NỀN TẢNG CÔNG NGHỆ THÔNG TIN 2

TUPLE - LIST - SET - DICT

LaTeX bởi Ngô Hoàng Tùng

I. Mục tiêu bài học

1. Tuple

a. Hiểu được tuple là gì, cách khai báo và sử dụng tuple trong Python.

b. Biết cách truy cập các phần tử trong tuple và các phương thức cơ bản của tuple.

2. List

a. Hiểu được list là gì, cách khai báo và sử dụng list trong Python.

b. Biết cách truy cập các phần tử trong list và các phương thức cơ bản của list.

3. Set

a. Hiểu được set là gì, cách khai báo và sử dụng set trong Python.

b. Biết cách truy cập các phần tử trong set và các phương thức cơ bản của set.

4. Dict

a. Hiểu được dict là gì, cách khai báo và sử dụng dict trong Python.

b. Biết cách truy cập các phần tử trong dict và các phương thức cơ bản của dict.

5. So sánh tuple, list, set và dict

a. Hiểu được sự khác biệt giữa tuple, list, set và dict trong Python.

b. Biết cách sử dụng từng loại dữ liệu phù hợp với từng tình huống cụ thể.

c. Biết cách chuyển đổi giữa các loại dữ liệu này.

II. Nội dung bài học

1. Tuple

a. Hiểu được tuple là gì, cách khai báo và sử dụng tuple trong Python.
– Khái niệm: Tuple là một kiểu dữ liệu trong Python dùng để lưu trữ một tập hợp các giá trị không thay đổi (immutable).
– Cách khai báo: Tuple được khai báo bằng cách đặt các giá trị trong dấu ngoặc đơn '()', các giá trị được phân tách bằng dấu phẩy ','.

```
#  Declare a tuple
my_tuple = (1, 2, 3, 'Python', True)
print(my_tuple)  # In ra: (1, 2, 3, 'Python', True)
# Accessing an element in a tuple
print(my_tuple[0])  # In ra: 1
```

```
print(my_tuple[3])  # In ra: 'Python'
```

b. Biết cách truy cập các phần tử trong tuple và các phương thức cơ bản của tuple.
– Truy cập phần tử: Sử dụng chỉ số (index) để truy cập phần tử trong tuple, chỉ số bắt đầu từ 0.
– Các phương thức cơ bản:
+ 'count()': Đếm số lần xuất hiện của một giá trị trong tuple.
+ 'index()': Trả về chỉ số của lần xuất hiện đầu tiên của một giá trị trong tuple.

```
# Accessing an element in a tuple
print(my_tuple[1])  # In ra: 2
print(my_tuple[-1])  # In ra: True.(access denied)
# Use basic methods
print(my_tuple.count(2))  # In ra: 1 (count of 2)
print(my_tuple.index('Python'))  # In ra: 3 (index of 'Python')
```

2. List

a. Hiểu được list là gì, cách khai báo và sử dụng list trong Python.
– Khái niệm: List là một kiểu dữ liệu trong Python dùng để lưu trữ một tập hợp các giá trị có thể thay đổi (mutable).
– Cách khai báo: List được khai báo bằng cách đặt các giá trị trong dấu ngoặc vuông '[]', các giá trị được phân tách bằng dấu phẩy ','.

```
# Declare a list
my_list = [1, 2, 3, 'Python', True]
print(my_list)  # In ra: [1, 2, 3, 'Python', True]
# Accessing an element in a list
print(my_list[0])  # In ra: 1
print(my_list[3])  # In ra: 'Python'
```

b. Biết cách truy cập các phần tử trong list và các phương thức cơ bản của list.
– Truy cập phần tử: Sử dụng chỉ số (index) để truy cập phần tử trong list, chỉ số bắt đầu từ 0.
– Các phương thức cơ bản:
+ 'append()': Thêm một phần tử vào cuối list.
+ 'remove()': Xóa một phần tử khỏi list.
+ 'sort()': Sắp xếp các phần tử trong list.

```
# Accessing an element in a list
print(my_list[1])  # In ra: 2
print(my_list[-1])  # In ra: True.(access denied)
# Use basic methods
my_list.append('New Element')  # Add a new element to the end of the list.
print(my_list)  # In ra: [1, 2, 3, 'Python', True, 'New Element']
my_list.remove(2)  # Remove element 2 from the list
print(my_list)  # In ra: [1, 3, 'Python', True, 'New Element']
my_list.sort()  # Sort the elements in the list
print(my_list)  # In ra: [1, 3, 'New Element', 'Python', True]
```

3. Set

a. Hiểu được set là gì, cách khai báo và sử dụng set trong Python.
– Khái niệm: Set là một kiểu dữ liệu trong Python dùng để lưu trữ một tập hợp các giá trị duy nhất (không trùng lặp).
– Cách khai báo: Set được khai báo bằng cách đặt các giá trị trong dấu ngoặc nhọn '', các giá trị được phân tách bằng dấu phẩy ','.

```
# Declare a set
my_set = {1, 2, 3, 'Python', True}
print(my_set)  # In ra: {1, 2, 3, 'Python', True}
# Accessing an element in a set
# (Sets do not support indexing, so we cannot access elements by index)
# However, we can check if an element exists in a set
print(2 in my_set)  # In ra: True (2 exists in the set)
print('Python' in my_set)  # In ra: True ('Python' exists in the set)
```

b. Biết cách truy cập các phần tử trong set và các phương thức cơ bản của set.
– Truy cập phần tử: Set không hỗ trợ truy cập theo chỉ số, nhưng có thể kiểm tra sự tồn tại của một phần tử trong set.
– Các phương thức cơ bản:
+ 'add()': Thêm một phần tử vào set.
+ 'remove()': Xóa một phần tử khỏi set.
+ 'union()': Tạo một tập hợp mới từ hai tập hợp.

```
# Accessing an element in a set
# (Sets do not support indexing, so we cannot access elements by index)
# However, we can check if an element exists in a set
print(2 in my_set)  # In ra: True (2 exists in the set)
print('Python' in my_set)  # In ra: True ('Python' exists in the set)
# Use basic methods
my_set.add('New Element')  # Add a new element to the set
print(my_set)  # In ra: {1, 2, 3, 'Python', True, 'New Element'}
my_set.remove(2)  # Remove element 2 from the set
print(my_set)  # In ra: {1, 3, 'Python', True, 'New Element'}
my_set2 = {3, 4, 5}  # Declare another set
my_union_set = my_set.union(my_set2)  # Create a new set from the union of two sets
print(my_union_set)  # In ra: {1, 3, 4, 5, 'Python', True, 'New Element'}
```

4. Dict

a. Hiểu được dict là gì, cách khai báo và sử dụng dict trong Python.
– Khái niệm: Dict (dictionary) là một kiểu dữ liệu trong Python dùng để lưu trữ các cặp khóa-giá trị (key-value pairs).
– Cách khai báo: Dict được khai báo bằng cách đặt các cặp khóa-giá trị trong dấu ngoặc nhọn '', các cặp được phân tách bằng dấu phẩy ','.

```
# Declare a dict
my_dict = {'name': 'Python', 'version': 3.10, 'is_active': True}
print(my_dict)  # In ra: {'name': 'Python', 'version': 3.10, 'is_active': True}
# Accessing a value in a dict
print(my_dict['name'])  # In ra: 'Python'
print(my_dict['version'])  # In ra: 3.10
```

b. Biết cách truy cập các phần tử trong dict và các phương thức cơ bản của dict.
– Truy cập phần tử: Sử dụng khóa (key) để truy cập giá trị tương ứng trong dict.
– Các phương thức cơ bản:
+ 'keys()': Trả về danh sách các khóa trong dict.
+ 'values()': Trả về danh sách các giá trị trong dict.
+ 'items()': Trả về danh sách các cặp khóa-giá trị trong dict.

```
# Accessing a value in a dict
print(my_dict['name'])  # In ra: 'Python'
print(my_dict['version'])  # In ra: 3.10
# Use basic methods
print(my_dict.keys())  # In ra: dict_keys(['name', 'version', 'is_active'])
print(my_dict.values())  # In ra: dict_values(['Python', 3.10, True])
print(my_dict.items())  # In ra: dict_items([('name', 'Python'), ('version', 3.10),
    ('is_active', True)])
# Adding a new key-value pair to the dict
my_dict['author'] = 'Guido van Rossum'  # Add a new key-value pair
print(my_dict)  # In ra: {'name': 'Python', 'version': 3.10, 'is_active': True, 'author':
    'Guido van Rossum'}
# Removing a key-value pair from the dict
my_dict.pop('is_active')  # Remove the key 'is_active'
print(my_dict)  # In ra: {'name': 'Python', 'version': 3.10, 'author': 'Guido
van Rossum'}
# Updating a value in the dict
my_dict['version'] = 3.11  # Update the value of the key 'version'
print(my_dict)  # In ra: {'name': 'Python', 'version': 3.11, 'author': 'Guido
van Rossum'}
```

5. So sánh tuple, list, set và dict

a. Hiểu được sự khác biệt giữa tuple, list, set và dict trong Python.
– Tuple: Là kiểu dữ liệu không thay đổi (immutable), có thể chứa các giá trị khác nhau, truy cập bằng chỉ số.
– List: Là kiểu dữ liệu có thể thay đổi (mutable), có thể chứa các giá trị khác nhau, truy cập bằng chỉ số.
– Set: Là kiểu dữ liệu không có thứ tự (unordered), chỉ chứa các giá trị duy nhất, không hỗ trợ truy cập theo chỉ số.
– Dict: Là kiểu dữ liệu lưu trữ cặp khóa-giá trị, truy cập bằng khóa.

b. Biết cách sử dụng từng loại dữ liệu phù hợp với từng tình huống cụ thể.
– Sử dụng tuple khi cần lưu trữ một tập hợp các giá trị không thay đổi.
– Sử dụng list khi cần lưu trữ một tập hợp các giá trị có thể thay đổi.
– Sử dụng set khi cần lưu trữ một tập hợp các giá trị duy nhất và không quan tâm đến thứ tự.
– Sử dụng dict khi cần lưu trữ các cặp khóa-giá trị.

c. Biết cách chuyển đổi giữa các loại dữ liệu này.

```
# Converting between different data types
# Convert tuple to list
my_tuple = (1, 2, 3)
my_list = list(my_tuple)
print(my_list)  # In ra: [1, 2, 3]
# Convert list to tuple
my_list = [4, 5, 6]
my_tuple = tuple(my_list)
print(my_tuple)  # In ra: (4, 5, 6)
# Convert set to list
my_set = {7, 8, 9}
my_list = list(my_set)
```

```
print(my_list)  # In ra: [7, 8, 9]
# Convert list to set
my_list = [10, 11, 12]
my_set = set(my_list)
print(my_set)  # In ra: {10, 11, 12}
# Convert dict keys to list
my_dict = {'a': 1, 'b': 2, 'c': 3}
my_keys = list(my_dict.keys())
print(my_keys)  # In ra: ['a', 'b', 'c']
# Convert dict values to list
my_values = list(my_dict.values())
print(my_values)  # In ra: [1, 2, 3]
# Convert dict items to list of tuples
my_items = list(my_dict.items())
print(my_items)  # In ra: [('a', 1), ('b', 2), ('c', 3)]
# Convert list to dict (using index as key)
my_list = ['a', 'b', 'c']
my_dict = {i: my_list[i] for i in range(len(my_list))}
print(my_dict)  # In ra: {0: 'a', 1: 'b', 2: 'c'}
# Convert dict to list of keys
my_dict = {'x': 10, 'y': 20, 'z': 30}
my_keys = list(my_dict.keys())
print(my_keys)  # In ra: ['x', 'y', 'z']
# Convert dict to list of values
my_values = list(my_dict.values())
print(my_values)  # In ra: [10, 20, 30]
# Convert dict to list of items (key-value pairs)
my_items = list(my_dict.items())
print(my_items)  # In ra: [('x', 10), ('y', 20), ('z', 30)]
```

III. Bài tập

IV. chia sẻ kinh nghiệm

– Trong quá trình học tập và làm việc với các kiểu dữ liệu tuple, list, set và dict trong Python, có một số kinh nghiệm hữu ích mà bạn có thể áp dụng:

+ Sử dụng tuple khi bạn cần lưu trữ một tập hợp các giá trị không thay đổi, ví dụ như các hằng số hoặc các giá trị cấu hình. Tuple có hiệu suất truy cập nhanh hơn so với list do tính chất không thay đổi của nó.

+ Sử dụng list khi bạn cần lưu trữ một tập hợp các giá trị có thể thay đổi, ví dụ như danh sách các mục trong một ứng dụng. List cung cấp nhiều phương thức hữu ích để thao tác với các phần tử, như thêm, xóa, sắp xếp.

+ Sử dụng set khi bạn cần lưu trữ một tập hợp các giá trị duy nhất và không quan tâm đến thứ tự, ví dụ như danh sách các từ khóa trong một ngôn ngữ lập trình. Set cung cấp các phương thức để kiểm tra sự tồn tại của phần tử và thực hiện các phép toán tập hợp.

+ Sử dụng dict khi bạn cần lưu trữ các cặp khóa-giá trị, ví dụ như cấu hình của một ứng dụng hoặc thông tin người dùng. Dict cho phép truy cập nhanh đến giá trị dựa trên khóa và cung cấp các phương thức để thao tác với các cặp khóa-giá trị.

+ Khi làm việc với các kiểu dữ liệu này, hãy chú ý đến hiệu suất và tính chất của từng loại dữ liệu để chọn lựa phù hợp với nhu cầu của bạn. Ví dụ, nếu bạn cần truy cập nhanh đến các phần tử theo chỉ số, hãy sử dụng list hoặc tuple. Nếu bạn cần lưu trữ các giá trị duy nhất, hãy sử dụng set. Nếu bạn cần lưu trữ các cặp khóa-giá trị, hãy sử dụng dict.

+ Ngoài ra, hãy chú ý đến việc chuyển đổi giữa các kiểu dữ liệu này khi cần thiết. Python cung cấp các hàm tích hợp để chuyển đổi giữa tuple, list, set và dict một cách dễ dàng. Việc này giúp bạn linh hoạt trong việc xử lý dữ liệu và tối ưu hóa hiệu suất của chương trình.
+ Cuối cùng, hãy luôn kiểm tra và xử lý các trường hợp ngoại lệ khi làm việc với các kiểu dữ liệu này, ví dụ như truy cập phần tử không tồn tại trong list hoặc dict, hoặc thêm phần tử trùng lặp vào set. Việc này giúp đảm bảo chương trình của bạn hoạt động ổn định và tránh lỗi không mong muốn.

HẾT