Programavimo kalba

Python

trečioji paskaita

Marius Gedminas
<mgedmin@b4net.lt>

http://mg.b4net.lt/python/

Matematikos ir Informatikos fakultetas

2006 ruduo

Naujiena

Python 2.5 pasirodė prieš dvi dienas

(2006 m. rugsėjo 19 d.)

Trumpas kartojimas

```
# Komentaras
x = y
fn(x)
print x
import module
from module import name
```

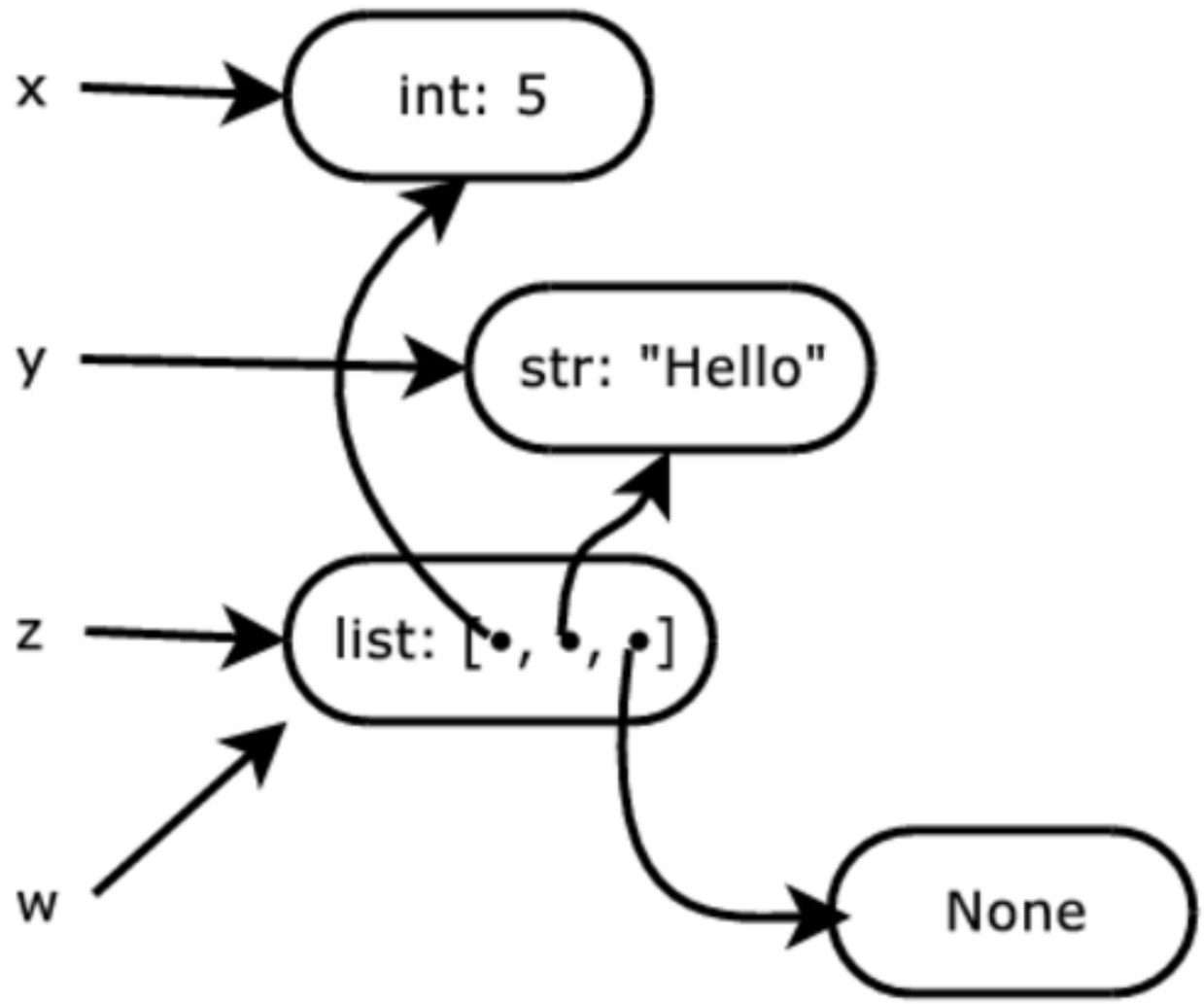
Trumpas kartojimas

```
if x:
    y
elif q:
    w
else:
    z

while x:
    break
```

```
for x in some_list:
    continue

def fn(args):
    return value

class Cls(base):
    def method(self, args):
        self.attr = args
```

```
#!/usr/bin/env python
# coding: -*- utf-8 -*-
"""
Ši programa daro ...

Autorius: Vardas Pavardė <e@mail.as>
"""

import x

def fn(...):
    """Ši funkcija skaičiuoja ..."""

if __name__ == '__main__':
    main()
```

Vardai ir objektai


x
y
z
w
int: 5
str: "Hello"
list: [•, •, •]
None

Konstanta None

Duomenų tipai

int, float, complex,

str, unicode, bool

list, dict, set

Python turi daug patogių dviračių, apie kuriuos verta žinoti

Simbolių eilutės

x = "Mano namas buvo du"

```
y = x.replace("namas", "batai")
```

str klasės metodai

lower()

```
'EiNu NaMo'.lower() == 'einu namo'
```

upper()

```
'EiNu NaMo'.upper() == 'EINU NAMO'
```

title()

```
'EiNu NaMo'.title() == 'Einu Namo'
```

capitalize()

```
'einu namo'.capitalize() == 'Einu namo'
```

center(width[, fillchar])

'abc'.center(9, '-') == '---abc---'

ljust(width[, fillchar])

'abc'.ljust(9, '-') == 'abc------'

rjust(width[, fillchar])

'abc'.rjust(9, '-') == '------abc'

strip([chars])

```
' abc '.strip() == 'abc'
```

lstrip([chars])

```
' abc '.lstrip() == 'abc '
```

rstrip([chars])

```
' abc '.rstrip() == ' abc'
```

count(substr[, start[, end]])

'lia lia lia kva'.count('lia') == 3

find(substr[, start[, end]])

'xyz'.find('z') == 2, 'xyz'.find('q') == -1

index(substr[, start[, end]])

meta IndexError jei neranda

rfind(substr[, start[, end]])

ieško dešiniausio

rindex(substr[, start[, end]])

ieško dešiniausio

startswith(suffix[, start[, end]])

ar simbolių eilutė prasideda suffix?

endswith(suffix[, start[, end]])

ar simbolių eilutė baigiasi suffix?

decode(encoding[, errors])

verčia nurodytą koduotę į Unikodą

encode(encoding[, errors])

verčia Unikodą į nurodytą koduotę

expandtabs([tabsize])

keičia TAB simbolius tarpais

isalnum()

ar tai raidė/skaitmuo?

isalpha()

ar tai raidė?

isdigit()

ar tai skaitmuo?

islower()

ar tai mažoji raidė?

isupper()

ar tai didžioji raidė?

isspace()

ar tai tarpas?

istitle()

ar tai pradinė raidė? (Unikode yra simboliai 'DZ', 'Dz' ir 'dz')

Šie metodai veikia ir jei len(s) > 1:

'1234'.isdigit() == True

'123a'.isdigit() == False

join(sequence)

', '.join(['a', 'b', 'c']) == 'a, b, c'

split([separator])

'a, b, c'.split(',') == ['a', ' b', ' c']

'a b\n c '.split() == ['a', 'b', 'c']

splitlines([keep])

'a\nb\n'.splitlines() == ['a', 'b']

'a\nb\n'.splitlines(True) == ['a\n', 'b\n']

replace(old, new[, count])

'xyzzy'.replace('y', 'q') == 'xqzzq'

Sąrašai

$$x = [1, 2, 3, 4]$$

list klasės metodai

append(object)

x = [1, 2]; x.append(5); x == [1, 2, 5]

extend(seq)

x = [1]; x.extend([3, 4]); x == [1, 3, 4]

insert(index, object)

x = [1, 2]; x.insert(0, 5); x == [5, 1, 2]

x = [1, 2, 3, 4]; x.insert(-1, 5); x == [1, 2, 3, 5, 4]

count(value)

x = [1, 2, 3, 2, 1]; x.count(1) == 2

index(value)

x = [5, 5, 1, 2, 3]; x.index(1) == 2

x = [5, 5, 1, 2, 3]; x.index(6) ->

IndexError

value in a_list

2 in [1, 2, 3] == True

4 in [1, 2, 3] == False

remove(value)

x = [5, 5, 1, 2, 3, 1]; x.remove(1); x == [5, 5, 2, 3, 1]

pop([index])

x = [1, 2, 3, 4]; x.pop() == 4; x == [1, 2, 3]

x = [1, 2, 3, 4]; x.pop(0) == 1; x == [2, 3, 4]

a_list[start:end], a_list[start:end:step]

x = ['a', 'b', 'c', 'd', 'e']; x[1:-1] == ['b', 'c', 'e']

x = ['a', 'b', 'c']; x[::-1] == ['c', 'b', 'a']

x = ['a', 'b', 'c']; x[1:2] = ['q', 'w']; x == ['a', 'q', 'w', 'c']

x = ['a', 'b', 'c']; del x[1:2]; x == ['a', 'c']

Žodynai

```
x = {'a': 1, 'b': 2}
```

keys()

grąžina visus raktus

values()

grąžina visas reikšmes

items()

grąžina raktų ir reikšmių poras

has_key(value)

x.has_key(y) == y in x

get(value[, default])

x.get(k, d) grąžins d jei rakto k nėra žodyne; x[k] mes KeyError, jei rakto nėra žodyne

setdefault(value, default)

x.setdefault(k, d) grąžins d jei rakto k nėra žodyne, o taip pat įdės reikšmę į žodyną; jei raktas jau yra žodyne, setdefault grąžins x[k] ir žodyne nieko nekeis

pop(key[, default])

x.pop(k) grąžins x[k] ir išmes k iš x

popitem()

x.popitem() parinks kurį nors raktą k ir grąžins (k, x.pop(k))

clear()

išmeta visus raktus

update(dict)

x.update(y) == for k in y: x[k] = y[k]

copy()

grąžins žodyno kopiją

fromkeys(keys)

sukurs naują žodyną su nurodytais raktais

Vidinės funkcijos

```
x = min(a, b)
```

```
y = min([1, 2, 3, 4, 5, 6, 7])
```

Kur rasti dokumentaciją?

```
>>> help('modulis')
```

$ pydoc modulis

http://www.python.org/doc/

__specialūs_vardai__

Nedarykite!

```
x = s.__len__()
s = w.__str__()
c = a.__add__(b)
```

Darykite

```
x = len(s)
s = str(w)
c = a + b
```

Standartinė biblioteka ("batteries included")

Standartinėje Python bibliotekoje yra daug naudingų modulių

Trumpai juos apžvelgsiu, kad įsivaizduotumėte, ko ten galima tikėtis

Pilna dokumentacija

http://www.python.org/doc/

pydoc moduliovardas

Modulis '__builtin__'

visuomet prieinamos funkcijos

jo nereikia importuoti!

True, False

loginės konstantos

None

konstanta "nėra reikšmės"

abs(x)

|x|

chr(i)

simbolis, kurio kodas i

ord(c)

simbolio c kodas

unichr(i)

unikodinis simbolis, kurio kodas i

dir(x)

objekto x atributų sąrašas

divmod(a, b)

(a/b, a%b)

getattr(x, a[, default])

objekto x atributas a

hasattr(x, a)

ar objektas x turi atributą a?

hash(x)

objekto x hashas

id(x)

objekto x identitetas

isinstance(x, class)

ar objektas x yra klasės class?

issubclass(c1, c2)

ar c1 yra c2 poklasė?

len(l)

sekos ilgis

min(x), min(a, b)

mažiausias elementas

max(x), max(a, b)

didžiausias elementas

pow(x, y[, z])

kėlimas laipsniu

range(n), range(a, b[, step])

skaičių seka

raw_input([prompt])

įvedimas iš klaviatūros

repr(x)

objekto x reprezentacija

str(x)

objekto x vertimas simbolių eilute

int(x[, base])

simbolių eilutės vertimas sveiku skaičiumi

float(x)

simbolių eilutės vertimas skaičiumi

sum(l[, start])

sekos suma

zip(a, b)

sekų sutraukimas

open(fn, mode), file(fn, mode)

failo atidarymas

bool, str, list, int, float, file

iš tiesų tai klasės, o ne funkcijos

Modulis 'sys'

sisteminiai dalykėliai

sys.argv

komandų eilutės argumentų sąrašas

sys.exit([kodas])

išėjimas iš programos

sys.path

kelias, kuriame ieškoma modulių

sys.version

Python versija

sys.stdin/sys.stdout/sys.stderr

įvedimo/išvedimo įrenginiai

sys.argv

```
print " ".join(sys.argv[1:])
```

sys.exit([kodas])

```
sys.exit(1)
```

sys.path

```
sys.path.append('/mano/moduliai')
```

sys.version

```
print sys.version
```

sys.stdin/sys.stdout/sys.stderr

```
print >> sys.stderr, "klaida: ..."
```

Modulis 'os'

darbas su operacine sistema

os.listdir(dir)

katalogo skaitymas

os.mkdir(dir)

katalogo kūrimas

os.system(cmd), os.popen(cmd, mode)

komandos paleidimas

os.walk(dir)

katalogų medžio apėjimas

os.environ

aplinkos kintamieji

os.listdir(dir)

```
for fn in os.listdir(dir):
    print fn
```

os.mkdir(dir)

```
os.mkdir('subdir')
```

os.system(cmd), os.popen(cmd, mode)

```
os.system("make")
for line in os.popen("ls"):
    print line
```

os.walk(dir)

```
for dir, dirs, files in os.walk(os.curdir):
    print "%s: %s files, %s subdirs" % (
        dir, len(dirs), len(files)
```

os.environ

```
print "HOME =", os.environ["HOME"]
```

Modulis 'os.path'

darbas su keliais

Unix: /path/to/some/file

Windows: c:\path\to\some\file

MacOS 9: Neturiu:Zalio:Supratimo

os.path.join(path, name[, ...])

kelių sujungimas

os.path.abspath(path)

absoliutus kelias

os.path.split(path)

kelio dalinimas

os.path.basename(path)

tik failo vardas

os.path.dirname(path)

tik katalogas

os.path.splitext(path)

failo vardo dalinimas

os.path.walk(dir, fn, arg)

katalogų medžio apėjimas

os.path.sep

skirtukas

os.path.pardir

lipimas aukštyn

os.path.join(path, name[, ...])

```
>>> os.path.join('foo/', 'bar', 'baz')
'foo/bar/baz'
```

os.path.abspath(path)

```
>>> os.path.abspath('foo')
'/home/mg/foo'
```

os.path.split(path)

```
>>> os.path.split('/path/to/fn.py')
('/path/to', 'fn.py')
```

os.path.basename(path)

```
>>>
os.path.basename('/path/to/fn.py')
'fn.py'
```

os.path.dirname(path)

```
>>> os.path.dirname('/path/to/fn.py')
'/path/to'
```

os.path.splitext(path)

```
>>> os.path.splitext('/path/to/fn.py')
('/path/to/fn', '.py')
```

os.path.walk(dir, fn, arg)

```
def callback(arg, dir, files):
    print "%s: %s files and subdirs" % (
        dir, len(files)
os.path.walk(callback, None)
```

os.path.sep

```
>>> os.path.sep
'/'
```

os.path.pardir

```
>>> os.path.pardir
'..'
```

Modulis 'math'

matematika

math.exp(x)

eksponentė

math.log(x[, base])

logaritmai

math.hypot(x, y)

atstumas, t.y. sqrt(x^2+y^2)

math.sin(x)/math.cos(x)/math.tan(x)

trigonometrija

math.e, math.pi

konstantos e, pi

Modulis 'random'

atsitiktiniai skaičiai

random.randrange(n)

atsitiktinis sveikas skaičius nuo (0 <= x < n)

random.randrange(a, b)

a <= x < b

random.choice(l)

atsitiktinis sekos elementas

random.shuffle(l)

išmaišo masyvą vietoje

random.sample(l, n)

atsitiktinė imtis (n elementų)

random.gauss(mu, sigma)

Gauso pasiskirstymas (vidurkis, std. nuokrypis)

random.uniform(a, b)

atsitiktinis realus skaičius (a <= x < b)

random.seed(n)

pasikartojančios sekos

random.Random

nepriklausomas generatorius

Modulis 'sets'

aibės

sets.Set([l])

aibė

set([l])

aibė (Python 2.4)

set1.add(x)

elemento pridėjimas

x in set1

patikrinimas

set1 | set2

sąjunga

set1 & set2

sankirta

set1 - set2

skirtumas

set1 ^ set2

simetrinis skirtumas

set1 <= set2

poaibis?

Kiti moduliai

StringIO, cStringIO

įvedimas/išvedimas į simbolių eilutę

datetime

data ir laikas

time

senas modulis darbui su laiku

optparse

komandų eilutės argumentai

glob

failų paieška pagal šabloną

difflib

skirtumų paieška

textwrap

teksto lankstymas

pickle, cPickle

objektų „marinavimas“

copy

duomenų struktūrų kopijavimas

pprint

gražus objektų vaizdavimas

string

pasenęs modulis, nenaudokite

re

reguliarios išraiškos (regexpai)

struct

dvejetainės duomenų struktūros

curses

tekstinio režimo UI

tempfile

laikini failai ir katalogai

shutil

failų/katalogų medžių kopijavimas/išmetimas

locale

programų lokalizacija

gettext

lokalizacija: tekstų vertimas

logging

darbo žurnalas

threading

darbas su gijomis

Queue

darbų eilė gijoms

zipfile

darbas su ZIP failais

readline

įvedimo eilutė su istorija

csv

Comma Separated Values failai

email

elektroninio pašto pranešimų konstravimas

inspect

gyvų objektų apžvalga

linecache

source kodo eilučių ištraukimas

traceback

iškvietimų steko formatavimas

decimal

aritmetika su dideliais realiais skaičiais

cmath

kompleksinių skaičių matematika

array

greiti masyvai

itertools

iteratorių kombinacijos

fileinput

įvedimas iš kelių failų

subprocess

programų paleidimas ir valdymas

socket

žemo lygio darbas su tinklu

pdb

Python debugeris

hotshot

profiliatorius

cgi

CGI skriptų rašymas

urllib

tinklo resursų parsiuntimas

httplib

HTTP protokolo klientas

imaplib

IMAP protokolo klientas

smtplib

SMTP protokolo klientas

xmlrpclib

XML-RPC protokolas

xml.dom.minidom

XML parseris (DOM)

xml.dom.pulldom

XML parseris (taupus)

audioop

garso duomenų apdorojimas

md5

MD5 kriptografinė hash funkcija

sha

SHA-1 kriptografinė hash funkcija

Tkinter

paprasta grafinė vartotojo aplinka

distutils

programų paketų ruošimas

unittest, doctest

automatizuoti testai

ConfigParser

.INI stiliaus konfigūracijos failai

ir tai ne viskas

http://www.python.org/doc/