Списки у Python

Розділ 8

Python для всіх
www.py4e.com

Програмування

- Алгоритм

  - Набір правил або кроків, які використовуються для розв’язку задачі

- Структури даних

  - Спосіб організації даних у комп'ютері

https://en.wikipedia.org/wiki/Algorithm
https://en.wikipedia.org/wiki/Data_structure

Що не належить до «колекцій»?

Більшість наших змінних мають одне значення – коли ми вводимо нове значення у змінну, старе значення перезаписується

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

Всі списки – це колекції

- Колекція дозволяє нам помістити багато значень в одну «змінну».
- Колекція хороша тим, що ми можемо запакувати багато значень в один зручний пакунок.

```
friends = [ 'Joseph', 'Glenn', 'Sally' ]

carryon = [ 'socks', 'shirt', 'perfume' ]
```

Константи у списках

- Константи у списку беруться в квадратні дужки, а елементи списку відокремлюються комами
- Елементом списку може бути будь-який об'єкт Python – навіть інший список.
- Список може бути порожнім

```
>>> print([1, 24, 76])
[1, 24, 76]
>>> print(['red', 'yellow',
'blue'])
['red', 'yellow', 'blue']
>>> print(['red', 24, 98.6])
['red', 24, 98.6]
>>> print([ 1, [5, 6], 7])
[1, [5, 6], 7]
>>> print([])
[]
```

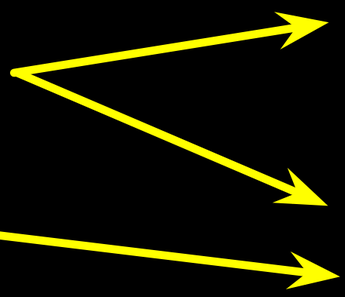
Ми вже використовували списки!

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

```
5
4
3
2
1
Blastoff!
```

Списки та визначені цикли - найкращі друзі

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends :
    print('Happy New Year:',  friend)
print('Done!')
```



```
З Новим роком: Джозеф
З Новим роком: Гленн
З Новим роком: Саллі
Готово!
```

```
z = ['Joseph', 'Glenn', 'Sally']
for x in z:
    print('Happy New Year:',  x)
print('Done!')
```

Всередині списків

Як і у випадку з рядками, ми можемо дістатися до будь-якого елемента списку за допомогою індексу, вказаного у квадратних дужках

Joseph	Glenn	Sally
0	1	2

```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]
>>> print(friends[1])
Glenn
>>>
```

Списки можна змінювати

- Рядки «незмінні» – ми не можемо змінити вміст рядка – ми повинні створити новий рядок, щоб внести будь-які зміни

- Списки «змінювані» – ми можемо змінити елемент списку за допомогою оператора індексу

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

Яка довжина списку?

- Функція len() отримує список як параметр і повертає кількість елементів у списку

- Фактично len() повертає нам кількість елементів будь-якої множини або послідовності (наприклад, рядка...)

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

Використання функції range

- Функція range повертає список чисел, які знаходяться в діапазоні від нуля до значення параметру мінус один

- Ми можемо побудувати індексований цикл з використанням for та цілочисельного ітератора

```
>>> print(range(4))
[0, 1, 2, 3]
>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print(len(friends))
3
>>> print(list(range(len(friends))))
[0, 1, 2]
>>>
```

Казка про два цикли...

```
friends = ['Joseph', 'Glenn', 'Sally']

for friend in friends :
    print('Happy New Year:',  friend)

for i in range(len(friends)) :
    friend = friends[i]
    print('Happy New Year:',  friend)
```

```
>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print(len(friends))
3
>>> print(list(range(len(friends))))
[0, 1, 2]
>>>
```

Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally

Конкатенація списків з використанням +

Ми можемо створити новий список шляхом об'єднання двох наявних списків

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

Можна робити зрізи списків з використанням :

```
>>> 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]
```

Пам'ятай: Так само, як і в рядках, друге число – «до, але не включаючи»

Методи списків

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

http://docs.python.org/tutorial/datastructures.html

Створення списку

- Ми можемо створити порожній список, а потім додати елементи за допомогою методу append
- Список залишається впорядкованим, а нові елементи додаються в кінець списку

```
>>> stuff = list()
>>> stuff.append('book')
>>> stuff.append(99)
>>> print(stuff)
['book', 99]
>>> stuff.append('cookie')
>>> print(stuff)
['book', 99, 'cookie']
```

Чи є щось у списку?

- У мові Python є два оператори, які дозволяють перевірити, чи є елемент у списку
- Це логічні оператори, які повертають значення True або False
- Вони не змінюють список

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

Списки впорядковані

- Список може містити багато елементів і зберігає їх у такому порядку, доки ми не змінимо порядок
- Список можна сортувати (тобто змінювати його порядок)
- Метод sort (на відміну від рядків) означає «відсортуй себе»

```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]
>>> friends.sort()
>>> print(friends)
['Glenn', 'Joseph', 'Sally']
>>> print(friends[1])
Joseph
>>>
```

Вбудовані функції та списки

- У Python є ряд вбудованих функцій, які приймають списки як параметри

- Пам'ятаєте цикли, які ми створювали? Можна зробити набагато простіше

```
>>> 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('Enter a number: ')
    if inp == 'done' : break
    value = float(inp)
    total = total + value
    count = count + 1

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

```
Enter a number: 3
Enter a number: 9
Enter a number: 5
Enter a number: done
Average: 5.66666666667
```

```
numlist = list()
while True :
    inp = input('Enter a number: ')
    if inp == 'done' : break
    value = float(inp)
    numlist.append(value)

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

Найкращі друзі: рядки та списки

```
>>> abc = 'With three words'
>>> stuff = abc.split()
>>> print(stuff)
['With', 'three', 'words']
>>> print(len(stuff))
3
>>> print(stuff[0])
With
```

```
>>> print(stuff)
['With', 'three', 'words']
>>> for w in stuff :
...     print(w)
...
With
Three
Words
>>>
```

split розділяє рядок на частини і створює список рядків. Ми розглядаємо їх як слова. Ми можемо отримати доступ до конкретного слова або перебрати всі слова

```
>>> line = 'A lot                    of spaces'
>>> etc = line.split()
>>> print(etc)
['A', 'lot', 'of', 'spaces']
>>>
>>> line = 'first;second;third'
>>> thing = line.split()
>>> print(thing)
['first;second;third']
>>> print(len(thing))
1
>>> thing = line.split(';')
>>> print(thing)
['first', 'second', 'third']
>>> print(len(thing))
3
>>>
```

- Якщо ви не вказуєте роздільник, кілька пробілів розглядаються як один роздільник
- Ви можете вказати, який роздільник використовувати при розбитті

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']
>>>
```

Подвійне розділення

Іноді ми розділяємо рядок, а потім беремо один з фрагментів і розділяємо його знову

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

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

Подвійне розділення

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

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

stephen.marquard@uct.ac.za

Подвійне розділення

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

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

```
stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
```

Подвійне розділення

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

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

```
stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
'uct.ac.za'
```

Підсумки

- Поняття колекцій
- Списки та визначені цикли
- Індексування та пошук
- Зміни у списках
- Функції: len, min, max, sum
- Зріз у списках
- Методи списків: append, remove
- Сортування списків
- Розділення рядків на списки слів
- Використання функції split для парсингу рядків

Права власності / Застереження

...