

Phát triển phần mềm nâng cao cho tính toán khoa học Cấu trúc dữ liệu cơ bản

Vũ Tiến Dũng

Khoa Toán - Cơ - Tin học

Trường ĐH Khoa học Tự Nhiên Hà Nội

Nội dung

- 1 Xâu ký tự - String
- 2 Danh sách - List
- 3 Bộ - Tuple
- 4 Tập hợp - Set
- 5 Từ điển - Dictionary

Xâu ký tự - String

- Kiểu dữ liệu xâu ký tự (string) trong Python là một mảng của các byte biểu diễn các ký tự unicode.
- Tuy nhiên, Python không có kiểu dữ liệu riêng để biểu diễn ký tự, một ký tự trong xâu đơn giản là một xâu độ dài một.
- Một xâu là một mảng các ký tự, các phần tử trong mảng có thể được tham chiếu qua chỉ số đặt trong ngoặc vuông [].

```
>>> a = 'Hello world'
>>> print(a)
Hello world
>>> type(a)
<class 'str'>
>>> print(a[1])
e
>>> type(a[1])
<class 'str'>
>>>
```

Xâu ký tự - String

- Các phần tử trong xâu có thể được truy cập theo lát (Slice).

```
1      b = "Hello, World!"  
2      print(b[2:5])
```

- Chú ý chỉ số các phần tử trong xâu, hay trong mảng nói chung được tính từ 0, trong ví dụ trên phần tử có chỉ số 2 là **l**, phần tử có chỉ số 5 là dấu **,** kết quả là **llo**. Chú ý kết quả chỉ lấy đến trước phần tử có chỉ số 5.

Xâu ký tự - String

Hình 1: Tham chiếu các phần tử trong mảng (danh sách - list) theo lát (slice)

```
>>> b = "Hello, World"
>>> b[2:5]
'llo'
>>> b[2:-1]
'llo, Worl'
>>> b[-1:2]
''
>>> b[2:-1:2]
'lo ol'
>>> b[:2]
'Hl'
>>> b[5:]
'Hello'
>>> b[5:]
', World'
```

Một số phương thức thường dùng với xâu ký tự

- Lấy độ dài xâu `len()`

```
1 a = "Hello, World!"  
2 print(len(a))
```

- Xóa khoảng trắng thừa `strip()`

```
1 a = " Hello, World! "  
2 print(a.strip()) # returns "Hello, World!"
```

- Chuyển ký tự về ký tự hoa / thường `lower()` / `upper()`

```
1 a = "Hello, World!"  
2 print(a.lower())  
3 print(a.upper())
```

Một số phương thức thường dùng với xâu ký tự

- Thay thế xâu `replace()`

```
1 a = "Hello, World!"  
2 print(a.replace("H", "J"))
```

- Tách xâu `split()`

```
1 a = "Hello, World!"  
2 print(a.split(",")) # returns ['Hello', ' World!']
```

- Kiểm tra xâu có chứa / Không chứa xâu con `in` / `not in`

```
1 txt = "The rain in Spain stays mainly in the plain"  
2 x = "ain" in txt  
3 print(x)  
4 txt = "The rain in Spain stays mainly in the plain"  
5 x = "ain" not in txt  
6 print(x)
```

Một số phương thức thường dùng với xâu ký tự

- Cộng xâu / nối xâu

```
1   a = "Hello"  
2   b = "World"  
3   c = a + b  
4   print(c)
```

- Chú ý việc cộng xâu chỉ thực hiện giữa 2 xâu ký tự, nếu muốn thực hiện cộng xâu với số, ta phải chuyển số sang dạng xâu ký tự, tương tự với các kiểu dữ liệu khác

```
1   year = 2019  
2   print("Happy new year "+str(year))
```


Một số phương thức thường dùng với xâu ký tự

- Định dạng xâu `format()`

```
1      age = 36
2      txt = "My name is John, and I am {}"
3      print(txt.format(age))
4      quantity = 3
5      itemno = 567
6      price = 49.95
7      myorder = "I want {} pieces of item {} for {}
dollars."
8      print(myorder.format(quantity, itemno, price))
9
10     quantity = 3
11     itemno = 567
12     price = 49.95
13     myorder = "I want to pay {2} dollars for {0} pieces
of item {1}."
14     print(myorder.format(quantity, itemno, price))
```

Một số phương thức thường dùng với xâu ký tự

- Các phương thức thường dùng cho xâu ký tự - **string**

1

```
capitalize(); casefold(); center(); count();  
encode(); endswith(); expandtabs(); find(); format();  
format_map(); index(); isalnum(); isalpha();  
isdecimal(); isdigit(); isidentifier(); islower();  
isnumeric(); isprintable(); isspace(); istitle();  
isupper(); join(); ljust(); lower(); lstrip();  
maketrans(); partition(); replace(); rfind(); rindex();  
rjust(); rpartition(); rsplit();rstrip(); split();  
splitlines(); startswith(); strip(); swapcase();  
title(); translate(); upper(); zfill();
```

Danh sách - List

- Một danh sách là một bộ có tính thứ tự và có thể thay đổi được. Trong Python một danh sách được đặt trong một cặp ngoặc vuông []

```
1      thislist = ["apple", "banana", "cherry"]  
2      print(thislist)
```

- Các phần tử trong danh sách được truy cập đến thông qua chỉ số của phần tử đó. Chỉ số được đánh bắt đầu từ 0, và xoay vòng (phần tử cuối cùng tương ứng với chỉ số -1)

```
1      thislist = ["apple", "banana", "cherry"]  
2      print(thislist[1])  
3      print(thislist[-1])
```

Danh sách - List

- Truy cập đến một vùng các phần tử theo chỉ số

```
1      thislist = ["apple", "banana", "cherry", "orange",  
2                  "kiwi", "melon", "mango"]  
3      print(thislist[2:5])#['cherry', 'orange', 'kiwi']  
4      print(thislist[-4:-1]) #['orange', 'kiwi', 'melon']
```

- Thay đổi giá trị của phần tử trong danh sách

```
1      thislist = ["apple", "banana", "cherry"]  
2      thislist[1] = "blackcurrant"  
3      print(thislist)
```

- Duyệt qua các phần tử trong danh sách với vòng lặp **for**

```
1      thislist = ["apple", "banana", "cherry"]  
2      for x in thislist:  
3          print(x)
```

Danh sách - List

- Phần tử được thêm vào cuối danh sách bằng lệnh **append()**

```
1      thislist = ["apple", "banana", "cherry"]
2      thislist.append("orange")
3      print(thislist)
```

- Để thêm phần tử vào vị trí cụ thể dùng lệnh **insert()**

```
1      thislist = ["apple", "banana", "cherry"]
2      thislist.insert(1, "orange")
3      print(thislist)
```

- Xóa phần tử khỏi danh sách **remove()**

```
1      thislist = ["apple", "banana", "cherry"]
2      thislist.remove("banana")
3      print(thislist)
```

Danh sách - List

- Lấy phần tử và xóa khỏi danh sách **pop()**
- Mặc định thì lệnh **pop()** xóa phần tử ở cuối danh sách nếu không có tham số về chỉ số phần tử muốn xóa.

```
1      thislist = ["apple", "banana", "cherry"]
2      thislist.pop()
3      print(thislist)
```

- lệnh **pop()** cũng được sử dụng để lấy và xóa phần tử ở một vị trí cụ thể.

```
1      thislist = ["apple", "banana", "cherry"]
2      thislist.pop(1)
3      print(thislist)
```

- Xóa phần tử trong danh sách **del**

```
1      thislist = ["apple", "banana", "cherry"]
2      del thislist[0]
3      print(thislist)
```

Danh sách - List

- Làm trống danh sách `clear()`

```
1      thislist = ["apple", "banana", "cherry"]
2      thislist.clear()
3      print(thislist)
```

- Sao chép danh sách `copy()`

```
1      thislist = ["apple", "banana", "cherry"]
2      mylist = thislist.copy()
3      print(mylist)
```

Danh sách - List

Hình 2: Ví dụ về copy danh sách

```
>>> thislist = ["apple", "banana", "cherry"]
>>> mylist = thislist.copy()
>>> print(mylist)
['apple', 'banana', 'cherry']
>>> mylist.append(3)
>>> print(mylist)
['apple', 'banana', 'cherry', 3]
>>> print(thislist)
['apple', 'banana', 'cherry']
```


Danh sách - List

- Chú ý phép gán

```
1 list2 = list1
```

- Chỉ là phép gán tham chiếu, khi đó việc thay đổi 1 trong 2 danh sách thì danh sách kia cũng thay đổi theo.

```
>>> list = [1,2,3,4]
>>> list1 = [1,2,3,4]
>>> list2 = list1
>>> list2.append(5)
>>> print(list1)
[1, 2, 3, 4, 5]
>>> print(list2)
[1, 2, 3, 4, 5]
>>> list1.append(5)
>>> print(list2)
[1, 2, 3, 4, 5, 5]
>>> print(list2)
```

Danh sách - List

- Kết hợp hai danh sách

```
1 list1 = ["a", "b" , "c"]
2 list2 = [1, 2, 3]
3 list3 = list1 + list2
4 print(list3)
```

- Một cách khác có thể để kết hợp 2 danh sách là dùng lệnh `extend()`

```
1 list1 = ["a", "b" , "c"]
2 list2 = [1, 2, 3]
3 list1.extend(list2)
4 print(list1)
```

- Hàm dựng của danh sách `list()`

```
1 thislist = list(("apple", "banana", "cherry")) # note
the double round-brackets
2 print(thislist)
```

Danh sách - List, Ví dụ

```
1  def sumEven(a):
2  return sum(a[i] for i in range(0, len(a)) if a[i]%2 == 0)
3
4  def sumOdd(a):
5  return sum(i for i in a if i%2 == 1)
6
7  def sumList(a):
8  return sum(a)
9
10 n = int(input('Enter the number of elements in array n =
11 '))
12 aList = []
13 for i in range(0, n):
14     aList.append( int(input('list[' + str(i) + '] = ')))
15 print('Sum of even numbers in array = ', sumEven(aList))
16 print('Sum of odd numbers in array = ', sumOdd(aList))
17 print('Sum of all elements in array = ', sumList(aList))
```

Danh sách - List, Ví dụ

```
1 aList = [0 for j in range(0, n)]  
2  
3 for i in range(0, n):  
4 aList[i] = int(input('list[' + str(i) + '] = '))
```

Danh sách - List, Ví dụ

```
1 def wordCount(s):
2     tokens = s.split(' ')
3     words = list(dict.fromkeys(tokens))
4     counts = [tokens.count(words[i]) for i in range(0,
5         len(words))]
6     return words, counts
7
8 text = "the quick brown fox jumps over the lazy dog "
9
10 words, counts = wordCount(text)
11
12 for i in range(0, len(words)):
13     print(words[i], ': ', counts[i])
```

Danh sách - List, Ví dụ

```
1 >>> fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi',  
2             'apple', 'banana']  
3  
4 >>> fruits.count('apple')  
5 2  
6 >>> fruits.count('tangerine')  
7 0  
8 >>> fruits.index('banana')  
9 3  
10 >>> fruits.index('banana', 4) # Find next banana starting a  
11    position 4  
12 6  
13 >>> fruits.append('grape')  
14 >>> fruits  
15 ['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange',  
16    'grape']  
17 >>> fruits.sort()  
18 >>> fruits  
19 ['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi',  
20    'orange', 'pear']  
21 >>> fruits.pop() # 'pear'
```

Sử dụng danh sách như ngăn xếp

```
1 >>> stack = [3, 4, 5]
2 >>> stack.append(6)
3 >>> stack.append(7)
4 >>> stack
5 [3, 4, 5, 6, 7]
6 >>> stack.pop()
7 7
8 >>> stack
9 [3, 4, 5, 6]
10 >>> stack.pop()
11 6
12 >>> stack.pop()
13 5
14 >>> stack
15 [3, 4]
```

Sử dụng danh sách làm hàng đợi

```
1 >>> from collections import deque
2 >>> queue = deque(["Eric", "John", "Michael"])
3 >>> queue.append("Terry") # Terry arrives
4 >>> queue.append("Graham") # Graham arrives
5 >>> queue.popleft() # The first to arrive now leaves
6 'Eric'
7 >>> queue.popleft() # The second to arrive now leaves
8 'John'
9 >>> queue # Remaining queue in order of arrival
10 deque(['Michael', 'Terry', 'Graham'])
```


Bộ - tuple

- Bộ là một tập hợp có thứ tự và không thể thay đổi, trong Python bộ được khai báo trong cặp ngoặc đơn ().

```
1 thistuple = ("apple", "banana", "cherry")
2 print(thistuple) #>>>('apple', 'banana', 'cherry')
```

- Truy cập đến các phần tử của bộ cũng giống như việc truy cập đến các phần tử trong danh sách.

```
1 thistuple = ("apple", "banana", "cherry")
2 print(thistuple[1]) #>>>banana
3
4 thistuple = ("apple", "banana", "cherry", "orange", "kiwi",
               "melon", "mango")
5 print(thistuple[-1]) #>>>mango
6 print(thistuple[2:5]) #>>>('cherry', 'orange', 'kiwi')
7 print(thistuple[-4:-1]) #>>> ???
```

Bộ - tuple

- Khi một bộ được tạo ra thì các giá trị của các phần tử trong bộ không thể thay đổi được, tuy nhiên chúng ta có thể chuyển đổi bộ sang danh sách để thực hiện các thay đổi nếu cần thiết.

```
1 x = ("apple", "banana", "cherry")
2 y = list(x)
3 y[1] = "kiwi"
4 x = tuple(y)
5 print(x) #>>>('apple', 'kiwi', 'cherry')
```

- Duyệt qua các phần tử trong bộ

```
1 thistuple = ("apple", "banana", "cherry")
2 for x in thistuple:
3     print(x)
```

- Kiểm tra phần tử có trong bộ hay không

```
1 thistuple = ("apple", "banana", "cherry")
2 if "apple" in thistuple:
3     print("Yes, 'apple' is in the fruits tuple")
```


Bộ - tuple

- Gộp bộ

```
1 tuple1 = ("a", "b", "c")
2 tuple2 = (1, 2, 3)
3
4 tuple3 = tuple1 + tuple2
5 print(tuple3) #>>>('a', 'b', 'c', 1, 2, 3)
```

- Hàm dựng

```
1 thistuple = tuple(("apple", "banana", "cherry"))
2 # note the double round-brackets
3 print(thistuple)
```

- Đếm số phần tử có trong bộ

```
1 print(thistuple.count("cherry"))
```

- Tìm vị trí của phần tử trong bộ

```
1 print(thistuple.index("cherry"))
```

Một số ví dụ về bộ - tuple

```
1 >>>t = 12345, 54321, 'hello!'
2 >>>type(t)
3 <class 'tuple'>
4 >>> t = (12345, 54321, 'hello!')
5 >>> # Tuples may be nested:
6 ... u = t, (1, 2, 3, 4, 5)
7 >>> u
8 ((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
9
10 >>> # Tuples are immutable:
11 ... t[0] = 88888
12 Traceback (most recent call last):
13   File "<stdin>", line 1, in <module>
14 TypeError: 'tuple' object does not support item assignment
15 >>> # but they can contain mutable objects:
16 ... v = ([1, 2, 3], [3, 2, 1])
17 >>> v
18 ([1, 2, 3], [3, 2, 1])
```

Một số ví dụ về bộ - tuple

```
1 >>> empty = ()
2 >>> singleton = 'hello', # <-- note trailing comma
3 >>> len(empty)
4 0
5 >>> len(singleton)
6 1
7 >>> singleton
8 ('hello',)
```

Tập hợp - Set

- Tập hợp là một bộ không có thứ tự và không có chỉ số, trong Python một tập hợp được đặt trong cặp ngoặc nhọn `{}`.

```
1 thisset = {"apple", "banana", "cherry"}  
2     print(thisset)  
3     for x in thisset:  
4         print(x)  
5     print("banana" in thisset)
```

- Khi được tạo ra các phần tử trong tập hợp không thể thay đổi giá trị, nhưng có thể thêm phần tử mới vào tập hợp
- Để thêm một phần tử vào tập hợp sử dụng phương thức `add()`

```
1 thisset = {"apple", "banana", "cherry"}  
2 thisset.add("orange")  
3 print(thisset)
```

Tập hợp - Set

- Để thêm nhiều phần tử vào tập hợp ta sử dụng phương thức `update()`

```
1 thisset = {"apple", "banana", "cherry"}  
2 thisset.update(["orange", "mango", "grapes"])  
3 print(thisset)
```

- Kích thước tập hợp

```
1 thisset = {"apple", "banana", "cherry"}  
2 print(len(thisset))
```

- Xóa phần tử khỏi tập hợp `remove()/discard()`

```
1 thisset = {"apple", "banana", "cherry"}  
2 thisset.remove("banana")  
3 print(thisset)  
4 thisset.discard("cherry")  
5 print(thisset)
```


Tập hợp - Set

- Phương thức **pop()** cũng có thể được sử dụng để xóa phần tử khỏi tập hợp, tuy nhiên phương thức **pop()** sẽ xóa phần tử cuối cùng nhưng do tập hợp không có tính thứ tự nên chúng ta không biết được phần tử nào sẽ bị xóa khỏi tập hợp

```
1 thisset = {"apple", "banana", "cherry"}
2 x = thisset.pop()
3 print(x)
4 print(thisset)
```

- Phương thức **clear()** xóa tất cả các phần tử trong tập hợp

```
1 thisset = {"apple", "banana", "cherry"}
2 thisset.clear()
3 print(thisset)
4
5 # The del method deletes the entire set from memory
6 thisset = {"apple", "banana", "cherry"}
7 del thisset
8 print(thisset)
```

Tập hợp - Set

- Hợp hai tập hợp `union()/update()`

```
1 set1 = {"a", "b" , "c"}
2 set2 = {1, 2, 3}
3 set3 = set1.union(set2)
4 print(set3)
5
6 set1.update(set2)
7 print(set1)
```

- Hàm dựng của tập hợp

```
1 thisset = set(("apple", "banana", "cherry"))
2 # note the double      round-brackets
3 print(thisset)
```

Một số ví dụ trên Tập hợp - Set

```
1 >>> a = {x for x in 'abracadabra' if x not in 'abc'}
2 >>> a
3 {'r', 'd'}
4
5 >>> basket = {'apple', 'orange', 'apple', 'pear', 'orange',
6               'banana'}
7 >>> print(basket) # show that duplicates have been removed
8 {'orange', 'banana', 'pear', 'apple'}
9 >>> 'orange' in basket # fast membership testing
10 True
11 >>> 'crabgrass' in basket
False
```

Một số ví dụ trên Tập hợp - Set

```
1 >>> a = set('abracadabra')
2 >>> b = set('alacazam')
3 >>> a # unique letters in a
4 {'a', 'r', 'b', 'c', 'd'}
5 >>> a - b # letters in a but not in b
6 {'r', 'd', 'b'}
7 >>> a | b # letters in a or b or both
8 {'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
9 >>> a & b # letters in both a and b
10 {'a', 'c'}
11 >>> a ^ b # letters in a or b but not both
12 {'r', 'd', 'b', 'm', 'z', 'l'}
```

Từ điển - Dictionary

- Từ điển là một bộ không có thứ tự, có thể thay đổi được và có chỉ số. Trong Python, một từ điển được viết trong cặp ngoặc nhọn và mỗi phần tử trong từ điển là một cặp gồm **khóa** và **giá trị**.

```
1 thisdict = {  
2     "brand": "Ford",  
3     "model": "Mustang",  
4     "year": 1964  
5 }  
6 print(thisdict)  
7 >>>{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Từ điển - Dictionary

- Truy cập các phần tử trong từ điển
Giá trị của mỗi phần tử trong từ điển có thể được lấy thông qua khóa của nó.

```
1 thisdict = { "brand": "Ford",  
2             "model": "Mustang",  
3             "year": 1964}  
4     print(thisdict)  
5 >>>{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}  
6 x = thisdict["model"]
```

Giá trị của một khóa cũng có thể được lấy thông qua phương thức `get()`

```
1 x = thisdict.get("model")
```

Thay đổi giá trị

```
1 thisdict["year"] = 2019
```

Từ điển - Dictionary

- Duyệt các phần tử trong từ điển
Dùng vòng lặp để duyệt qua các khóa trong từ điển

```
1 for x in thisdict:
2     print(x)
3 # các khóa của từ điển ("brand", "model", "year")
```

- Lấy từng giá trị trong từ điển

```
1 for x in thisdict:
2     print(thisdict[x])
```

- Sử dụng phương thức **values()** để lấy ra danh sách các giá trị có trong từ điển

```
1 for x in thisdict.values():
2     print(x)
```

- Sử dụng phương thức **items()** lấy các phần tử trong từ điển là các cặp khóa – giá trị

```
1 for x, y in thisdict.items():
```

Từ điển - Dictionary

- Sử dụng phương thức `items()` lấy các phần tử trong từ điển là các cặp khóa – giá trị

```
1 for x, y in thisdict.items():  
2     print(x, y)
```

- Kiểm tra xem khóa có trong từ điển hay không

```
1 if "model" in thisdict:  
2     print("Yes, 'model' is one of the keys in the  
   thisdict dictionary")  
3 # Kích thước từ điển  
4 print(len(thisdict))  
5 # Thêm phần tử vào từ điển  
6 thisdict["color"] = "red"  
7     print(thisdict)
```


Từ điển - Dictionary

- Xóa phần tử khỏi từ điển

```
1 # Xóa phần tử theo khóa bằng phương thức pop()
2     thisdict.pop("model")
3     print(thisdict)
4 # Xóa phần tử ngẫu nhiên bằng phương thức popitem()
5     thisdict.popitem()
6 # Xóa phần tử theo khóa bằng phương thức del
7     del thisdict["model"]
8 # Xóa cả từ điển bằng phương thức del
9     del thisdict
10 # Xóa trống từ điển bằng phương thức clear()
11     thisdict.clear()
```

Từ điển - Dictionary

● Tạo bản sao của từ điển

```
1 # Tạo bản sao của từ điển bằng phương thức copy()
2     thisdict = {
3         "brand": "Ford",
4         "model": "Mustang",
5         "year": 1964
6     }
7     mydict = thisdict.copy()
8     print(mydict)
9 # Một cách khác để tạo bản sao của từ điển là
10 # sử dụng phương thức dict()
11     mydict = dict(thisdict)
```

Từ điển - Dictionary

• Từ điển lồng nhau

```
1 myfamily = {  
2     "child1" : {  
3         "name" : "Emil",  
4         "year" : 2004  
5     },  
6     "child2" : {  
7         "name" : "Tobias",  
8         "year" : 2007  
9     },  
10    "child3" : {  
11        "name" : "Linus",  
12        "year" : 2011  
13    }  
14 }
```

Từ điển - Dictionary

• Từ điển lồng nhau

```
1 child1 = {  
2     "name" : "Emil",  
3     "year" : 2004  
4 }  
5 child2 = {  
6     "name" : "Tobias",  
7     "year" : 2007  
8 }  
9 child3 = {  
10    "name" : "Linus",  
11    "year" : 2011  
12 }  
13  
14 myfamily = {  
15     "child1" : child1,  
16     "child2" : child2,  
17     "child3" : child3  
18 }  
19
```

Từ điển - Dictionary

● Hàm dựng của từ điển

```
1  
2 thisdict = dict(brand="Ford", model="Mustang", year=1964)  
3 # note that keywords are not string literals  
4 # note the use of equals rather than colon for the  
   assignment  
5 print(thisdict)  
6 >>> {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Một số ví dụ với Từ điển - Dictionary

```
1 >>> tel = {'jack': 4098, 'sape': 4139}
2 >>> tel['guido'] = 4127
3 >>> tel
4 {'jack': 4098, 'sape': 4139, 'guido': 4127}
5 >>> tel['jack']
6 4098
7 >>> del tel['sape']
8 >>> tel['irv'] = 4127
9 >>> tel
10 {'jack': 4098, 'guido': 4127, 'irv': 4127}
11
12 >>> list(tel)
13 ['jack', 'guido', 'irv']
14 >>> sorted(tel)
15 ['guido', 'irv', 'jack']
16 >>> 'guido' in tel
17 True
18 >>> 'jack' not in tel
19 False
```

Một số ví dụ với Từ điển - Dictionary

```
1 >>> dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])
2 {'sape': 4139, 'guido': 4127, 'jack': 4098}
3
4 >>> {x: x**2 for x in (2, 4, 6)}
5 {2: 4, 4: 16, 6: 36}
6
7 >>> dict(sape=4139, guido=4127, jack=4098)
8 {'sape': 4139, 'guido': 4127, 'jack': 4098}
9
10 >>> knights = {'gallahad': 'the pure', 'robin': 'the brave'}
11 >>> for k, v in knights.items():
12 ...     print(k, v)
13 ...
14 gallahad the pure
15 robin the brave
```

Một số ví dụ với Từ điển - Dictionary

```
1 # ví dụ về enumerate (liệt kê)
2 >>> for i, v in enumerate(['tic', 'tac', 'toe']):
3     ... print(i, v)
4     ...
5 0 tic
6 1 tac
7 2 toe
```


Một số ví dụ với Từ điển - Dictionary

```
1 # Hàm zip() để duyệt trên nhiều chuỗi đồng thời
2 >>> questions = ['name', 'quest', 'favorite color']
3 >>> answers = ['lancelot', 'the holy grail', 'blue']
4 >>> for q, a in zip(questions, answers):
5 ...     print('What is your {0}? It is {1}.'.format(q, a))
6 ...
7 What is your name? It is lancelot.
8 What is your quest? It is the holy grail.
9 What is your favorite color? It is blue.
```

Một số ví dụ với Từ điển - Dictionary

```
1 # Lặp theo thứ tự đảo ngược
2 >>> for i in reversed(range(1, 10, 2)):
3     ...     print(i)
4     ...
5 9
6 7
7 5
8 3
9 1
```

Một số ví dụ với Từ điển - Dictionary

```
1 # Lặp trên chuỗi được sắp xếp
2 >>> basket = ['apple', 'orange', 'apple', 'pear', 'orange',
3               'banana']
4 >>> for f in sorted(set(basket)):
5     ...     print(f)
6 apple
7 banana
8 orange
9 pear
```

Một số ví dụ với Từ điển - Dictionary

```
1 # Tạo một danh sách mới lưu kết quả (đơn giản và an toàn hơn)
2 >>> import math
3 >>> raw_data = [56.2, ('NaN'), 51.7, 55.3, 52.5, ('NaN'), 47.8]
4 >>> filtered_data = []
5 >>> for value in raw_data:
6 ...     if not math.isnan(value):
7 ...         filtered_data.append(value)
8 ...
9 >>> filtered_data
10 [56.2, 51.7, 55.3, 52.5, 47.8]
```

Một số ví dụ với Từ điển - Dictionary

```
1 # Tạo một danh sách mới lưu kết quả (đơn giản và an toàn hơn)
2 >>> import math
3 >>> raw_data = [56.2, ('NaN'), 51.7, 55.3, 52.5, ('NaN'), 47.8]
4 >>> filtered_data = []
5 >>> for value in raw_data:
6 ...     if not math.isnan(value):
7 ...         filtered_data.append(value)
8 ...
9 >>> filtered_data
10 [56.2, 51.7, 55.3, 52.5, 47.8]
```

Một số ví dụ với Từ điển - Dictionary

```
1 # So sánh theo bộ
2 (1, 2, 3) < (1, 2, 4)
3 [1, 2, 3] < [1, 2, 4]
4 'ABC' < 'C' < 'Pascal' < 'Python'
5 (1, 2, 3, 4) < (1, 2, 4)
6 (1, 2) < (1, 2, -1)
7 (1, 2, 3) == (1.0, 2.0, 3.0)
8 (1, 2, ('aa', 'ab')) < (1, 2, ('abc', 'a'), 4)
```