Kurs rozszerzony języka Python

Wykład 2.

Marcin Młotkowski

13 października 2023

Plan wykładu

Plan wykładu

Literały

Stałe (pangramy)

```
'Mężny bądź, chroń pułk twój i sześć flag.'
"Zwölf große Boxkämpfer jagen Viktor quer \
über den Sylter Deich."
```

Literały

Stałe (pangramy)

```
'Mężny bądź, chroń pułk twój i sześć flag.'
"Zwölf große Boxkämpfer jagen Viktor quer \
über den Sylter Deich."
```

Długie napisy

```
"""'Tak nám zabili Ferdinanda,' řekla posluhovačka
panu Švejkovi, který opustiv před léty vojenskou službu,
když byl definitivně prohlášen vojenskou lékařskou
komisí za blba, živil se prodejem psů, ošklivých
nečistokrevných oblud, kterým padělal rodokmeny."""
```

Formatowanie: old-style

```
"dwa plus dwa to %s czyli %i" % ('cztery', 4)
```

Niezalecany

New-style formatting

```
"pi to {0:.2f}, a 'e' to {1}".format(math.pi, math.e)
```

New-style formatting

```
"pi to {0:.2f}, a 'e' to {1}".format(math.pi, math.e)
```

```
"pi to 3.14, a 'e' to 2.718281828459045"
```

f-strings

```
f"Wyrażenie w nawiasach: {2 * math.e}"
```

```
f"\u03C0 = {math.pi:.10f}"
```

Rożne wyrażenia do budowania stringów

```
('Ala' 'ma' "kota")
'Ala' + 'ma' + "kota"

"*" * 30
```

Funkcje i metody

```
"małe".upper()

'DUŻE'.lower()

sorted('To jest napis')
```

Szablony

```
from string import Template

szablon = Template('Nazywam się $imie.')
szablon.substitute(imie="Marcin")
```

Przetwarzanie napisów

```
"igła" in "stóg siana"
```

Na przykład

```
if "igła" in "W tym stogu siana jest jedna igła.":
    print("Jeest!!!")
```

Przetwarzanie fragmentów stringów

```
'informatyka'[2:5]
'informatyka'[3:]
'informatyka'[:4]
'informatyka'[:-3]
'informatyka'[::2]
```

Przetwarzanie fragmentów stringów

```
'informatyka'[2:5]
'informatyka'[3:]
'informatyka'[:4]
'informatyka'[:-3]
'informatyka'[::2]
'informatyka'[::-1]
```

Ogólna zasada

Stringi są niemutowalne, tj. nie można ich modyfikować, można tylko tworzyć nowe.

Tu tworzone są nowe stringi:

```
"string".replace("i", "o")
"Python" + "jest" + "super"
```

Strumienie

String jako strumień

```
import io

tekst = io.String()
tekst.write("Początek\n")
tekst.write("Koniec")
tekst.getvalue()

# zwolnienie pamięci
tekst.close()
```

Sekwencje specjalne w stringach

```
print("To\t jest\t długi\n\"tekst\"")
```

Sekwencje specjalne w stringach

```
print("To\t jest\t długi\n\"tekst\"")
```

Neutralizacja znaków sterujących, wersja 1.

```
print("To\\t jest\\t długi\\n\\tekst\\")
```

Sekwencje specjalne w stringach

```
print("To\t jest\t długi\n\"tekst\"")
```

Neutralizacja znaków sterujących, wersja 1.

```
print("To\\t jest\\t długi\\n\\tekst\\")
```

Neutralizacja znaków sterujących, wersja 2.: stringi 'surowe' (raw)

```
print(r"To\t jest\t długi\n\"tekst\"")
```

String jako ciąg bajtów

Stringi niemutowalne

```
b"byte"
bytes([34,56,50,40])
```

String jako ciąg bajtów

Stringi niemutowalne

```
b"byte"
bytes([34,56,50,40])
```

Mutowalne

```
bytearray(b"byte")
bytearray([34,56,50,40])
```

Plan wykładu

Przykłady kolekcji

- Listy: `[12,3]`
- Napisy: `'abcdef'`, `"Zażółć gęślą żółtą jaźń"`
- Krotki: `(1, 'jeden', (1, 1+1j, 0xFF))`
- Słowniki
- Zbiory

Krotki

```
brown = 165, 42, 42
NavyBlue = (0, 0, 128)
htmlColor = { 'turquoise' : (64,224,208),
              'NavyBlue' : NavyBlue }
r, g, b = htmlColor['NavyBlue']
```

Przypomnienie

podstawienie

$$a, b = 1, 2$$

Przypomnienie

podstawienie

$(a, b) = (1, 2)$

Przypomnienie

podstawienie

$(a, b) = (1, 2)$

Po co krotki

```
def gcd(a,b):
    while b !=0:
        a, b = b, a % b
    return a
```

Po co krotki (2)

```
def minmax(lista):
    min, max = float('+inf'), float('-inf')
    for n in lista:
        if n < min:
            min = n
        if n > max:
            max = n
    return (min, max)
```

Listy: streszczenie

```
male = [1, 2, 3]
duze = [100, 200, 300]
liczby = male + duze
figury = ['K', 'D', "W"] + list((range(2, 11))) + ["A"]
```

Podstawowe operacje na listach (i nie tylko)

```
len(['K', 'D', "W"] + list((range(2, 11))) + ["A"])
len("Python")
```

Podstawowe operacje na listach (i nie tylko)

```
len(['K', 'D', "W"] + list((range(2, 11))) + ["A"])
len("Python")
```

Długość krotki

```
len(1,2,3)
len((1, 2, 3))
```

Podstawowe operacje na listach (i nie tylko)

```
len(['K', 'D', "W"] + list((range(2, 11))) + ["A"])
len("Python")
```

Długość krotki

```
len(1,2,3)
len((1, 2, 3))
```

Zagadka: jak wygląda krotka długości jeden?

Podstawowe operacje: zawieranie

Operator in

```
'bc' in 'abcdefghijklmnoprstuwxyz'
4 in [2, 3, 5, 7, 11]
5 in (3, 8)
```

Podstawowe operacje: mnożenie

```
[0] * 16
(8, 8) * 8
```

Podstawowe operacje: przetwarzanie elementów

```
for n in [2, 3, 5, 7, 11, 13, 17]:
    print(n)
```

Podstawowe operacje: przetwarzanie elementów

```
for n in [2, 3, 5, 7, 11, 13, 17]:
    print(n)
for n in (2, 3, 5, 7, 11, 13, 17):
    print(n)
```

Podstawowe operacje: przetwarzanie elementów

```
for n in [2, 3, 5, 7, 11, 13, 17]:
    print(n)
for n in (2, 3, 5, 7, 11, 13, 17):
    print(n)
for n in "abcdefghijklmnoprstuwxyz":
    print(n)
```

Iterables

Iterables

To są te typy w Pythonie, na których można wykonać instrukcję for in.

Iterables

Iterables

To są te typy w Pythonie, na których można wykonać instrukcję for in.

krotki
stringi
listy
zbiory
generatory

Podstawowe operacje: fragment kolekcji

```
'informatyka'[2:5]
[1, 2, 3, 4, 5, 6, 7][2:5:3]
(1, 2, 3, 4, 5)[:-2]
```

Podstawowe operacje: fragment kolekcji

```
'informatyka'[2:5]
[1, 2, 3, 4, 5, 6, 7][2:5:3]
(1, 2, 3, 4, 5)[:-2]
```

Przypomnienie

```
[1, 2, 3, 4, 5, 6, 7][::-1]
```

Operacje mniej podstawowe

```
x = [1,2,3]
y = [4,5,6]
prod = 0
for i in range(len(x)):
    prod += x[i] * y[i]
```

Operacje mniej podstawowe

```
x = [1,2,3]
y = [4,5,6]
prod = 0
for i in range(len(x)):
    prod += x[i] * y[i]

x = [1,2,3]
y = [4,5,6]
prod = 0
for i, v in enumerate(x):
    prod += v * y[i]
```

Jeszcze inne rozwiązanie

```
x = [1,2,3]
y = [4,5,6]
prod = 0
for a, b in zip(x, y):
  prod += a * b
```

Plan wykładu

Garść informacji

Fakt 1.

Stringi i krotki są niemutowalne.

Garść informacji

Fakt 1.

Stringi i krotki są niemutowalne.

Fakt 2.

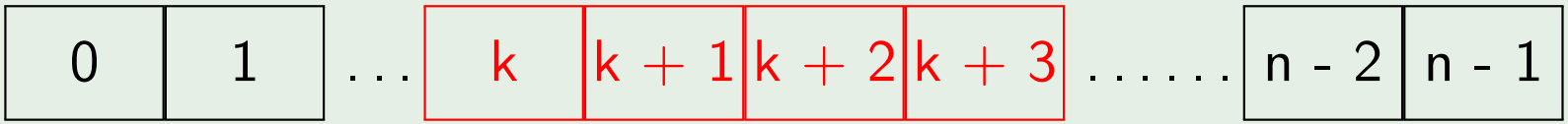
Listy są mutowalne.

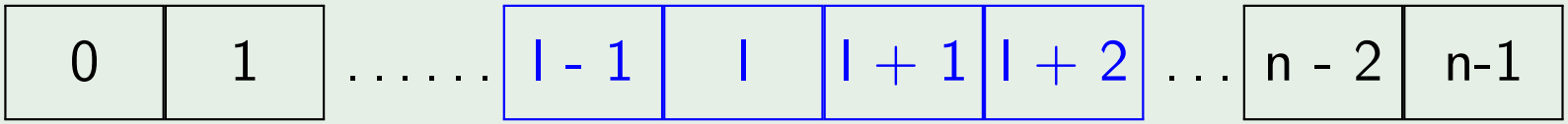
Wymiana elementów w liście

```
lista = [1,2,3]
lista[1] = 5
lista[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]
```

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]

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

```
lista = [ "żółty", "zielony", "czerwony", "niebieski" ]
del lista[3]
# wynik: ["żółty", "zielony", "czerwony"]

del lista[1:]
# wynik: ["żółty" ]
```

Różne operacje na listach

```
lista = [1,2,3]
lista.append(4)
print(lista.pop())
```

Różne operacje na listach

```
lista = [1,2,3]
lista.append(4)
print(lista.pop())
```

Inne operacje

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

Innde operacje, cd.

Przykłady

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

Innde operacje, cd.

Przykłady

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

Odwracanie listy: zwrócenie wyniku

```
lista = [0, 1, 2, 3]
reversed(lista)[a] # zwraca wynik
```

[a] a właściwie list(reversed(lista))

Rozpinanie list

```
lista = list(range(3))
a, b, c = lista
```

Rozpinanie list

```
lista = list(range(3))
a, b, c = lista
```

```
lista = list(range(10))
a, b, *c = lista
*a, b = lista
```

Plan wykładu

Implementacja i efektywność list

Implementacja: wektor wskaźników
Czas dostępu:

$$O(1)$$

Wstawianie/usuwanie elementów na końcu: zamortyzowany czas

$$O(1)$$

Wstawianie/usuwanie elementów, dowolne miejsce:

$$O(n)$$

Kolejki

Efektywne ($O(1)$) wstawianie i usuwanie elementów z końców

```
from collections import deque

d = deque('abcd')
d.append('e')
print(d.pop())
```

Tablice z elementami tego samego typu

```
from array import array

a = array('l', [1,2,3,4])
print(a[2])
a[2] = 5
```