Programavimo kalba

Python

trečioji paskaita

Marius Gedminas
<mgedmin@b4net.lt>

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

Matematikos ir Informatikos fakultetas

2005 ruduo

Pagrindai

žvilgsnis iš arčiau

Atėjus iš kitų programavimo kalbų dažnai norisi išradinėti dviratį

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 žodyno nieko nekeis

pop(key)

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/