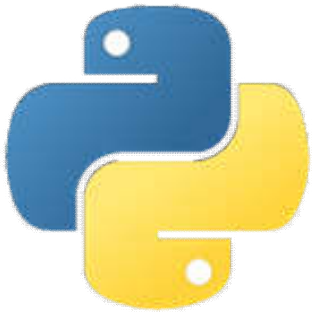


List và Tuple

Nội dung

1. **Nhắc lại bài cũ**
2. **List**
3. **Tuple**
4. **Q & A**

- Các loại toán tử: *Toán tử số học, toán tử so sánh, toán tử gán, toán tử logic…*
- Nhập, xuất dữ liệu: *input/ print*
- Cấu trúc điều khiển luồng dữ liệu (rẽ nhánh): *If/elif/else*
- Cấu trúc lặp: *For/While*
- Kiểu dữ liệu: *number, string*

Nội dung

1. **Nhắc lại bài cũ**
2. **List**
3. **Tuple**
4. **Q & A**

1.1 List (danh sách)

- List là kiểu dữ liệu linh hoạt nhất trong Python.
- Nó là một dẫy (sequence) bao gồm nhiều phần tử (element).
- List cho phép loại bỏ, hoặc thêm các phần tử vào danh sách, đồng thời cho phép cắt lát các phần tử.
- List trong Python là *kiểu dữ liệu thay đổi* (mutable), nghĩa là Python sẽ không tạo một List mới lập trình viên thực hiện sửa đổi một hay nhiều phần tử trong List đó.

❖List được khai báo bằng cách đặt các phần tử nằm trong cặp ngoặc vuông [ ] và ngăn cách nhau bởi dấu phẩy.

❖Các phần tử trong danh sách tự động được đánh chỉ số (index) từ trái sang phải bắt đầu từ chỉ số 0.

```
fruitList = ["apple", "apricot", "banana","coconut", "lemen"]
otherList = [100, "one", "two", 3]

print ("Fruit List:")
print (fruitList)
print (" ------------------------- ")
print ("Other List:")
print (otherList)
```

1.2 Truy cập vào phần tử

❖Sử dụng vòng lặp for để truy cập vào tất cả các phần tử của danh sách:

```
fruitList = ["apple", "apricot", "banana", "coconut", "plum", "pear"]

for fruit in fruitList:
    print("Fruit: ", fruit)
```

- Lập trình viên cũng có thể truy cập vào mỗi một phần tử của list thông qua chỉ số (index). Các phần tử của list được đánh chỉ số từ trái sang phải, bắt đầu từ 0.
- Cú pháp:
  - Truy cập vào 1 phần tử cụ thể:

```
list[i]
```

  - Tạo ra sublist từ list ban đầu đã cho. Sublist này được tạo từ các phần tử từ i đến j của list ban đầu:

```
list[i:j]
```

```
fruitList = ["apple", "apricot", "banana", "coconut", "plum", "pear"]
print(fruitList)

# Số phần tử
print("Element count: ", len(fruitList))

for i in range(0, len(fruitList)):
    print("Element at ", i, "= ", fruitList[i])

# Một danh sách con chứa các phần tử từ index 1 đến 4 (1, 2, 3)
subList = fruitList[1: 4]

# ['apricot', 'banana', 'coconut']
print("Sub List [1:4] ", subList)
```

❖Có thể truy cập vào các phần tử của danh sách theo chỉ số âm (Negative index), các phần tử được đánh chỉ số từ phải sang trái với các giá trị -1, -2, ...

```
fruitList = ["apple", "apricot", "banana", "coconut", "plum", "pear"]
print(fruitList)

print("Element count: ", len(fruitList))
print("fruitList[-1]: ", fruitList[-1])
print("fruitList[-2]: ", fruitList[-2])

subList1 = fruitList[-4:]

print("\n")
print("Sub List fruitList[-4: ] ")
print(subList1)

subList2 = fruitList[-4:-2]

print("\n")
print("Sub List fruitList[-4:-2] ")
print(subList2)
```

1.3 Cập nhật danh sách

❖ Cập nhật danh sách theo chỉ số (index)

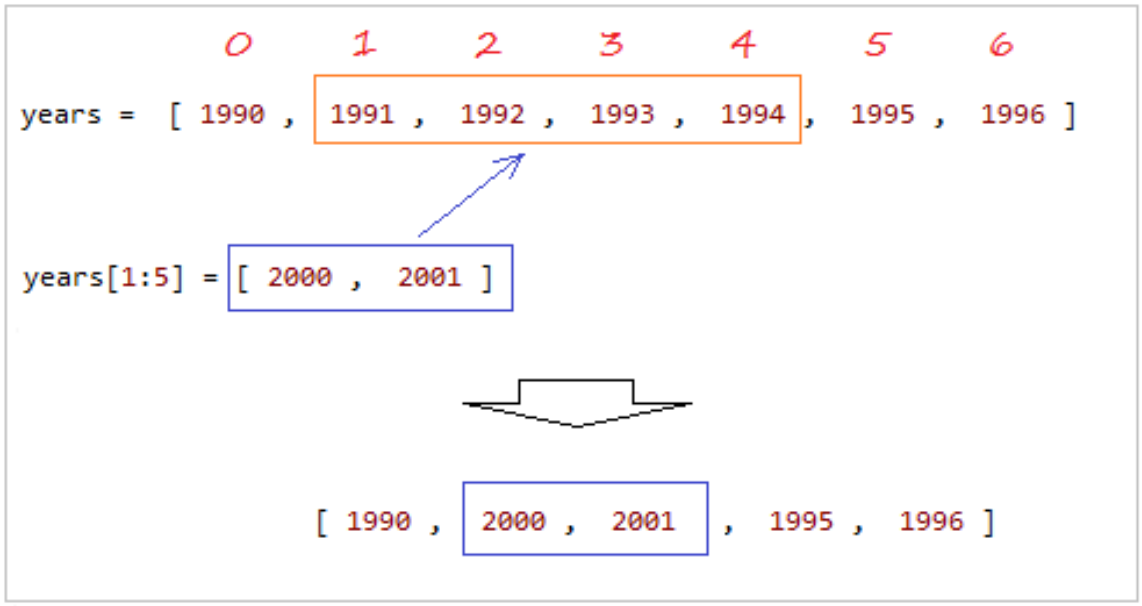
```
years = [1991, 1995, 1992]

print("Years khi chưa cập nhật: ", years)

# Cập nhật danh sách tại vị trí 1
years[1] = 2000

#In ra danh sách sau cập nhật
print("Years sau khi cập nhật: ", years)
```

❖Lập trình viên cũng có thể cập nhập giá trị cho một lát (Slice) các phần tử. Đây là cách để cập nhập nhiều phần tử một lúc.



```
years = [1990, 1991, 1992, 1993, 1994, 1995, 1996]

print("Years: ", years)

print("Cập nhật Slice: years[1:5] = [2000, 2001]")

years[1:5] = [2000, 2001]

print("Years: ", years)
```

1.4 Thêm phần tử vào danh sách

❖Dùng câu lệnh append(new_element) để nối thêm phần tử vào cuối danh sách.

```
years = [1991, 1995, 1992]

print("Append 2015, 2016 to list:")

# Nối (append) thêm một phần tử vào cuối danh sách.
years.append(2015)
years.append(2016)

print("Years: ", years)
```

❖Câu lệnh insert dùng để thêm phần tử vào vị trí bất kỳ của danh sách.

❖Cú pháp: *list.insert (i, new_element).* Trong đó: i là vị trí cần thêm vào, new_element là phần tử mới.

```
years = [1990, 1991, 1992, 1993, 1993, 1993, 1994]

years.insert(1, 9999)

print("Years: ", years)
```

1.5 Xóa phần tử trong danh sách

- Để xóa một hoặc nhiều phần tử trong một danh sách (list) có thể sử dụng một trong các phương pháp:
  - Dùng lệnh del (del statement): để xóa một hoặc nhiều phần tử theo chỉ số.
  - Hoặc sử dụng phương thức remove()

❖ Dùng lệnh del(del statement):

```
years = [1990, 1991, 1992, 1993, 1994, 1995, 1996]

print("Years: ", years)

print("\n del years[6]")

# Xóa phần tử tại vị trí có index = 6.
del years[6]

print("Years: ", years)
print("\n del years[1:4]")

# Xóa nhiều phần tử tại index = 1,2,3
del years[1:4]
print("Years: ", years)
```

❖Dùng remove(value)

```
years = [1990, 1991, 1992, 1993, 1994, 1993, 1993]

print("Years: ", years)

print("\n years.remove(1993)")

# Loại bỏ phần tử đầu tiên có giá trị 1993 trong danh sách
years.remove(1993)

print("Years: ", years)
```

1.6 Toán tử trong danh sách

Toán tử	Mô tả	Ví dụ
+	Toán tử sử dụng để nối (concatenate) 2 List để tạo ra một List mới.	[1, 2, 3] + ["One","Two"] --> [1, 2, 3, "One", "Two"]
*****	Toán tử sử dụng để nối (concatenate) nhiều bản copy của cùng một List. Và tạo ra một List mới.	[1, 2] * 3 --> [1, 2, 1, 2, 1, 2]
in	Kiểm tra một phần tử có nằm List hay không, kết quả trả về True hoặc False.	"Abc" in ["One", "Abc"] --> True

S

```
list1 = [1, 2, 3]

list2 = ["One", "Two"]

print("list1: ", list1)
print("list2: ", list2)
print("\n")

list12 = list1 + list2

print("list1 + list2: ", list12)

list2x3 = list2 * 3

print("list2 * 3: ", list2x3)

hasThree = "Three" in list2

print("'Three' in list2? ", hasThree)
```

1.7 Hàm sử dụng với list

Hàm	Mô tả
len(list)	Trả về số phần tử của danh sách
max(list)	Trả về phần tử trong danh sách với giá trị lớn nhất.
min(list)	Trả về phần tử trong danh sách với giá trị nhỏ nhất.
list(seq)	Chuyển đổi một tuple thành danh sách.

```
list1 = [1991, 1994, 1992]
list2 = [1991, 1994, 2000, 1992]

print("list1: ", list1)
print("list2: ", list2)

# Trả về số lượng phần tử của danh sách.
print("len(list1): ", len(list1))
print("len(list2): ", len(list2))

# Giá trị lớn nhất trong danh sách (list).
maxValue = max(list1)

print("Max value of list1: ", maxValue)

# Giá trị nhỏ nhất trong danh sách (list).
minValue = min(list1)

print("Min value of list1: ", minValue)
```

1.8 Phương thức sử dụng với list

Phương thức	**Mô tả**
list.append(obj)	Nối (append) một đối tượng vào danh sách.
list.count(obj)	Trả về số lần xuất hiện 'obj' trong danh sách
list.extend(seq)	Nối nội dung của một dẫy (sequence) và danh sách
list.index(obj)	Trả về chỉ số nhỏ nhất trong danh sách tìm thấy phần tử có giá trị là **obj**.
list.insert(index, obj)	Trèn một phần tử vào danh sách tại chỉ số ***index***.
list.pop([index])	Nếu có tham số index, loại bỏ và trả về phần tử tại vị trí index. Ngược lại loại bỏ và trả về phần tử cuối cùng của danh sách.
list.remove(obj)	Loại bỏ phần tử đầu tiên có giá trị là **obj** ra khỏi danh sách.
list.reverse()	Đảo ngược các phần tử trong danh sách
list.sort(key=None, reverse=False)	Sắp xếp các phần tử theo khóa (key) cung cấp bởi tham số.

```
years = [1990, 1991, 1992, 1993, 1993, 1993, 1994]
print("Years: ", years)

# Đảo ngược danh sách.
years.reverse()
print("Years (After reverse): ", years)
aTuple = (2001, 2002, 2003)
print("\n - Extend: ", aTuple)
years.extend(aTuple)
print("Years (After extends): ", years)
print("\n - Append 3000")
years.append(3000)
print("Years (After appends): ", years)
print("\n - Remove 1993")
years.remove(1993)
print("Years (After remove): ", years)
print("\n - years.pop()")

# Loại bỏ phần tử cuối cùng của danh sách.
lastElement = years.pop()
print("last element: ", lastElement)
print("\n")
# Count
print("years.count(1993): ", years.count(1993))
```

1.9 Luyện tập

- **Bài 1**: Viết chương trình python tính tổng của các phần tử trong một list sau: _list= [1, 2, 3, 4, 5, 6, 7, 8, 9].
- **Bài 2**: Viết chương trình python tính tích của các phần tử trong một list sau: _list= [1, 2, 3, 4, 5].
- **Bài 3**: Cho đầu vào là _list= [1, 2, 3, 4, 5, 6, 7, 8, 9]. Viết chương trình python tạo 2 list mới từ list đó cho. Trong đó 1 list chỉ bao gồm số chẵn (even), 1 list chỉ bao gồm số lẻ (odd).
- **Bài 4**: Cho danh sách ban đầu _list = ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow']. Tạo danh sách mới chứa 2 phần tử ở vị trí thứ 2 và thứ 3 _new = ['White', 'Black']
- **Bài 5**: Cho danh sách đầu vào _list = [‘zero’, ‘three’]. Tạo ra danh sách mới _new = [‘zero’, ‘one’, ‘two’, ‘three’] bằng cách thêm các phần tử vào danh sách ban đầu.

❖**Bài 6**: Viết một chương trình Python để đếm số chuỗi thỏa mãn điều kiện: ~~độ dài chuỗi lớn hơn hoặc bằng một giá trị đầu vào,~~ ký tự đầu tiên và cuối cùng là giống nhau từ một danh sách cho trước. Ví dụ cho list đầu vào : ['abc', 'xyz', 'aba', '1221', ‘ii’, ‘ii2’, ‘5yhy5’] → Kết quả cho ra ~~3.~~ 4

❖**Bài 7**: Viết chương trình tạo một list mới bằng cách loại bỏ các phần tử có giá trị giống nhau trong list đầu vào. Ví dụ: nhập list đầu vào là _list= ['abc', 'xyz', 'abc', '12', ‘ii’, ‘12’, ‘5a’] → kết quả list mới là _new= ['xyz', ‘ii’, ‘5a’]

❖**Bài 8**: Viết chương trình lấy ra số lớn nhất trong list: _list= [11, 2, 23, 45, 6, 9].

❖**Bài 9**: Viết chương trình lấy ra số nhỏ nhất trong list: _list= [11, 2, 23, 45, 6, 9]

❖**Bài 10**: Viết chương trình copy một list cho trước thành một list mới.

❖**Bài 11**: Nhập vào từ bàn phím số n và list cho trước, tìm các từ có độ dài lớn hơn n từ list đó.

Nội dung

1. **Nhắc lại bài cũ**
2. **List**
3. **Tuple**
4. **Q & A**

2.1 Tuple

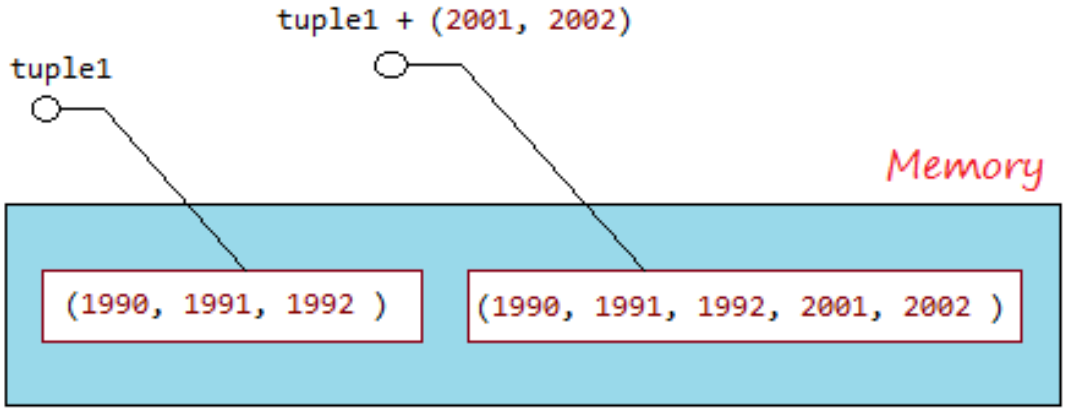
- Tuple (bộ dữ liệu) là một dẫy (sequence) các giá trị, nó chứa nhiều phần tử (element), về cơ bản nó khá giống với List (danh sách).
- Nhưng khác với List, Tuples là một kiểu dữ liệu bất biến (immutable), mọi thao tác cập nhập trên Tuple đều tạo ra một thực thể mới trên bộ nhớ (memory).
- Để viết một Tuple, bạn đặt các phần tử nằm trong cặp ngoặc ( ) và ngăn cách nhau bởi dấu phẩy. Các phần tử trong danh sách được đánh chỉ số (index) bắt đầu từ chỉ số 0.

```
fruitTuple = ("apple", "apricot", "banana", "coconut", "lemen")

otherTuple = (100, "one", "two", 3)
```

❖Tuple là một đối tượng bất biến (immutable), nó không có các phương thức append(), remove(),.. như list.

❖Một số phương thức, hoặc toán tử dùng để cập nhập Tuple sẽ dựa trên Tuple ban đầu để tạo ra một Tuple mới.



1.2 Thêm/Sửa/Xóa Tuple

- Toán tử với tuple: +; *; in
- Thêm: đổi tuple thành list và sử dụng như 1 list hoặc sử dụng các subtuple và toán tử +
- Sửa: tương tự list
- Xóa: chỉ có thể dùng lệnh del mà không dùng được remove như list.

1.2 Phương thức

Phương thức	**Mô tả**
tuple.count(obj)	Trả về số lần xuất hiện 'obj' trong Tuple
tuple.index(obj, [start, [stop]])	Trả về chỉ số nhỏ nhất trong danh sách tìm thấy phần tử có giá trị là **obj**. Ném ra **ValueError** nếu không tìm thấy. Nếu có tham số **start**, **stop**, chỉ tìm từ chỉ số **start** đến chỉ số **stop** (Và không bao gồm **stop**).

```
years = (1990, 1991, 1993, 1991, 1993, 1993, 1993)

print("Years: ", years)

print("\n")

# Trả về số lần xuất hiện của 1993
print("years.count(1993): ", years.count(1993))

# Tìm vị trí xuất hiện 1993
print("years.index(1993): ", years.index(1993))

# Tìm vị trí xuất hiện 1993, bắt đầu từ chỉ số 3
print("years.index(1993, 3): ", years.index(1993, 3))

# Tìm vị trí xuất hiện 1993, bắt đầu từ chỉ số 4 tới 5 (Không bao gồm 6)
print("years.index(1993, 4, 6): ", years.index(1993, 4, 6))
```

❖Tại sao chúng ta sử dụng tuple?

- ❑Trình xử lý các tuple là nhanh hơn các List.
- ❑Làm cho dữ liệu an toàn hơn bởi vì tuple là không thay đổi (immutable) và vì thế nó không thể bị thay đổi.

❖**Bài 1**: Cho tuple: _tuple = (‘a’, ‘b’, ‘d’, ‘e’). Hãy thêm phần tử ‘c’ vào index 2 của _tuple trên để tạo ra _new_tuple = (‘a’, ‘b’, ’c’, ‘d’, ‘e’).

❖**Bài 2**: Viết chương trình loại bỏ các phần tử có giá trị giống nhau trong một tuple, để tạo 1 tuple mới. Ví dụ: nhập vào _tuple = (‘ab’, ‘b’, ‘e’, ‘c’, ‘d’, ‘e’, ‘ab’), thì thu được _new_tuple = (‘b’, ‘c’, ‘d’). Gợi ý: dùng hàm count.


Q
&
A