

Kurs języka Python

Wykład 3.

Marcin Młotkowski

19 października 2009

- 1 Listy, cd
- 2 Moduły
- 3 Funkcje
- 4 Listy, ponownie
- 5 Zakończenie

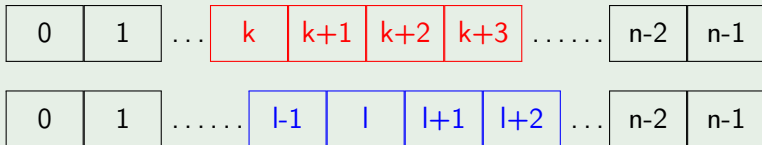
Operacje na listach

Wymiana elementów

```
lista = [1,2,3]
lista[1] = 5      # [1, 5, 3]
lista[1:] = [2,3,4] # [1,2,3,4]
```

Zamiana podlisty

```
lista[zakres] = innaLista
```



Zamiana podlisty

Przykłady

```
lista = [0,1,2,3]
lista[1:3] = ["jeden"] # [0, 'jeden', 3]
lista[1:1] = [1] # [0, 1, 'jeden', 3]
lista[2:3] = [2] # [0, 1, 2, 3]
```

Przykład ze slicingiem

```
lista = [0, 1, 2, 3]
lista[::2] = [4, 5]
>>> [4, 1, 5, 3]
```

Zamiana podlisty

Przykłady

```
lista = [0,1,2,3]
lista[1:3] = ["jeden"] # [0, 'jeden', 3]
lista[1:1] = [1] # [0, 1, 'jeden', 3]
lista[2:3] = [2] # [0, 1, 2, 3]
```

Przykład ze slicingiem

```
lista = [0, 1, 2, 3]
lista[::2] = [4, 5]
>>> [4, 1, 5, 3]
```

Dodawanie i usuwanie elementów

Przykłady

```
lista = [0, 1, 2, 3]
lista[ len(lista): ] = [4, 5, 6]
>>> [0, 1, 2, 3, 4, 5, 6]
lista = [0, 1, 2, 3, 4, 5]
lista[4:6] = []
>>> [0, 1, 2, 3]
```

Dodawanie i usuwanie elementów

Przykłady

```
lista = [0, 1, 2, 3]
lista[ len(lista): ] = [4, 5, 6]
>>> [0, 1, 2, 3, 4, 5, 6]
lista = [0, 1, 2, 3, 4, 5]
lista[4:6] = []
>>> [0, 1, 2, 3]
```

Instrukcja `del`

Przykłady

```
lista = [ 'żółty', 'zielony', 'czerwony', 'niebieski' ]
```

```
del lista[3]
```

```
>>> [ 'żółty', 'zielony', 'czerwony' ]
```

```
del lista[1:]
```

```
>>> [ 'żółty' ]
```

Instrukcja `del`

Przykłady

```
lista = [ 'żółty', 'zielony', 'czerwony', 'niebieski' ]  
del lista[3]  
>>> [ 'żółty', 'zielony', 'czerwony' ]  
del lista[1:]  
>>> [ 'żółty' ]
```

del dla słowników

Przykład

```
htmlCol = { 'NavyBlue' : (0,0,128), 'turquoise' : (64,224,208) }  
del htmlCol['turquoise']
```

Operacje na listach

Inne operacje

append, extend, insert, remove, pop, index, count, sort, reverse

Przykłady

```
lista = [0, 1, 2, 3]  
lista.reverse() # Nie zwraca wyniku
```

Operacje na listach

Inne operacje

append, extend, insert, remove, pop, index, count, sort, reverse

Przykłady

```
lista = [0, 1, 2, 3]
```

```
lista.reverse() # Nie zwraca wyniku
```

Moduł

modul.py

```
"""Moduł z kilkoma prostymi funkcjami"""  
def fib(n):  
    """n-ty wyraz ciągu Fibonacciego"""  
    if (n < 2): return 1  
    else: return fib(n - 1) + fib(n - 2)
```

Użycie modułu

main.py

```
import modul  
print modul.fib(10)
```

Import nazw

main.py

```
from modul import fib  
# from modul import *  
print fib(10)
```

Kącik porad: testowanie modułów

modul.py

```
"""Moduł z kilkoma prostymi funkcjami"""  
def fib(n):  
    """n-ty wyraz ciągu Fibonacciego"""  
    if (n < 2): return 1  
    else: return fib(n - 1) + fib(n - 2)  
  
if __name__ == "__main__":  
    for i in range(5): print i, fib(i)
```

Kącik porad: testowanie modułów

modul.py

```
"""Moduł z kilkoma prostymi funkcjami"""  
def fib(n):  
    """n-ty wyraz ciągu Fibonacciego"""  
    if (n < 2): return 1  
    else: return fib(n - 1) + fib(n - 2)  
  
if __name__ == "__main__":  
    for i in range(5): print i, fib(i)
```

Dokumentowanie modułu

modul.py

```
"""Moduł z kilkoma prostymi funkcjami"""  
def fib(n):  
    """n-ty wyraz ciągu Fibonacciego"""  
    if (n < 2): return 1  
    else: return fib(n - 1) + fib(n - 2)
```

Dokumentowanie modułu

```
>>> import modul
>>> print modul.__doc__
Moduł kilku prostych funkcji
>>> print modul.fib.__doc__
n-ty wyraz ciągu Fibonacciego
```

Dokumentowanie modułu

```
>>> import modul  
>>> help(modul)
```

Rezultat

NAME

modul - Moduł kilku prostych funkcji

FILE

/home/marcinm/python/modul.py

FUNCTIONS

fib(n)

n-ty wyraz ciągu Fibonacciego

Dokumentowanie modułu

```
>>> import modul  
>>> help(modul)
```

Rezultat

NAME

modul - Moduł kilku prostych funkcji

FILE

/home/marcinm/python/modul.py

FUNCTIONS

fib(n)

n-ty wyraz ciągu Fibonacciego

Funkcje

Przykład użycia funkcji

```
def calka(f, a, b):  
    krok, suma, x = .1, 0, a  
    while x + krok < b:  
        suma += f(x)*krok  
        x += krok  
    return suma  
  
def fun(n): return n * n  
  
print calka(fun, 0, 5)
```

Funkcje, cd

Inne przykłady

```
def square(n): return n*n
```

```
def double(n): return 2 * n
```

```
funList = [ fib, double ]
```

```
for f in funList:  
    print f(10)
```

Lambda funkcje

```
double = lambda x: 2*x  
square = lambda x: x*x  
funList = [ double, square ]  
print calka(square, 0, 10)
```

Lambda funkcje

```
double = lambda x: 2*x
```

```
square = lambda x: x*x
```

```
funList = [ double, square ]  
print calka(square, 0, 10)
```

Lambda funkcje

```
double = lambda x: 2*x  
square = lambda x: x*x  
funList = [ double, square ]  
print calka(square, 0, 10)
```

Lambda funkcje, cd

```
funList = [ lambda x: 2*x, lambda x: x*x ]  
print calka(lambda x: x*x, 0, 10)
```

Dwuargumentowe funkcje lambda

```
f = lambda x, y: 2*x + y
```

Operacje na listach

Stałe

```
lista = range(100)

def fun(n): return n % 2 == 0

print filter(fun, lista)
print map(lambda x: 2*x, lista)
print reduce(lambda x, y: x + y, lista, 0)
```

Operacje na listach

Stałe

```
lista = range(100)
```

```
def fun(n): return n % 2 == 0
```

```
print filter(fun, lista)
```

```
print map(lambda x: 2*x, lista)
```

```
print reduce(lambda x, y: x + y, lista, 0)
```

Operacje na listach

Stałe

```
lista = range(100)
```

```
def fun(n): return n % 2 == 0
```

```
print filter(fun, lista)
```

```
print map(lambda x: 2*x, lista)
```

```
print reduce(lambda x, y: x + y, lista, 0)
```

Operacje na listach

Stałe

```
lista = range(100)

def fun(n): return n % 2 == 0

print filter(fun, lista)
print map(lambda x: 2*x, lista)
print reduce(lambda x, y: x + y, lista, 0)
```

Operacje na listach

Stałe

```
lista = range(100)

def fun(n): return n % 2 == 0

print filter(fun, lista)
print map(lambda x: 2*x, lista)
print reduce(lambda x, y: x + y, lista, 0)
```

Listy składane

Przykłady

```
lista = range(10)  
[ 2 * x for x in lista ]
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```
[ (x, x*x*x) for x in lista if x % 3 == 0 ]
```

```
[(0, 0), (3, 27), (6, 216), (9, 729)]
```

Listy składane

Przykłady

```
lista = range(10)  
[ 2 * x for x in lista ]
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```
[ (x, x*x*x) for x in lista if x % 3 == 0 ]
```

```
[(0, 0), (3, 27), (6, 216), (9, 729)]
```

Listy składane

Przykłady

```
lista = range(10)  
[ 2 * x for x in lista ]
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```
[ (x, x*x*x) for x in lista if x % 3 == 0 ]
```

```
[(0, 0), (3, 27), (6, 216), (9, 729)]
```

Listy składane

Przykłady

```
lista = range(10)  
[ 2 * x for x in lista ]
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```
[ (x, x*x*x) for x in lista if x % 3 == 0 ]
```

```
[(0, 0), (3, 27), (6, 216), (9, 729)]
```

Listy składane, dalsze przykłady

Przetwarzanie list stringów

```
lista = [ "mOnty", "pyTHon's", "FlyinG", "circus" ]
```

```
lista = [ e[0].upper() + e[1:].lower() for e in lista ]
```

Listy składane, dalsze przykłady

Przetwarzanie list stringów

```
lista = [ "mOnty", "pyTHon's", "FlyinG", "circus" ]
```

```
lista = [ e[0].upper() + e[1:].lower() for e in lista ]
```

Listy składane zagnieżdżone

Kolejne potęgi dwójki

```
a = [1, 8, 64]
```

```
b = [1, 2, 3]
```

```
print [ x << y for x in a for y in b]
```

Koniec

