LẬP TRÌNH PYTHON

Bài 6: Kiểu tập hợp và kiểu tập tĩnh

Tóm tắt nội dung bài trước

- Ngoài kiểu chuỗi (str), Python có những kiểu dữ liệu tuần tự khác: danh sách (list), hàng (tuple), miền (range)
- Danh sách là kiểu tuần tự mạnh mẽ và uyển chuyển:
  - Có thể chứa bên trong nó tất cả các loại dữ liệu
  - Nhiều cách khởi tạo:
    - Khai báo trực tiếp trong cặp ngoặc vuông
    - Khởi tạo bằng hàm list
    - Khởi tạo bằng một đoạn for ngắn (bộ suy diễn danh sách)
  - Duyệt các phần tử con bằng vòng lặp hoặc truy cập qua chỉ số
  - Hỗ trợ các phép toán: ghép nối (+), nhân bản (*), kiểm tra (in)
  - Hai danh sách có thể so sánh với nhau theo thứ tự từ điển
  - Phép cắt lát cho phép lấy phần con của danh sách dễ dàng
  - Rất nhiều phương thức hỗ trợ khác

Tóm tắt nội dung bài trước

- Hàng cũng là một dãy các dữ liệu như danh sách, nhưng không thể thay đổi sau khi khởi tạo
  - Cũng có 3 cách khởi tạo:
    - Khai báo trực tiếp trong cặp ngoặc tròn
    - Khởi tạo bằng hàm tuple
    - Khởi tạo bằng hàm sinh (một loại bộ suy diễn dành cho kiểu hàng)
  - Cũng duyệt phần tử con bằng for hoặc truy cập qua chỉ mục
  - Cũng hỗ trợ các phép toán +, *, in và cắt lát
  - Các phương thức hỗ trợ chỉ có đếm (count) và tìm kiếm (index)
  - Hàng nhanh hơn danh sách, vì hàng "tĩnh" hơn
- Miền là kiểu dữ liệu thiết kế đặc biệt cho vòng for theo chỉ số, nhưng cũng có một vài đặc trưng của kiểu dữ liệu tuần tự như kiểm tra, chỉ mục, cắt lát,...

Nội dung

1. Set (tập hợp)
   - Khởi tạo
   - Phép toán
   - Duyệt các phần tử
   - Các phương thức hỗ trợ
2. Frozenset (tập hợp tĩnh)
3. Bài tập

Phần 1

Set (tập hợp)

Giới thiệu

- Tập hợp (set) là kiểu dữ liệu đặc sắc trong Python, lấy cảm hứng từ khái niệm tập hợp trong toán học
  - Các đối tượng con đôi một khác nhau: nếu đưa các đối tượng giống nhau vào tập hợp, Python sẽ chỉ giữ lại một
  - Không có tính thứ tự: không thể truy cập đến phần tử con thông qua hệ thống chỉ mục
  - Không phải dữ liệu nào cũng đưa được vào tập hợp: dữ liệu con bắt buộc phải là dạng bất biến (immutable)
  - Thêm: Python sử dụng cấu trúc dữ liệu bảng băm (hashtable) cho tập hợp, đây chính là lý do dữ liệu con phải bất biến để tránh việc dữ liệu bị thay đổi một cách bất lường

Khởi tạo

- Tương tự như danh sách và hàng, khởi tạo tập hợp đơn giản nhất bằng cách liệt kê các phần tử con:
  - Đặt trong cặp ngoặc nhọn {}
  - Ngăn cách bởi phẩy
  - Chú ý: cách này không dùng để khởi tạo tập rỗng (hãy thử xem)

```
>>> basket = {'apple', 'orange', 'apple', 'pear'}
>>> print(basket)
{'orange', 'pear', 'apple'}    # xóa phần tử trùng nhau
```

- Tạo tập hợp bằng hàm tạo hoặc các phép toán tập hợp

```
s1 = set([1, 2, 3, 4])      # {1, 2, 3, 4} – copy từ list
s2 = set((1, 1, 1))         # {1} – copy từ tuple, bỏ lặp
s3 = s1 – s2                # {2, 3, 4} – hiệu của hai tập
s4 = set(range(1, 100))     # {1, 2, 3,…, 98, 99}
s5 = set()                  # {} – tập rỗng
```

Khởi tạo

- Tập hợp cũng có thể khởi bằng bộ suy diễn tập hợp (set comprehension), cú pháp tương tự như danh sách
- Dạng thông dụng nhất: lặp
  - `{ <biểu thức> for <biến> in <tuần tự> }`
  - `{ x for x in 'abracadabra' }`
- Dạng phức tạp hơn: lặp và điều kiện
  - `{ <biểu thức> for <biến> in <tuần tự> if <điều kiện> }`
  - `{ x for x in 'abracadabra' if x not in 'abc' }`
- Dạng phức tạp hơn nữa: lặp và điều kiện rẽ nhánh
  - Trường hợp này phải kết hợp lặp và phép toán if (không dùng rẽ nhánh if được)
  - `{ <A> if <điều kiện> else <B> for <biến> in <tuần tự> }`
  - `{ '?' if x in 'abc' else x for x in 'abracadabra' }`

Khởi tạo

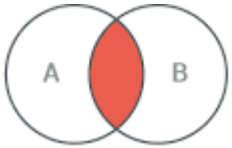
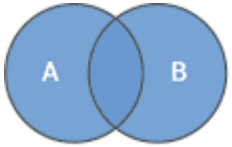
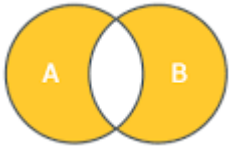
- Bộ suy diễn tập hợp đôi khi khá phức tạp
- Ví dụ: tạo tập hợp chứa mọi hoán vị của chuỗi ‘abc’

```
s = {
    x + y + z
    for x in 'abc'
        for y in 'abc'
            for z in 'abc'
                if x != y != z != x
}
```

- Set không thể chứa những đối tượng mutable (có thể bị thay đổi), mặc dù chính set lại có thể thay đổi

```
a = set(([1,2], [2,3]))  # lỗi
a = set(((1,2), (2,3)))  # {(1, 2), (2, 3)}
a.add("abc")             # {(1, 2), "abc", (2, 3)}
```

Các phép toán trên set

STT	Tên	Kí hiệu	Giải thích	Minh họa
1	Phép giao	**&**	Lấy phần chung của hai tập	
2	Phép hợp	**|**	Lấy phần gộp của hai tập	
3	Phép hiệu	**-**	Lấy phần riêng của một tập	
4	Phép loại	**^**	Lấy phần khác (loại bỏ phần chung)	
5	Kiểm tra tồn tại	**in**	Trả về **True** nếu phần tử nằm trong tập hợp	
6	Kiểm tra không thuộc	**not in**	Trả về **True** nếu phần tử không thuộc tập hợp	

Các phép toán trên set

STT	Tên	Kí hiệu	Giải thích
7	So sánh “bằng”	==	Trả về **True** nếu hai tập giống nhau
8	So sánh “khác”	!=	Trả về **True** nếu có ít nhất một phần tử thuộc tập này mà không thuộc tập kia
9	So sánh “lớn hơn”	>	Trả về **True** nếu mọi phần tử của tập thứ hai đều có trong tập thứ nhất và có ít nhất một phần tử thuộc tập thứ nhất không xuất hiện trong tập thứ hai
10	So sánh “lớn hơn hoặc bằng”	>=	Trả về **True** nếu tập thứ nhất bằng hoặc lớn hơn tập thứ hai
11	So sánh “nhỏ hơn”	<	Trả về **True** nếu mọi phần tử thuộc tập thứ nhất đều có trong tập thứ hai và có ít nhất một phần tử thuộc tập thứ hai không xuất hiện trong tập thứ nhất
12	So sánh “nhỏ hơn hoặc bằng”	<=	Trả về **True** nếu tập thứ nhất bằng hoặc nhỏ hơn tập thứ hai

Các phép toán trên set

```
a = set('abracadabra')      # {'d', 'r', 'c', 'b', 'a'}
b = set('alacazam')         # {'z', 'c', 'm', 'l', 'a'}

# Phép Hiệu: thuộc a nhưng không thuộc b
print(a - b)                # {'r', 'd', 'b'}
# Phép Hợp: thuộc a hoặc b
# {'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
print(a | b)
# Phép Giao: thuộc cả a và b
print(a & b)                # {'a', 'c'}
# Phép Xor: thuộc hoặc a, hoặc b nhưng không phải cả 2
# {'r', 'd', 'b', 'm', 'z', 'l'}
print(a ^ b)
```

Duyệt các phần tử của tập hợp

- Cách đơn giản nhất là duyệt bằng vòng for

```
a = { 10, 2, 3, 1, 3, 2, 10 }
for e in a:
    print(e)
```

- Cẩn thận: thứ tự in kết quả của vòng for không nhất thiết phải theo thứ tự các phần tử đưa vào tập hợp
- Trường hợp cần duyệt theo chỉ mục, sử dụng kết hợp với hàm enumerate:

```
a = { 10, 2, 3, 1, 3, 2, 10 }
for index, value in enumerate(a):
    print(index, value)
```

Các phương thức của set

- Một số phương thức thường hay sử dụng
  - add(e): thêm e vào tập hợp
  - clear(): xóa mọi phần tử trong tập hợp
  - copy(): tạo một bản sao của tập hợp
  - difference(x): tương đương với phép trừ đi x
  - difference_update(x): loại bỏ những phần tử trong x khỏi tập (cập nhật vào tập hiện tại)
  - discard(e): bỏ e khỏi tập
  - intersection(x): tương đương với phép giao với x
  - intersection_update(x): tương đương với phép giao với x (cập nhật vào tập hiện tại)

Các phương thức của set

- Một số phương thức thường hay sử dụng
  - isdisjoint(x): trả về True nếu tập không có phần chung nào với x
  - issubset(x): trả về True nếu tập là con của x, tương đương với phép so sánh <=x
  - issuperset(x): trả về True nếu x là tập con của tập, tương đương với phép so sánh >=x
  - pop(): lấy một phần tử ra khỏi tập (không biết trước)
  - remove(e): bỏ e khỏi tập, báo lỗi nếu không tìm thấy e
  - symmetric_difference(x): tương đương với phép ^x
  - symmetric_difference_update(x): tương đương với phép ^x (cập nhật vào tập hiện tại)
  - union(x): tương đương với phép hợp với x
  - update(x): đưa các phần tử của tập x vào tập hiện tại

Phần 2

Frozenset (tập hợp tĩnh)

Frozenset (tập tĩnh)

- Frozenset giống set, nhưng không thể bị thay đổi

```
b = frozenset(((1,2), (2,3)))  # {(1, 2), (2, 3)}
b.add("abc")                   # lỗi
```

- Sự khác biệt giữa tập tĩnh và tập hợp:
  - Tập tĩnh là kiểu bất biến, tập hợp là kiểu khả biến (dù các thành phần của nó phải là bất biến)
  - Tập tĩnh thường nhanh và ít tốn bộ nhớ hơn tập hợp
  - Tập tĩnh không sử dụng được các phương thức thay đổi nội dung như add() hoặc remove()
  - Tập tĩnh vẫn hỗ trợ các phép toán của kiểu tập hợp (giao, hợp, hiệu,…)
  - Tập tĩnh vẫn hỗ trợ các phép so sánh với tập hợp
  - Tập tĩnh có thể dùng làm khóa cho dữ liệu từ điển (học sau)

Phần 3

Bài tập

Bài tập

1. Tạo một tập hợp gồm các phần tử từ 0 đến 99, in chúng ra màn hình
2. Tạo một tập hợp gồm các số nguyên lẻ trong khoảng từ 1 đến 199, in chúng ra màn hình
3. Tạo một tập hợp gồm các số nhập vào từ bàn phím (nhập trên 1 dòng, cách nhau bởi ký tự trống), tìm và in ra số phần tử của tập, giá trị lớn nhất và nhỏ nhất trong tập
4. Nhập vào từ bàn phím họ và tên đầy đủ của các sinh viên trong lớp, mỗi người trên một dòng. Việc nhập sẽ kết thúc khi người dùng gõ vào dòng trống. Sau đó, hãy in ra các họ và các tên của sinh viên trong lớp

Bài tập

5. Nhập số nguyên N, tạo một tập hợp các số nguyên dương d là ước số của N
6. Nhập 2 số nguyên a và b, hãy tạo ra một tập hợp các số d là ước số chung của cả a và b
7. Nhập một dãy số nguyên từ bàn phím, các số được viết liên tiếp, ngăn cách nhau bởi dấu chấm phẩy (;), hãy đếm xem dãy nhập vào có bao nhiêu số khác nhau
8. Vé Vietlott Mega là bộ 6 số chỉ từ 01 đến 45. Người chơi sẽ thắng nếu chọn đúng ít nhất 5 trong 6 số, thứ tự không quan trọng. Hãy viết chương trình nhập vào N bộ 6 số của N người, sau đó nhập tiếp 6 số của giải đặc biệt và in ra các bộ số của người chơi thắng cuộc.

Bài tập

9. Một công ty có 3 phòng chức năng có dùng thế dùng chung nhân viên là phòng nhân sự, phòng hành chính và phòng truyền thông. Các nhân viên có mã nhân viên là các số nguyên dương. Hãy thực hiện các việc:
   a. Nhập danh sách mã nhân viên của cả 3 phòng, danh sách được viết liên tục trên một dòng, ngăn cách bởi dấu phấy
   b. Ba phòng ban này sử dụng bao nhiêu nhân viên?
   c. In ra danh sách các mã nhân viên thuộc cả 3 phòng
   d. In ra danh sách các mã nhân viên chỉ thuộc 1 phòng
   e. Tìm cặp phòng dùng chung nhiều nhân viên nhất, nếu có nhiều cặp phòng như vậy thì in ra tất cả các cặp
   f. Với từng phòng, in ra mã nhân viên đầu tiên của phòng (có mã nhỏ nhất)