

СЛОВНИКИ у Python

Розділ 9

Python для всіх
www.py4e.com



Що таке колекція?

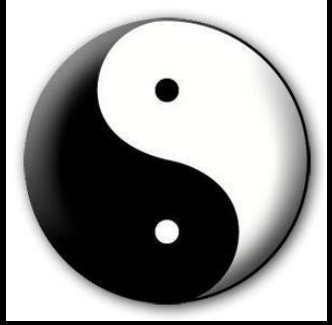


- Колекція хороша тим, що ми можемо помістити в неї більше одного значення та складати їх, наче у валізи.
- У нас є купа значень в одній «змінній».
- Це можливо завдяки наявності більше одного місця «всередині» («in») змінної.
- У нас є способи знайти різні місця у змінній.

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
```



Казка про дві колекції..

- Список

Лінійна колекція значень
Шукає за індексом від 0 до (довжина-1)



- Словник

Лінійна колекція пар «ключ-значення»
Шукає за «мітками» або «ключами»



https://en.wikipedia.org/wiki/Index_card#/media/File:LA2-katalogkort.jpg

<https://commons.wikimedia.org/wiki/File:Shelves-of-file-folders.jpg>

СЛОВНИКИ

- Словники – найпотужніші колекції даних у Python
- Словники дозволяють нам виконувати швидкі операції, подібні до баз даних, у Python
- Схожі концепції в різних мовах програмування
 - Асоціативні масиви - Perl / PHP
 - Властивості або карти або HashMap - Java
 - Контейнер властивостей - C# / .Net



Розвиток словників у Python

- До версії Python 3.7 словники не зберігали елементи в порядку додавання
- Словники Python 3.7 (2018) і пізніших версій зберігають елементи в порядку додавання
- «Порядок додавання» не завжди є «відсортованим порядком»

На нижчому рівні абстрагування

- Списки, словники та кортежі в Python – це «абстрактні об'єкти», створені для того, щоб бути простими у використанні
- Наразі ми просто розберемося з ними, будемо їх використовувати і подякуємо творцям Python за те, що вони зробили їх простими для нас
- Використовувати колекції в Python легко. Створення коду для їхньої підтримки є складним і використовує концепції комп'ютерних наук, такі як динамічна пам'ять, масиви, зв'язані списки, хеш-карти та трімепи.
- Але ці деталі реалізації будуть розглянуті в іншому курсі...

Списки (огляд)

- Значення додаються в кінець списку і шукаємо їх за позицією (індексом).
- Значення вставляються в **СЛОВНИК** за допомогою ключа, отримати їх також можна за допомогою ключа

```
>>> cards = list()
>>> cards.append(12)
>>> cards.append(3)
>>> cards.append(75)
>>> print(cards)
[12, 3, 75]
>>> print(cards[1])
3
>>> cards[1] = cards[1] + 2
>>> print(cards)
[12, 5, 75]
```



СЛОВНИКИ



- Значення додаються в кінець списку і шукаємо їх за позицією (індексом).
- Значення вставляються в **СЛОВНИК** за допомогою ключа, отримати їх також можна за допомогою ключа

```
>>> cabinet = dict()
>>> cabinet['summer'] = 12
>>> cabinet['fall'] = 3
>>> cabinet['spring'] = 75
>>> print(cabinet)
{'summer': 12, 'fall': 3, 'spring': 75}
>>> print(cabinet['fall'])
3
>>> cabinet['fall'] = cabinet['fall'] + 2
>>> print(cabinet)
{'summer': 12, 'fall': 5, 'spring': 75}
```

Порівняння списків і словників

Словники схожі на списки, за винятком того, що вони використовують **ключі** замість індексів для пошуку **значень**

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> ddd = dict()
>>> ddd['age'] = 21
>>> ddd['course'] = 182
>>> print(ddd)
{'age': 21, 'course': 182}
>>> ddd['age'] = 23
>>> print(ddd)
{'age': 23, 'course': 182}
```

Літерали словника (константи)

- Літерали словника використовують фігурні дужки і мають пари **ключ-значення**
- **Порожній словник** можна зробити за допомогою порожніх фігурних дужок

```
>>> jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100 }
>>> print(jjj)
{'chuck': 1, 'fred': 42, 'jan': 100}
>>> ooo = { }
>>> print(ooo)
{}
>>>
```

Найпоширеніші імена?

Найпоширеніші імена?

marquard	cwen	cwen
zhen	marquard	zhen
csev	zhen	csev
zhen	csev	marquard
		zhen

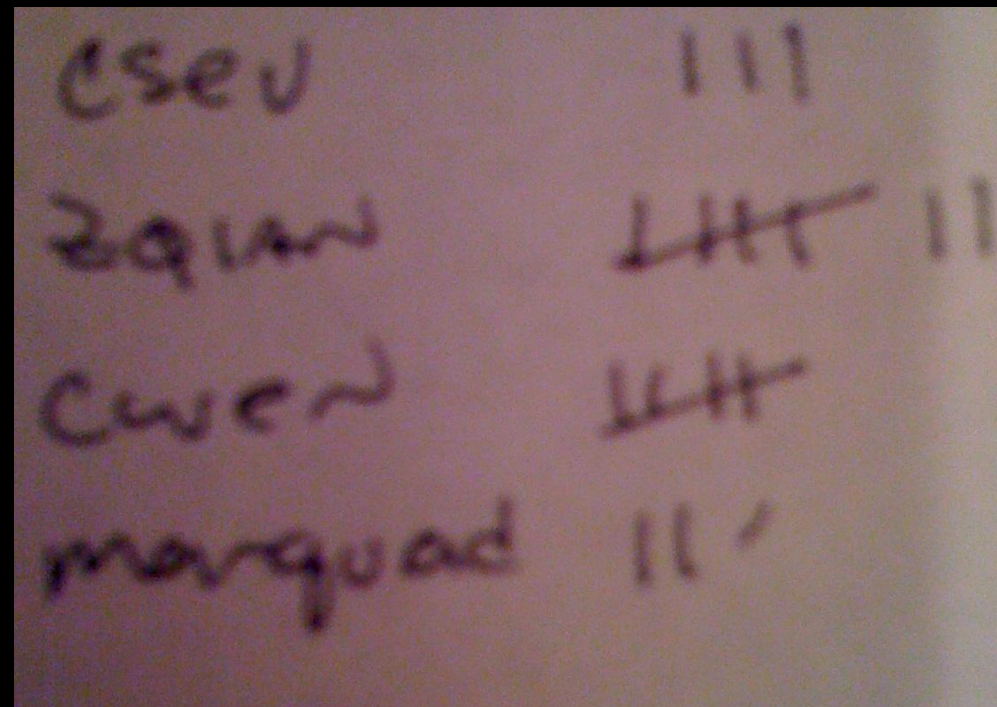
Найпоширеніші імена?

marquard

cwen

cwen

zhen



zhen

csev

csev

zhen

csev

marquard

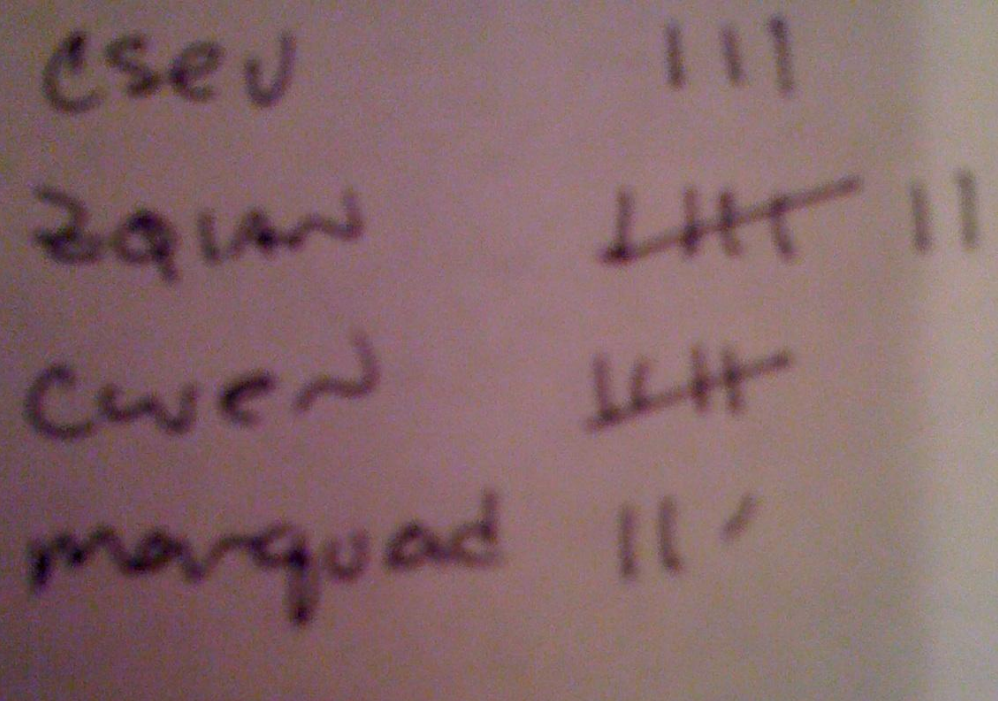
zhen

Багато лічильників зі словником

Одне з найпоширеніших застосувань словників – це підрахунок того, як часто ми щось «бачимо»

```
>>> ccc = dict()
>>> ccc['csev'] = 1
>>> ccc['cwen'] = 1
>>> print(ccc)
{'csev': 1, 'cwen': 1}
>>> ccc['cwen'] = ccc['cwen'] + 1
>>> print(ccc)
{'csev': 1, 'cwen': 2}
```

Ключ Значення



A photograph of a piece of paper with handwritten text. The text is organized into two columns. The left column contains the words 'csev', 'zqian', 'cwen', and 'marquod'. The right column contains the counts '111', '1111', '111', and '111' respectively. The handwriting is in blue ink on a light-colored background.

csev	111
zqian	1111
cwen	111
marquod	111

Помилки у словниках

- **Помилкою** є посилання на ключ, якого немає у словнику
- Можна використати оператор **in**, щоб перевірити, чи є ключ у словнику

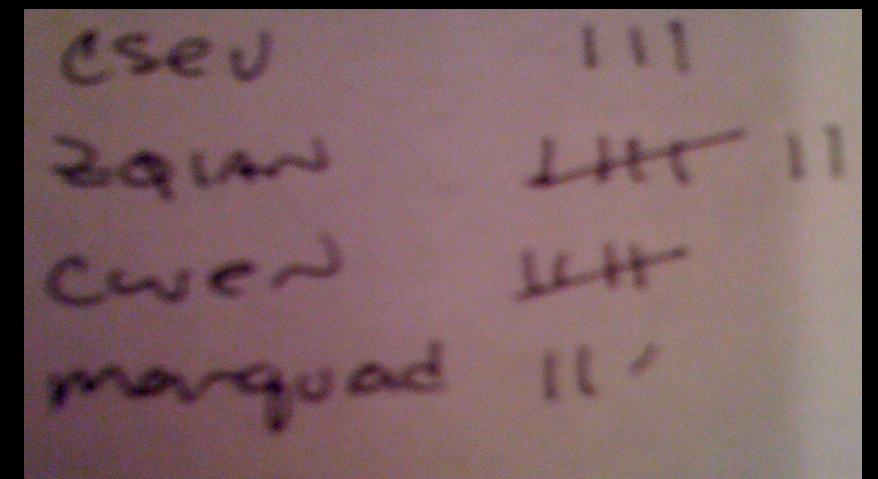
```
>>> ccc = dict()
>>> print(ccc['csev'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'csev'
>>> 'csev' in ccc
False
```

Коли з'являється нове ім'я

Коли ми зустрічаємо нове ім'я, нам потрібно додати новий запис до **словника**, і якщо ми бачимо це **ім'я** вдруге або пізніше, ми просто додаємо одиницю до підрахунку в словнику під цим **ім'ям**

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names:
    if name not in counts:
        counts[name] = 1
    else:
        counts[name] = counts[name] + 1
print(counts)
```

`{'csev': 2, 'cwen': 2, 'zqian': 1}`



Метод `get` для словників

Шаблон перевірки наявності
ключа в словнику і прийняття
значення за замовчуванням, якщо
ключа там немає, настільки
поширений, що існує метод `get()`,
який робить це за нас.

Значення за замовчуванням,
якщо ключ не існує (і немає
помилки).

```
if name in counts:  
    x = counts[name]  
else :  
    x = 0
```

```
x = counts.get(name, 0)
```

```
{'csev': 2, 'cwen': 2, 'zqian': 1}
```

Спрощений підрахунок за допомогою `get()`

Ми можемо використовувати `get()` і надати значення **за замовчуванням нуль**, коли **ключа** ще немає в словнику, а потім просто додавати одиницю

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
    counts[name] = counts.get(name, 0) + 1
print(counts)
```

За замовчуванням

`{'csev': 2, 'cwen': 2, 'zqian': 1}`

Підрахунок слів у тексті

Writing programs (or programming) is a very creative and rewarding activity. You can write programs for many reasons ranging from making your living to solving a difficult data analysis problem to having fun to helping someone else solve a problem. This book assumes that everyone needs to know how to program and that once you know how to program, you will figure out what you want to do with your newfound skills.

We are surrounded in our daily lives with computers ranging from laptops to cell phones. We can think of these computers as our “personal assistants” who can take care of many things on our behalf. The hardware in our current-day computers is essentially built to continuously ask us the question, “What would you like me to do next?”

Our computers are fast and have vast amounts of memory and could be very helpful to us if we only knew the language to speak to explain to the computer what we would like it to do next. If we knew this language we could tell the computer to do tasks on our behalf that were repetitive. Interestingly, the kinds of things computers can do best are often the kinds of things that we humans find boring and mind-numbing.

Шаблон підрахунку

```
counts = dict()
print('Enter a line of text:')
line = input('')

words = line.split()

print('Words:', words)

print('Counting...')
for word in words:
    counts[word] = counts.get(word, 0) + 1
print('Counts', counts)
```

Загальний шаблон підрахунку слів у рядку тексту полягає в тому, що ви розбиваєте рядок на слова, потім циклічно переглядаєте їх і за допомогою словника відстежуєте підрахунок кожного слова незалежно від інших.


```
python wordcount.py
```

Введіть рядок тексту:

```
the clown ran after the car and the car ran into the tent  
and the tent fell down on the clown and the car
```

Слова: ['the', 'clown', 'ran', 'after', 'the', 'car',
'and', 'the', 'car', 'ran', 'into', 'the', 'tent', 'and',
'the', 'tent', 'fell', 'down', 'on', 'the', 'clown',
'and', 'the', 'car']

Підрахунок...

Кількість {'the': 7, 'clown': 2, 'ran': 2, 'after': 1,
'car': 3, 'and': 3, 'into': 1, 'tent': 2, 'fell': 1,
'down': 1, 'on': 1}

```
counts = dict()
line = input('Enter a line of text:')
words = line.split()

print('Words:', words)
print('Counting...')

for word in words:
    counts[word] = counts.get(word, 0) + 1
print('Counts', counts)
```

python wordcount.py

Введіть рядок тексту:

the clown ran after the car and the car ran
into the tent and the tent fell down on the
clown and the car

Слова: ['the', 'clown', 'ran', 'after', 'the', 'car',
'and', 'the', 'car', 'ran', 'into', 'the', 'tent', 'and',
'the', 'tent', 'fell', 'down', 'on', 'the', 'clown',
'and', 'the', 'car']

Підрахунок...

Кількість {'the': 7, 'clown': 2, 'ran': 2, 'after':
1, 'car': 3, 'and': 3, 'into': 1, 'tent': 2, 'fell': 1,
'down': 1, 'on': 1}

Визначені цикли та словники

Ми можемо написати цикл **for**, який перебирає всі записи в словнику – насправді він перебирає всі **ключі** в словнику і шукає **значення**

```
>>> counts = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> for key in counts:
...     print(key, counts[key])
...
chuck 1
fred 42
jan 100
>>>
```

Отримання списків ключів і значень

Ви можете
отримати список
ключів, значень
або елементів
(і того, і іншого) зі
словника

```
>>> jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}  
>>> print(list(jjj))  
['chuck', 'fred', 'jan']  
>>> print(list(jjj.keys()))  
['chuck', 'fred', 'jan']  
>>> print(list(jjj.values()))  
[1, 42, 100]  
>>> print(list(jjj.items()))  
[('chuck', 1), ('fred', 42), ('jan', 100)]  
>>>
```



Що таке «кортеж»? Поговоримо пізніше...

Бонус: дві ітераційні змінні!

- Ми перебираємо пари **ключ-значення** у словнику, використовуючи ***дві*** ітераційні змінні
- На кожній ітерації перша змінна — це **ключ**, а друга змінна — відповідне **значення** для ключа

```
jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}  
for aaa,bbb in jjj.items() :  
    print(aaa, bbb)
```

```
chuck 1  
fred 42  
jan 100
```

aaa	bbb
[chuck]	1
[fred]	42
[jan]	100

```
name = input('Enter file:')
handle = open(name)

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

python words.py
Введіть файл: words.txt
to 16

python words.py
Введіть файл : clown.txt
the 7

Використання двох вкладених циклів

Підсумки

- Що таке колекції?
- Списки проти словників
- Константи словників
- Найпоширеніше слово
- Використання методу `get()`
- Написання циклів у словниках
- Трохи наперед: Кортежі

Права власності / Застереження



Авторські права на ці слайди з 2010 року належать Чарльзу Северенсу (www.dr-chuck.com) зі Школи інформації Мічиганського університету та захищені ліцензією Creative Commons Attribution 4.0. Будь ласка, збережіть цей фінальний слайд у всіх копіях документа, щоб відповідати вимогам ліцензії щодо посилань на джерела. При повторній публікації матеріалів, якщо щось зміните, додайте ім'я та організацію до переліку співавторів нижче.
Першоджерело: Чарльз Северенс, Школа інформації Мічиганського університету
Переклад: платформа Prometheus