Listy w Pythonie

Rozdział 8

Python dla wszystkich
www.py4e.pl

Programowanie

- Algorytm
  - Zestaw zasad lub kroków używanych do rozwiązania problemu
- Struktura danych
  - Ustalony sposób organizowania danych w komputerze

https://pl.wikipedia.org/wiki/Algorytm
https://pl.wikipedia.org/wiki/Struktura_danych

Co nie jest “kolekcją”?

Większość z naszych zmiennych zawiera jedną wartość – kiedy przypisujemy nową wartość do zmiennej, stara wartość jest nadpisywana

```
$ python3
>>> x = 2
>>> x = 4
>>> print(x)
4
```

Lista jest typem kolekcji

- Kolekcja pozwala nam przypisać wiele wartości do jednej "zmiennej"
- Kolekcje są fajne, bo możemy zapakować wiele wartości do jednej wygodnej "walizki"

```
friends = ['Józek', 'Gienek', 'Staszek']

carryon = [ 'skarpetki', 'koszula', 'perfumy' ]
```

Stałe będące listami

- Stałe będące listami zapisujemy w kwadratowych nawiasach, a ich elementy rozdzielamy przecinkami
- Elementem listy może być dowolny obiekt Pythona, nawet inna lista
- lista może być pusta

```
>>> print([1, 24, 76])
[1, 24, 76]
>>> print(['czerwony', 'żółty', 'niebieski'])
['czerwony', 'żółty', 'niebieski']
>>> print(['czerwony', 24, 98.6])
['czerwony', 24, 98.6]
>>> print([ 1, [5, 6], 7])
[1, [5, 6], 7]
>>> print([])
[]
```

Używamy już list!

```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Odpalamy!')
```

5
4
3
2
1
Odpalamy!

Listy i pętle określone – najlepsi kumple

```
friends = ['Józek', 'Gienek', 'Staszek']
for friend in friends:
    print('Szczęśliwego Nowego Roku:', friend)
print('Zrobione!')
```

Szczęśliwego Nowego Roku: Józek
Szczęśliwego Nowego Roku: Gienek
Szczęśliwego Nowego Roku: Staszek
Zrobione!

```
z = ['Józek', 'Gienek', 'Staszek']
for x in z:
    print('Szczęśliwego Nowego Roku:',  x)
print('Zrobione!')
```

Przyjrzyjmy się listom

Tak jak w przypadku ciągów znaków, możemy wybrać dowolny element z listy za pomocą indeksu określonego w nawiasach kwadratowych

Józek	Gienek	Staszek
0	1	2

```
>>> friends = ['Józek', 'Gienek', 'Staszek'
]
>>> print(friends[1])
Gienek
>>>
```

Listy są zmienne

- Ciągi znaków są “niezmienne” – nie możemy zmienić zawartości ciągu – musimy stworzyć nowy ciąg, żeby wprowadzić zmiany
- Listy są “zmienne” – możemy zmienić element listy, korzystając z operatora indeksu

```
>>> fruit = 'Banan'
>>> fruit[0] = 'b'
Traceback
TypeError: 'str' object does not
support item assignment
>>> x = fruit.lower()
>>> print(x)
banan
>>> lotto = [2, 14, 26, 41, 63]
>>> print(lotto)
[2, 14, 26, 41, 63]
>>> lotto[2] = 28
>>> print(lotto)
[2, 14, 28, 41, 63]
```

Jaka jest długość listy?

- Funkcja len() przyjmuje listę jako parametr i zwraca liczbę elementów na liście
- Tak naprawdę len() poda nam liczbę elementów dowolnej sekwencji (takiej jak ciąg znaków...)

```
>>> greet = 'Witaj Bob'
>>> print(len(greet))
9
>>> x = [ 1, 2, 'joe', 99]
>>> print(len(x))
4
>>>
```

Używanie funkcji range

- Funkcja range zwraca listę liczb od zera do wartości mniejszej o jeden od parametru
- Możemy stworzyć indeksowaną pętlę for z całkowitym iteratorem

```
>>> print(range(4))
[0, 1, 2, 3]
>>> friends = ['Józek', 'Gienek', 'Staszek']
>>> print(len(friends))
3
>>> print(range(len(friends)))
[0, 1, 2]
>>>
```

O dwóch takich... pętlach

```
friends = ['Józek', 'Gienek', 'Staszek']

for friend in friends :
    print('Szczęśliwego Nowego Roku:', friend)

for i in range(len(friends)) :
    friend = friends[i]
    print('Szczęśliwego Nowego Roku:', friend)
```

```
>>> friends = ['Józek', 'Gienek', 'Staszek' ]
>>> print(len(friends))
3
>>> print(range(len(friends)))
[0, 1, 2]
>>>
```

Szczęśliwego Nowego Roku: Józek
Szczęśliwego Nowego Roku: Gienek
Szczęśliwego Nowego Roku: Staszek

Konkatenacja list przy użyciu +

Możemy stworzyć nową listę, łącząc ze sobą dwie istniejące

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print(c)
[1, 2, 3, 4, 5, 6]
>>> print(a)
[1, 2, 3]
```

Twórz wycinki list, używając :

```
>>> t = [9, 41, 12, 3, 74, 15]
>>> t[1:3]
[41,12]
>>> t[:4]
[9, 41, 12, 3]
>>> t[3:]
[3, 74, 15]
>>> t[:]
[9, 41, 12, 3, 74, 15]
```

Pamiętaj: Tak jak w ciągach "do, ale nie razem z"

Metody list

```
>>> x = list()
>>> type(x)
<type 'list'>
>>> dir(x)
['append', 'count', 'extend', 'index', 'insert',
'pop', 'remove', 'reverse', 'sort']
>>>
```

https://docs.python.org/3/tutorial/datastructures.html

Tworzenie listy od podstaw

- Możemy stworzyć pustą listę, a później dodać do niej elementy, używając metody append
- List zachowuje kolejność elementów, a nowe są dodawane na końcu

```
>>> stuff = list()
>>> stuff.append('książka')
>>> stuff.append(99)
>>> print(stuff)
['książka', 99]
>>> stuff.append('ciastko')
>>> print(stuff)
['książka', 99, 'ciastko']
```

Jesteście na liście?

- Python daje dwa operatory, które pozwalają sprawdzić, czy coś jest na liście
- To operatory logiczne zwracające wartości True albo False
- Nie zmieniają listy

```
>>> some = [1, 9, 21, 10, 16]
>>> 9 in some
True
>>> 15 in some
False
>>> 20 not in some
True
>>>
```

Listy są w porządku

- Lista może przechowywać wiele elementów i utrzymuje je w tej samej kolejności, dopóki specjalnie jej nie zmienimy
- Listę można posortować (czyli zmienić kolejność elementów)
- Metoda sort() (inaczej niż w przypadku ciągów) oznacza “posortuj siebie”

```
>>> friends = ['Józek', 'Gienek', 'Staszek' ]
>>> friends.sort()
>>> print(friends)
['Gienek', 'Józek', 'Staszek']
>>> print(friends[1])
Józek
>>>
```

Wbudowane funkcje i listy

- Wiele funkcji wbudowanych w Pythona przyjmuje listy jako parametry.
- Pamiętasz pętle, które tworzyliśmy? Te są o wiele prostsze.

```
>>> nums = [3, 41, 12, 9, 74, 15]
>>> print(len(nums))
6
>>> print(max(nums))
74
>>> print(min(nums))
3
>>> print(sum(nums))
154
>>> print(sum(nums)/len(nums))
25.6
```

```
total = 0
count = 0
while True :
    inp = input('Wprowadź liczbę: ')
    if inp == 'skończone' : break
    value = float(inp)
    total = total + value
    count = count + 1

average = total / count
print('Średnia:', average)
```

Wprowadź liczbę: 3
Wprowadź liczbę: 9
Wprowadź liczbę: 5
Wprowadź liczbę: skończone
Średnia: 5.66666666667

```
numlist = list()
while True :
    inp = input('Wprowadź liczbę: ')
    if inp == 'skończone' : break
    value = float(inp)
    numlist.append(value)

average = sum(numlist) / len(numlist)
print('Średnia:', average)
```

Najlepsi przyjaciele: ciągi i listy

```
>>> abc = 'W trzech słowach'
>>> stuff = abc.split()
>>> print(stuff)
['W', 'trzech', 'słowach']
>>> print(len(stuff))
3
>>> print(stuff[0])
W
```

```
>>> print(stuff)
['W', 'trzech', 'słowach']
>>> for w in stuff :
...     print(w)
...
W
trzech
słowach
>>>
```

Metoda split() dzieli ciąg znaków na części i tworzy listę ciągów. Myślimy o nich jak o słowach. Możemy uzyskać dostęp do poszczególnych słów, przechodząc przez wszystkie pętlą.

```
>>> line = 'Bardzo dużo               znaków spacji'
>>> etc = line.split()
>>> print(etc)
['Bardzo', 'dużo', 'znaków', 'spacji']
>>>
>>> line = 'pierwsza;druga;trzecia'
>>> thing = line.split()
>>> print(thing)
['pierwsza;druga;trzecia']
>>> print(len(thing))
1
>>> thing = line.split(';')
>>> print(thing)
['pierwsza', 'druga', 'trzecia']
>>> print(len(thing))
3
>>>
```

- Jeśli nie określisz separatora, wielokrotne spacje są traktowane jak jeden separator
- Możesz określić, jaki znak separatora ma być użyty do dzielenia

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    if not line.startswith('From ') : continue
    words = line.split()
    print(words[2])
```

Sat
Fri
Fri
Fri
...

```
>>> line = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
>>> words = line.split()
>>> print(words)
['From', 'stephen.marquard@uct.ac.za', 'Sat', 'Jan', '5', '09:14:16', '2008']
>>>
```

Wzorzec podwójnego dzielenia

Czasem dzielimy linię na jeden sposób i jedną z uzyskanych części dzielimy jeszcze raz

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008

words = line.split()
email = words[1]
print pieces[1]
```

Wzorzec podwójnego dzielenia

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008

words = line.split()
email = words[1]                    stephen.marquard@uct.ac.za
print pieces[1]
```

Wzorzec podwójnego dzielenia

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008

words = line.split()
email = words[1]                    stephen.marquard@uct.ac.za
pieces = email.split('@')           ['stephen.marquard', 'uct.ac.za']
print pieces[1]
```

Wzorzec podwójnego dzielenia

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008

words = line.split()
email = words[1]                    stephen.marquard@uct.ac.za
pieces = email.split('@')           ['stephen.marquard', 'uct.ac.za']
print(pieces[1])                    'uct.ac.za'
```

Podsumowanie

- Idea kolekcji
- Listy i pętle określone
- Indeksowanie i wyszukiwanie
- Zmienność list
- Funkcje: len, min max, sum
- Wycinki list
- Metody list: append, remove
- Sortowanie list
- Dzielenie ciągów znaków na listy słów
- Używanie split do parsowania ciągów

Podziękowania dla współpracowników

Autorstwo pierwszej wersji: Charles Severance, University of Michigan School of Information

Polska wersja powstała z inicjatywy Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu

Tłumaczenie: Agata i Krzysztof Wierzbiccy, EnglishT.eu

... wstaw tu nowych współpracowników i tłumaczy

...