Softech®
Gather to thrive together

Aptech™
COMPUTER EDUCATION
Unleash your potential

BÀI 5

Kiểu dữ liệu Lists, Tuples, Set trong Python

Tóm Tắt Nội Dung

Trong bài học này chúng ta sẽ đi tìm hiểu lần lượt các nội dung

1. Làm quen với các kiểu dữ liệu mảng với List
2. Các phương thức xử lý dữ liệu với List
3. Làm quen với các kiểu dữ liệu mảng với Tuple
4. Các phương thức xử lý dữ liệu với Tuple
5. Làm quen với các kiểu dữ liệu mảng với Set
6. Các phương thức xử lý dữ liệu với Set

5.1 Tổng quan về List

Giới thiệu về List (hay còn gọi là danh sách)

List là một kiểu dữ liệu được sử dụng để lưu trữ một tập hợp các phần tử trong dấu ngoặc vuông []

- Chứa các phần tử có cùng hoặc khác kiểu dữ liệu, cho phép trùng lặp phần tử
- Có thứ tự, tức là các phần tử trong list được sắp xếp theo một vị trí cụ thể và không thay đổi, **vị trí bắt đầu là số 0**
- Có thể thay đổi, tức là bạn có thể thêm, sửa đổi, hoặc xóa các phần tử trong list sau khi nó đã được tạo.

Ví dụ:

```
mylist = ["apple", "banana", "cherry"]
```

```
mylist = []
```

Các phần tử dược đặt trong dấu ngoặc vuông và cách nhau bằng dấu phẩy ,. Có thể là một list rỗng

5.2 Các phương thức tương tác với List

Truy cập đến phần tử của List

```
mylist = ["apple", "banana", "cherry“, "orange", "kiwi", "melon"]
print(mylist[1])
#output: banana
print(mylist[-1])
#output: cherry (chiều ngược lại)
```

Cú pháp: **list[start:end]**

- start (tùy chọn) mặc định là 0, có thể là số âm
- end (tùy chọn) mặc định lấy đến cuối list, , có thể là số âm

```
print(mylist[1:4])
#output: banana, cherry, orange (không tính kiwi)
```

5.2 Các phương thức tương tác với List

Truy cập đến phần tử của List

Ví dụ với start không được điền, thì mặc định start là 0

```
print(mylist[:4])
#output: banana, cherry, orange (không tính kiwi)
```

Ví dụ với end không được điền, thì mặc định lấy đến cuối danh sách

```
print(mylist[4:])
#output: kiwi, melon
```

Kiểm tra xem phần tử trong list có tồn tại không

```
if ‘apple’ in mylist:
    print(‘Apple có tồn tại’)
```

5.2 Các phương thức tương tác với List

Thay đổi giá trị của phần tử trong List

Thay đổi giá trị dựa vào **vị trí** của phần tử đó trong List

```
mylist[1] = 'blackcurrant'
print(mylist) #output: apple,  blackcurrant, cherry, orange, kiwi, melon
```

Thay đổi một lúc **nhiều** giá trị với **nhiều** vị trí tương tứng

```
mylist[1:3] = ['blackcurrant', 'watermelon']
print(mylist) #output: apple,  blackcurrant,  watermelon, orange, kiwi, melon
```

Nếu bạn có 1 vị trí, nhưng để nhiều giá trị thì nó sẽ chèn vào

```
mylist[1:2] = ['blackcurrant', 'watermelon']
print(mylist) #output: apple,  blackcurrant,  watermelon, cherry, orange, kiwi,
melon
```

5.2 Các phương thức tương tác với List

Thêm mới phần tử vào List

Sử dụng phương thức **append(element)** để thêm một phần tử vào cuối List

```
mylist = ["apple", "banana", "kiwi"]
mylist.append(“melon”)
print(mylist) #output: apple, banana, kiwi, melon
```

Sử dụng phương thức **insert(index, value)** để chèn vào giữa List

```
mylist = ["apple", "banana", "kiwi"]
mylist.insert(1, “melon”)
print(mylist) #output: apple,  melon , banana, kiwi
```

5.2 Các phương thức tương tác với List

Thêm mới phần tử vào List

Nối List (mở rộng list) với phương thức **extend(iterable)**

iterable: Đối tượng có thể lặp (list, tuple, set, string, etc.)

```
mylist = ["apple", "banana", "kiwi"]
tropical = ["mango", "pineapple", "papaya"]
mylist.extend(tropical)
print(mylist) #output: apple, banana, kiwi, mango, pineapple, papaya
```

Bạn có thể mở rộng list với **tuples, sets, dictionaries**… bằng **extend()**

```
mylist = ["apple", "banana", "kiwi"]
thistuple = ("kiwi", "orange")
mylist.extend(thistuple)
```

5.2 Các phương thức tương tác với List

Thêm mới phần tử vào List

Hoặc bạn có thể sử dụng toán tử **+** để join 2 List với nhau

```
mylist = ["apple", "banana", "kiwi"]
tropical = ["mango", "pineapple", "papaya"]
fruits = mylist + topical
print(fruits) #output: apple, banana, kiwi, mango, pineapple, papaya
```

5.2 Các phương thức tương tác với List

Xóa phần tử trong List

Sử dụng phương thức **remove()** để xóa phần tử mong muốn

```
mylist = ["apple", "banana", "kiwi"]
mylist.remove("banana")
print(mylist) #output: apple, kiwi
```

Xóa một phần tử dựa vào index với phương thức **pop(index).** Mặc định -1

```
mylist = ["apple", "banana", "kiwi"]
mylist.pop (1) #hoặc del mylist[1]
print(mylist) #output: apple, kiwi
```

Nếu pop() **không** truyền tham số, thì nó xóa phần tử ở cuối List

5.2 Các phương thức tương tác với List

Xóa tất cả trong List

Dùng lệnh **del** để xóa luôn List

```
mylist = ["apple", "banana", "kiwi"]
del mylist
```

Hoặc bạn có thể xóa tất cả phần tử trong List, còn lại list Rỗng với **clear()**

```
mylist = ["apple", "banana", "kiwi"]
mylist.clear ()
print(mylist) #output: [] một List rỗng
```

5.2 Các phương thức tương tác với List

Lặp các phần tử của List với for

Dùng lệnh **for** đã học bài trước để lặp qua các phần tử của List

```
mylist = ["apple", "banana", "kiwi"]
for x in mylist:
    print(x)
```

Hoặc cú pháp ngắn (short hand)

```
[print(x) for x in mylist]
```

Lặp qua và in ra vị trí index của phần tử trong List

```
for i in range(len(mylist)):
    print(i)
```

5.2 Các phương thức tương tác với List

Phương thức	Mô tả
list.append(element)	Thêm phần tử vào cuối List
list.clear()	Xóa tất cả phần tử trong mảng, cho ra List rỗng
new_list = list.copy()	Trả lại bản copy của List
list.count(element)	Đếm số lần xuất hiện của phần tử trong List
list.extend(iterable)	Mở rộng List hay gộp các List lại với nhau
list.index(element, start, end)	Trả về vị trí của phần tử trong List
list.insert(index, element)	Chèn phần tử vào giữa List
list.pop(index)	Xóa phần tử dựa vào vị trí chỉ định
list.remove(element)	Xóa phần tử dựa vào giá trị của phần tử
list.sort(key=None, reverse=False)	Sắp xếp phần tử trong List

5.3 Tổng quan về Tuple

Giới thiệu về Tuple (hay còn gọi là dãy số)

Tuple là một kiểu dữ liệu được sử dụng để lưu trữ một tập hợp các phần tử trong dấu ngoặc tròn ()

- Không thể thay đổi giá trị các phần tử bên trong tuple sau khi tạo ra
- Có thể truy cập các phần tử bên trong tuple dựa vào index
- Tuple có thể chứa các kiểu dữ liệu khác nhau: số nguyên, số thực, chuỗi, boolean, và thậm chí tuple khác
- Có thể trùng lặp các phần tử trong tuple

```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
```

Các phần tử dược đặt trong dấu ngoặc tròn và cách nhau bằng dấu phẩy ,

5.3 Tổng quan về Tuple

Ví dụ về Tuple

Tuple với 1 giá trị, lưu ý phải có dấu phẩy , phía sau phần tử đó

```
thistuple = ("apple",)
print(type(thistuple)) #output: tuple
```

Cú pháp sau đây có vẻ ok nhưng nó không phải tuple mà là str

```
thistuple = ("apple")
print(type(thistuple)) #output: str
```

Tuple chấp nhận nhiều kiểu giá trị

```
thistuple = ("abc", 34, True, 40, "male")
print(thistuple)
```

5.4 Các phương thức tương tác với Tuple

Truy cập đến phần tử của Tuple

Truy cập đến phần tử dựa vào vị trí index của phần tử đó trong Tuple

```
thistuple = ("apple", "banana", "cherry“, "orange", "kiwi", "melon")
print(thistuple[1])  #output: banana
```

Lấy theo chiều ngược lại (phải qua trái)

```
print(thistuple[-1]) #output: melon
```

Lấy theo một khoảng: [start:end] – start mặc định là 0, end mặc định vị trí cuối cùng

```
print(thistuple[1:2])
#Output: ('banana',)
```

5.4 Các phương thức tương tác với Tuple

Thay đổi giá trị của phần tử trong Tuple

Theo lý thuyết mỗi khi tuple được tạo thì bạn không thể thay đổi, thêm, sửa, xóa. Nhưng có một số cách giải quyết.

Convert một Tuple thành List ➔ Thay đổi ➔ Conver List thành Tuple

```
x = ("apple", "banana", "cherry")
y = list(x) #chuyển tuple thành list
y[1] = "kiwi“#thay đổi giá trị banane thành kiwi
x = tuple(y) #chuyển list thành tuple lại

print(x)
```

Như vậy bạn có thể áp dụng mọi phương thức của List để thay đổi

5.4 Các phương thức tương tác với Tuple

Phân rã một Tuple

Khi tạo ra một tuple thì có nghĩa bạn đang đóng gói nó (packing)

Trong Python bạn có thể mở gói hay xả một tuple (unpacking)

```
fruits = ("apple", "banana", "cherry")
(green, yellow, red) = fruits
print(green)
print(red)
print(yellow)
```

Trong ví dụ trên xả fruits ra 3 giá trị gán cho 3 biến green, yellow, red

5.4 Các phương thức tương tác với Tuple

Phân rã và chỉ lấy một phần của Tuple

Nếu bạn chỉ muốn lấy một phần tử trong Tuple với cách thức phân rã, bạn có thể sử dụng dấu _ (gạch dưới) để bỏ qua những phần tử không muốn lấy

```
fruits = ("apple", "banana", "cherry")
(_, selected_fruit, *_) = fruits
print(selected_fruit) # output: banana
```

Trong trường hợp này, chúng ta sử dụng _ để bỏ qua phần tử "apple", "*_" để bỏ qua các phần tử còn lại trong tuple fruits. Các phần tử được bỏ qua không được lưu trữ trong bất kỳ biến nào. Biến selected_fruit sẽ chứa giá trị "banana" và bạn có thể sử dụng biến đó trong các phần khác của chương trình của mình.

5.4 Các phương thức tương tác với Tuple

Phân rã một Tuple

Phân rã tuple với dấu *

```
fruits = ("apple", "banana", "cherry“, "strawberry", "raspberry")
(green, yellow, *red) = fruits
print(green) # apple
print(yellow) # banana
print(red) #"cherry“, "strawberry", "raspberry"
```

Trong ví dụ trên xả fruits ra:

- **Apple** được gán cho green
- **Banane** được gán cho yellow
- Các phần tử còn lại trong Tuple được dồn hết cho **red với kí tự dấu ***

5.4 Các phương thức tương tác với Tuple

Phân rã một Tuple

Rã tuple với dấu *

```
fruits = ("apple", "banana", "cherry“, "strawberry", "raspberry")
(green, *yellow, red) = fruits
print(green) # apple
print(yellow) # "banana", "cherry“, "strawberry"
print(red) #"raspberry"
```

Trong ví dụ trên xả fruits ra:

- **Apple** được gán cho green
- **Raspberry** được gán cho red
- Các phần tử còn lại trong Tuple được dồn hết cho **yellow với kí tự dấu ***

5.4 Các phương thức tương tác với Tuple

Lặp các phần tử của Tuple với for

Dùng lệnh **for** đã học bài trước để lặp qua các phần tử của Tuple

```
mytuple = ("apple", "banana", "kiwi“)
for x in  mytuple :
    print(x)
```

Hoặc cú pháp ngắn (short hand)

```
[print(x) for x in  mytuple]
```

Lặp qua và in ra vị trí index của phần tử trong Tuple

```
for i in range(len(mytuple)):
    print(i)
```

5.4 Các phương thức tương tác với Tuple

Join Tuples

Hoặc bạn có thể sử dụng toán tử **+** để join 2 Tuple với nhau

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

Hoặc bạn có thể sử dụng toán tử nhân ***** để nhân bản và **join** lại

```
fruits = ("apple", "banana", "cherry")
mytuple = fruits * 2

print(mytuple)
```

5.5 Tổng quan về Sets

Khái niệm về Sets

- Trong Python, Set là một kiểu dữ liệu hỗn hợp dùng để lưu trữ tập hợp các phần tử duy nhất (distinct) không theo thứ tự. Set là một dạng của Collection trong Python và giống với tập hợp (set) trong toán học

- Một số điểm quan trọng về Set trong Python:

  1. Set không chứa các phần tử trùng lặp

  2. Set không duy trì thứ tự của các phần tử. Khi bạn in Set, thứ tự của các phần tử có thể thay đổi mỗi lần bạn thực hiện in

  3. Có thể chứa các phần tử không thay đổi như số nguyên, số thực, chuỗi, tuple trong Set. Nhưng Set không thể chứa các phần tử có thay đổi như danh sách (list) hoặc set khác

  4. Set không hỗ trợ truy cập phần tử bằng chỉ số như danh sách, vì vậy bạn không thể sử dụng Set[index]. Bạn phải sử dụng vòng lặp hoặc các phương thức có sẵn để truy cập và xử lý phần tử trong Set.

5.5 Tổng quan về Sets

Cách khai báo một Sets

```
thisset = {"apple", 3.14 , (1, 2), True, False, 30}
print(thisset)
```

- Set được lưu trữ trong một cặp ngoặc nhọn
- Các phần tử cách nhau bởi dấu phẩy ,
- Các phần tử có thể chứa nhiều loại dữ liệu int, float, bool, str, tuple, None

5.2 Các phương thức xử lý với Sets

Truy cập đến các phần tử của Sets

Bạn chỉ có thể duyệt qua các phần tử của sets bằng vòng lặp

```
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x)
```

Sử dụng từ khóa in để kiểm tra một phần tử có tồn tại trong set không

```
print("banana" in thisset)
```

5.5 Các phương thức xử lý với Sets

Thêm mới phần tử vào Sets

Với set bạn không thể thay đổi các phần tử sau khi khởi tạo, tuy nhiên có thể thêm mới

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

Hoặc bạn có thể sử dụng phương thức **update()** để thêm mới

```
tropical = {"pineapple", "mango", "papaya"}
thisset.update(tropical)
print(thisset)
```

Bạn có thể thêm một tuples, lists, dictionaries vào set với update()

5.5 Các phương thức xử lý với Sets

Xóa phần tử ra khỏi Sets

Xóa với phương thức **remove()**

```
thisset = {"apple", "banana", "cherry"}
thisset.remove(" banana ")
print(thisset)
```

Hoặc bạn có thể sử dụng phương thức **discard()**

```
thisset.discard(" banana ")
print(thisset)
```

5.5 Các phương thức xử lý với Sets

Xóa phần tử ra khỏi Sets

Bạn cũng có thể sử dụng phương thức pop() để xóa, tuy nhiên nó xóa một phần tử **ngẫu nhiên** trong set

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

Xóa tất cả phần tử trong set, trả lại một set rỗng

```
thisset.clear()
```

Xóa hoàn toàn một set

```
del thisset
```

5.2 Các phương thức xử lý với Sets

Gộp 2 Sets

Bạn có thể sử dụng union() hay update() để gộp 2 set với nhau

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

union() trả lại một set3 mới từ 2 set đã gộp

update() là gộp thêm set 2 vào set 1

5.5 Các phương thức xử lý với Sets

Một số phương thức khác

Phương thức	Ý nghĩa
intersection()	Trả về một Set mới chứa các phần tử chung của hai Set
difference()	Trả về một Set mới chứa các phần tử chỉ có trong Set đầu tiên nhưng không có trong Set thứ ha
issubset()	Kiểm tra xem Set có là tập con của một Set khác hay không
issuperset()	Kiểm tra xem Set có là tập cha của một Set khác hay không

Set là một công cụ hữu ích trong Python để làm việc với các tập hợp duy nhất của dữ liệu mà không cần quan tâm đến thứ tự của chúng

5.6 Các phương thức tương tác với Tuple

Nguồn tham chiếu

- Tổng quan về List

  https://www.w3schools.com/python/python_lists.asp

- Các phương thức thao tác với List

  https://www.w3schools.com/python/python_lists_methods.asp

- Tổng quan về Tuple

  https://www.w3schools.com/python/python_tuples.asp

- Các phương thức thao tác với List

  https://www.w3schools.com/python/python_ref_tuple.asp

Tổng kết lại bài 5

1. Nắm được cách định nghĩa kiểu dữ liệu mảng với List
2. Nắm được các phương thức xử lý dữ liệu với List
3. Nắm được cách định nghĩa kiểu dữ liệu mảng với Tuple
4. Nắm được các phương thức xử lý dữ liệu với Tuple
5. Nắm được cách định nghĩa kiểu dữ liệu mảng với Set
6. Nắm được các phương thức xử lý dữ liệu với Set