XỬ LÝ DỮ LIỆU TRONG PYTHON

1

Nội dung

1. Kiểu dữ liệu tuần tự (sequential datatype)
2. String (chuỗi)
3. List (danhsách)
4. Tuple(hàng)
5. Range(miền)

2

Kiểu dữ liệu tuần tự (sequential datatype)

- Kiểu dữ liệu chứa bên trong nó các dữ liệu con nhỏ hơn, thường được xử lý bằng cách lấy ra từng phần-tử-một (bằng vòng lặp)
  - Các kiểu dữ liệu chứa bên trong nó các dữ liệu nhỏ hơn => các container (bộ chứa)
- Có 3 kiểu tuần tự thông dụng: **list**, **tuple** và **range**
- Có nhiều kiểu khác như **string, bytes, bytearray**,…

3

String (chuỗi)

- Một chuỗi được xem như một hàng (tuple) các chuỗi con độ dài 1.
  - Trong python không có kiểu kí tự (character)
- Hàm **len(s)** trả về độ dài (số chữ) của **s**
- Phép nối chuỗi (+): **s= "học" +" " +"nữa"**
- Phép nhân bản (*): ***s="AB" * 3***

4

Chỉ mục trong chuỗi

- Các các chữ trong chuỗi được đánh số thứ tự và có thể truy cập vào từng phần tử theo chỉ số.
- có 2 cách đánh chỉ mục:
  - Đánh từ trái qua phải: chỉ số đánh từ 0 trở đi cho đến cuối chuỗi
  - Đánh từ phải qua trái: chỉ số đánh từ -1 giảm dần về đầu chuỗi

L	O	V	E		Y	O	U
0	1	2	3	4	5	6	7
-8	-7	-6	-5	-4	-3	-2	-1

5

Cắt chuỗi

- Cho phép lấy nội dung bên trong của chuỗi:

```
<chuỗi>[vị trí X: vị trí Y]
<chuỗi>[vị trí Y: vị trí Y: bước nhảy]
```

- Tạo chuỗi con bắt đầu từ vị trí X đến trước vị trí Y (chuỗi con không gồm vị trí Y)
- Nếu không ghi vị trí X => mặc định lấy từ đầu
- Nếu không ghi vị trí Y => mặc định là đến hết chuỗi
- Nếu không ghi bước nhảy => mặc định bước nhảy là 1.
- Nếu bước nhảy giá trị âm thì sẽ nhận chuỗi ngược lại

6

Các phương thức chuỗi

- **Các phương thức chỉnh dạng**
  - **capitalize():** viết hoa chữ cái đầu, còn lại viết thường
  - **upper():** chuyển hết thành chữ hoa
  - **lower():** chuyển hết thành chữ thường
  - **swapcase():** chữ thường thành hoa và ngược lại
  - **title():** chữ đầu của mỗi từ viết hoa, còn lại viết thường
- **Các phương thức căn lề**
  - **center(width [,fillchar]):** căn lề giữa với độ dài width
  - **rjust(width [,fillchar]):** căn lề phải
  - **ljust(width [,fillchar]):** căn lề trái

7

List (danh sách)

- List = dãy các đối tượng
- Khai báo trực tiếp bằng cách liệt kê các phần tử con đặt trong ngoặc vuông ([ ]), cách nhau bởi dấu phẩy.
- Ví dụ:

```
[ ]                          # List rỗng
['x','y','z']                # List 3 chuỗi
[[11, 22], [13, 10]]         # List 2 list con
[5, 'xyz', [1, 'abc']]       # list hỗn hợp
```

- Kiểu chuỗi (str) có thể xem như một list, bên trong gồm toàn các **str** độ dài 1

8

Khởi tạo danh sách

```
a1 = list([4, 1, 3, 5]) # list 4 số nguyên
a2 = list('x y z ')     # list 3 chuỗi con
a3 = list()             # list rỗng

# list 50 số nguyên từ 0 đến 49
c = [n for n in range(50)]

# list gồm 5 list con là các cặp [a, a²]
# với a chạy từ 0 đến 5
Y = [[a, a*a] for a in range(5)]
```

9

9

Chỉ mục

- Chỉ mục của danh sách bắt đầu từ 0 đến n-1
- Truy cập đến các phần tử:

<ten_list>[index]

10

Một số hàm thường dùng

- **extend(x):** thêm các phần tử của x vào cuối list
- **insert (p, x):** chèn x vào vị trí p trong list
- **pop(p):** bỏ phần tử thứ p ra khỏi list (trả về giá trị của phần tử đó), nếu không chỉ định p thì lấy phần tử cuối
- **count(sub, [start, [end]]):** đếm số lần xuất hiện của sub
- **index(sub[, start[, end]]):** tìm vị trí xuất hiện của sub, trả về ValueError nếu không tìm thấy
- **clear():** xóa list
- append(x): thêm x vào cuối list

11

Một số hàm thường dùng

- **copy():** tạo bản sao của list (tương tự list[:])
- **remove(x):** bỏ phần tử đầu tiên trong list có giá trị x, báo lỗi ValueError nếu không tìm thấy
- **reverse():** đảo ngược các phần tử trong list

12

Hàm

- **cmp(list1, list2)**: so sánh các phần tử của 2 list
- **len(list)**: lấy về chiều dài của list
- **sum()**: Trả về tổng giá trị của các phần tử trong list. Hàm này chỉ làm việc với kiểu number.
- **max(list)**: Trả về phần tử có giá trị lớn nhất trong list
- **min(list)**: Trả về phần tử có giá trị nhỏ nhất trong list
- **list(seq)**: Chuyển đổi một tuple thành list

13

Tạo Tuple

- Tuple = dãy các đối tượng (list)
- Khai báo trực tiếp bằng cách liệt kê các phần tử con đặt trong cặp ngoặc tròn (), ngăn cách bởi dấu phẩy

```
(11, 32, 43, 56, 7)              # tuple 5 số nguyên
('ac', 'b', 'cx', 'yz')          # tuple 4 chuỗi
(23, 'abc', [1, 2, 'xyz'])       # tuple hỗn hợp
()                               # tuple rỗng
```

14

Tuple

- Tuple sử dụng chỉ mục giống List
- Tuple khác list:
  - Nhanh hơn
  - Chiếm ít bộ nhớ hơn

15

Ví dụ

Tạo tuble gồm 05 số nguyên

Hướng dẫn:

```
#tạo tuple 5 số nguyên nhap tu ban phim
count=0;
t=();
while count<5:
   count=count+1;
   x=int(input("Nhap :"));
   t=t+(x,);
print(t);
```

16

Tuple

- **count(v):** đếm số lần xuất hiện của v trong tuple
- **index(sub[, start[, end]]):** tương tự như str và list
- **all():** Trả về giá trị True nếu tất cả các phần tử của tuple là true hoặc tuple rỗng.
- **any():** Trả về True nếu bất kỳ phần tử nào của tuple là true, nếu tuple rỗng trả về False.
- **enumerated()**: Trả về đối tượng enumerate (liệt kê), chứa cặp index và giá trị của tất cả phần tử của tuple.
- **len():** Trả về độ dài (số phần tử) của tuple.

17

Ví dụ

Tạo tuble gồm 10 phần tử, bao gồm 3 phần tử trùng nhau

Hướng dẫn:

```
a=(1,2,3,3,3,4,5,6,7,8);
print(a.count(3));
```

18

Tuple

- **max():** Trả về phần tử lớn nhất của tuple.
- **min():** Trả về phần tử nhỏ nhất của tuple.
- **sorted():** Lấy phần tử trong tuple và trả về list mới được sắp xếp (tuple không sắp xếp được).
- **sum():** Trả về tổng tất cả các phần tử trong tuple.
- **tuple():** Chuyển đổi những đối tượng có thể lặp (list, string, set, dictionary) thành tuple.

19

Bài tập

Nhập vào một chuỗi gồm 10 phần tử, xuất các phần tử thứ 2,4,6,8,10

```
#nhập vào một chuỗi gồm 10 phần tử, xuất các phần tử thứ
2,4,6,8,10
a=("a","b","c","d","e","f","g","h","i","j",);
i=1;
while (i<len(a)):
   if(i%2!=0):
      print(a[i]);
   i=i+1;
```

20

Range (miền)

- **range(stop):** tạo miền từ 0 đến (stop-1)
- **range(start, stop, [step])**: tạo miền từ start đến (stop-1), với bước nhảy là step
  - Nếu không chỉ định thì step =1
  - Nếu step là số âm => tạo miền đếm giảm dần (start > stop)
- Range khác với một tuple:
  - Range chỉ chứa số nguyên
  - Range nhanh hơn rất nhiều
  - Range chiếm ít bộ nhớ hơn
  - Range vẫn hỗ trợ chỉ mục

21