi būti pastumti per 4 tarpelius.

Sąlyginis sakinys

- Sąlyginio sakinio pavyzdys:

```
if a == 2:
    b = 3
    c = 4
else:
    b = 5
    c = 6
```

- Iškart po „if“ užrašoma sąlyga, kuri vertinama kaip *bool* reikšmė.
- *else* dalies gali nebūti.
- Nereikia begin/end ar {}.
- Sakiniai if’o viduje turi būti pastumti per 4 tarpelius.

for ciklas

for ciklas perbėga per visus sąrašo elementus.

```
>>> for i in [3, 4, 5, 'a']:
...     print i * 3
9
12
15
aaa
```

while ciklas

```
>>> a = 1
>>> while a < 5:
...     print a
...     a = a * 2
>>> print a
```

Tuščias sakinys *pass*

Naudojamas dėl sintaksinių priežasčių tose vietose, kur būtinas sakinys.

```
>>> pass

>>> for i in range(5):
...     pass

>>> if a:
...     pass  # nedarykite taip, naudokite "not"
... else:
        a = a + 5
```

Pradžiai pakaks

Užduotys laukia.