Python Programming - Tuples (Bộ)

> **Mục tiêu học tập**: Làm chủ Tuples - cấu trúc dữ liệu immutable để lưu trữ collections không thay đổi và return multiple values.

1. Giới Thiệu về Tuples

1.1. Tuple là gì?

🤔 Đặt vấn đề

Đôi khi bạn cần lưu dữ liệu **không bao giờ thay đổi** (coordinates, RGB colors, dates).

```
# ❌ Dùng list - có thể bị sửa nhầm
coordinates = [10.5, 20.3]
coordinates[0] = 999  # Oops! Sửa nhầm
print(coordinates)  # [999, 20.3] - Dữ liệu sai!

# Hoặc dùng constants?
COORDINATE_X = 10.5
COORDINATE_Y = 20.3
# → Phải tạo nhiều biến, khó quản lý
```

Làm sao để **bảo vệ** dữ liệu khỏi bị thay đổi vô tình?

💡 Giải quyết: Tuples

Tuple là sequence **immutable** (không thay đổi được).

```
# ☑ Dùng tuple - không thể sửa
coordinates = (10.5, 20.3)

try:
    coordinates[0] = 999
except TypeError as e:
    print(e)  # 'tuple' object does not support item assignment

print(coordinates)  # (10.5, 20.3) - An toàn!
```

Đặc điểm của Tuple:

- ☑ **Ordered** (có thứ tự) - giữ thứ tự insertion
- ☑ **Immutable** (không đổi) - không thể thay đổi sau khi tạo
- ☑ **Allow duplicates** - cho phép giá trị trùng
- ☑ **Heterogeneous** - chứa nhiều kiểu dữ liệu

- ☑ **Indexable** - truy cập bằng index
- ☑ **Hashable** - dùng làm dict key (nếu elements hashable)

1.2. Khi Nào Dùng Tuple?

Dùng Tuple khi:

- Dữ liệu **không nên thay đổi** (coordinates, dates, config)
- Return **nhiều giá trị** từ function
- Dùng làm **dict keys** (list không được)
- **Unpacking** values
- Cần **performance** tốt hơn list (nhẹ hơn)

Dùng List khi:

- Dữ liệu **thay đổi thường xuyên**
- Cần **thêm/xóa** elements
- Không quan tâm immutability

1.3. Tuple vs List

Tiêu chí	Tuple	List
Mutability	❌ Immutable	☑ Mutable
Syntax	`(1, 2, 3)`	`[1, 2, 3]`
Performance	Nhanh hơn	Chậm hơn
Memory	Ít hơn	Nhiều hơn
Methods	2 (.count, .index)	11+ (.append, .remove, ...)
Dict key	☑ Được	❌ Không
Use case	Fixed data	Dynamic data

```
# List - mutable
my_list = [1, 2, 3]
my_list[0] = 10  # OK
my_list.append(4)  # OK

# Tuple - immutable
my_tuple = (1, 2, 3)
# my_tuple[0] = 10   # ❌ TypeError
# my_tuple.append(4) # ❌ AttributeError
```

---

2. Tạo Tuples

2.1. Tuple Literals

```
# Tuple với parentheses
numbers = (1, 2, 3, 4, 5)
print(numbers)  # (1, 2, 3, 4, 5)

# Tuple hỗn hợp
mixed = (1, "two", 3.0, True)
print(mixed)  # (1, 'two', 3.0, True)

# Tuple rỗng
empty = ()
print(empty)  # ()
print(type(empty))  # <class 'tuple'>

# ⚠ Tuple 1 phần tử - CẦN DẤU PHẨY
single = (5,)  # ☑ Tuple
print(type(single))  # <class 'tuple'>

not_tuple = (5)  # ❌ Chỉ là int trong ngoặc
print(type(not_tuple))  # <class 'int'>

# Tuple lồng nhau
nested = ((1, 2), (3, 4), (5, 6))
print(nested)  # ((1, 2), (3, 4), (5, 6))
```

2.2. Tuple Packing (Không Cần Ngoặc)

```
# Tuple packing - tự động tạo tuple
coordinates = 10.5, 20.3
print(coordinates)  # (10.5, 20.3)
print(type(coordinates))  # <class 'tuple'>

# Nhiều phần tử
person = "An", 25, "Hà Nội"
print(person)  # ('An', 25, 'Hà Nội')

# Dùng ngoặc rõ ràng hơn (khuyến nghị)
person = ("An", 25, "Hà Nội")
```

2.3. Hàm `tuple()`

```
# Từ list
numbers_list = [1, 2, 3, 4, 5]
numbers_tuple = tuple(numbers_list)
print(numbers_tuple)  # (1, 2, 3, 4, 5)

# Từ string
chars = tuple("hello")
print(chars)  # ('h', 'e', 'l', 'l', 'o')
```

```
# Từ range
numbers = tuple(range(5))
print(numbers)  # (0, 1, 2, 3, 4)

# Từ set
unique = tuple({3, 1, 2})
print(unique)  # (1, 2, 3) - thứ tự không đảm bảo
```

2.4. Tuple Comprehension? (KHÔNG TỒN TẠI!)

```
# ❌ Không có tuple comprehension
result = (x**2 for x in range(5))
print(type(result))  # <class 'generator'> - KHÔNG phải tuple!

# ☑ Phải dùng tuple() + generator
squared = tuple(x**2 for x in range(5))
print(squared)  # (0, 1, 4, 9, 16)

# Hoặc list comprehension + tuple()
squared = tuple([x**2 for x in range(5)])
print(squared)  # (0, 1, 4, 9, 16)
```

---

3. Truy Cập Elements

3.1. Indexing (Giống List)

```
fruits = ("apple", "banana", "cherry", "date")

# Positive indexing (từ trái)
print(fruits[0])   # apple
print(fruits[1])   # banana
print(fruits[3])   # date

# Negative indexing (từ phải)
print(fruits[-1])  # date
print(fruits[-2])  # cherry

# IndexError nếu index ngoài phạm vi
try:
    print(fruits[10])
except IndexError as e:
    print(f"Lỗi: {e}")
```

3.2. Slicing (Giống List)

```
numbers = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

# Cú pháp: tuple[start:stop:step]
print(numbers[2:5])     # (2, 3, 4)
print(numbers[:4])      # (0, 1, 2, 3)
print(numbers[5:])      # (5, 6, 7, 8, 9)
print(numbers[::2])     # (0, 2, 4, 6, 8)
print(numbers[::-1])    # (9, 8, 7, 6, 5, 4, 3, 2, 1, 0)

# Slicing trả về tuple mới
sub_tuple = numbers[2:5]
print(type(sub_tuple))  # <class 'tuple'>
```

3.3. Duyệt Qua Tuple

```
fruits = ("apple", "banana", "cherry")

# Cách 1: Trực tiếp
for fruit in fruits:
    print(fruit)

# Cách 2: Với index
for i in range(len(fruits)):
    print(f"{i}: {fruits[i]}")

# Cách 3: Với enumerate (KHUYẾN NGHỊ)
for i, fruit in enumerate(fruits):
    print(f"{i}: {fruit}")

# Output:
# 0: apple
# 1: banana
# 2: cherry
```

---

4. Immutability - Không Thể Thay Đổi

4.1. Không Thể Sửa Elements

```
numbers = (1, 2, 3, 4, 5)

# ❌ Không thể sửa
try:
    numbers[0] = 10
except TypeError as e:
    print(e)  # 'tuple' object does not support item assignment

# ❌ Không thể xóa
```

```
try:
    del numbers[0]
except TypeError as e:
    print(e)  # 'tuple' object doesn't support item deletion
```

4.2. Nhưng Có Thể Tạo Tuple Mới

```
# ☑ Nối tuples → tuple mới
tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)
combined = tuple1 + tuple2
print(combined)  # (1, 2, 3, 4, 5, 6)

# ☑ Nhân tuple → tuple mới
repeated = (1, 2) * 3
print(repeated)  # (1, 2, 1, 2, 1, 2)

# ☑ "Sửa" bằng cách convert → list → tuple
numbers = (1, 2, 3, 4, 5)
numbers_list = list(numbers)  # Convert sang list
numbers_list[0] = 10           # Sửa
numbers = tuple(numbers_list)  # Convert lại tuple
print(numbers)  # (10, 2, 3, 4, 5)
```

4.3. Nested Mutable Objects

```
# ⚠ Tuple chứa list - list VẪN mutable!
data = (1, 2, [3, 4])

# Không sửa được tuple
try:
    data[0] = 10
except TypeError:
    print("Không sửa được tuple")

# ☑ Nhưng SỬA ĐƯỢC list bên trong!
data[2].append(5)
print(data)  # (1, 2, [3, 4, 5])

data[2][0] = 99
print(data)  # (1, 2, [99, 4, 5])
```

Giải thích: Tuple không đổi **reference** đến list, nhưng list tự nó vẫn mutable.

---

5. Tuple Methods

Tuple chỉ có **2 methods** (vì immutable).

5.1. `.count()` - Đếm Xuất Hiện

```
numbers = (1, 2, 3, 2, 4, 2, 5)

# Đếm số lần xuất hiện
count_2 = numbers.count(2)
print(count_2)  # 3

count_10 = numbers.count(10)
print(count_10)  # 0 (không có)

# Ứng dụng: Kiểm tra trùng
letters = ('a', 'b', 'c', 'a', 'd')
if letters.count('a') > 1:
    print("'a' xuất hiện nhiều lần")
```

5.2. `.index()` - Tìm Vị Trí

```
fruits = ("apple", "banana", "cherry", "banana")

# Tìm index đầu tiên
index = fruits.index("banana")
print(index)  # 1

# ValueError nếu không tìm thấy
try:
    index = fruits.index("grape")
except ValueError as e:
    print(f"Lỗi: {e}")

# Tìm trong range
index = fruits.index("banana", 2)  # Tìm từ index 2
print(index)  # 3
```

5.3. So Sánh với List Methods

Method	Tuple	List
`.count()`	☑	☑
`.index()`	☑	☑
`.append()`	✗	☑
`.extend()`	✗	☑
`.insert()`	✗	☑
`.remove()`	✗	☑

Method	Tuple	List
`.pop()`	✗	☑
`.sort()`	✗	☑
`.reverse()`	✗	☑

6. Tuple Packing và Unpacking

6.1. Tuple Packing

Packing là gộp nhiều giá trị thành 1 tuple.

```
# Tự động pack thành tuple
coordinates = 10.5, 20.3
print(coordinates)  # (10.5, 20.3)

# Rõ ràng hơn với ngoặc
person = ("An", 25, "Hà Nội")

# Return nhiều giá trị từ function (packing)
def get_user_info():
    name = "Bình"
    age = 30
    city = "TP.HCM"
    return name, age, city  # Packing thành tuple

user = get_user_info()
print(user)  # ('Bình', 30, 'TP.HCM')
```

6.2. Tuple Unpacking

Unpacking là tách tuple thành các biến riêng.

```
# Unpacking cơ bản
coordinates = (10.5, 20.3)
x, y = coordinates
print(x)  # 10.5
print(y)  # 20.3

# Unpacking trực tiếp
name, age, city = ("An", 25, "Hà Nội")
print(name)  # An
print(age)   # 25
print(city)  # Hà Nội

# ⚠ Số biến phải khớp số elements
try:
    a, b = (1, 2, 3)  # 2 biến nhưng 3 elements
```

```
except ValueError as e:
    print(e)  # too many values to unpack
```

6.3. Extended Unpacking với *

```
# * lấy phần còn lại thành list
a, *b, c = (1, 2, 3, 4, 5)
print(a)  # 1
print(b)  # [2, 3, 4]
print(c)  # 5

# Lấy đầu và cuối
first, *middle, last = (10, 20, 30, 40, 50)
print(first)   # 10
print(middle)  # [20, 30, 40]
print(last)    # 50

# Bỏ qua giá trị với _
name, _, city = ("An", 25, "Hà Nội")
print(name)  # An
print(city)  # Hà Nội
```

6.4. Swap Variables

```
# ☑ Pythonic swap với tuple unpacking
a = 5
b = 10

a, b = b, a  # Swap!
print(a)  # 10
print(b)  # 5

# So sánh với cách truyền thống
a = 5
b = 10

temp = a
a = b
b = temp
```

6.5. Unpacking Function Return

```
def calculate_stats(numbers):
    """Return multiple statistics"""
    total = sum(numbers)
    count = len(numbers)
```

```
    avg = total / count
    return total, count, avg  # Return tuple

# Unpacking kết quả
numbers = [1, 2, 3, 4, 5]
total, count, average = calculate_stats(numbers)

print(f"Total: {total}")      # 15
print(f"Count: {count}")      # 5
print(f"Average: {average}")  # 3.0

# Hoặc lấy toàn bộ tuple
stats = calculate_stats(numbers)
print(stats)  # (15, 5, 3.0)
```

---

7. Tuple Operations

7.1. Concatenation (+)

```
tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)

# Nối tuples
combined = tuple1 + tuple2
print(combined)  # (1, 2, 3, 4, 5, 6)

# Nối nhiều tuples
result = (1, 2) + (3, 4) + (5, 6)
print(result)  # (1, 2, 3, 4, 5, 6)
```

7.2. Repetition (*)

```
# Nhân tuple
repeated = (1, 2) * 3
print(repeated)  # (1, 2, 1, 2, 1, 2)

# Tạo tuple lớn nhanh
zeros = (0,) * 10
print(zeros)  # (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
```

7.3. Membership (in)

```
fruits = ("apple", "banana", "cherry")

# Kiểm tra tồn tại
```

```
if "banana" in fruits:
    print("Có banana!")

if "grape" not in fruits:
    print("Không có grape!")

# Ứng dụng: Validate
valid_colors = ("red", "green", "blue")
user_color = "yellow"

if user_color not in valid_colors:
    print(f"Màu '{user_color}' không hợp lệ!")
```

7.4. Length (len)

```
numbers = (1, 2, 3, 4, 5)
print(len(numbers))  # 5

empty = ()
print(len(empty))  # 0
```

7.5. Min, Max, Sum

```
numbers = (3, 7, 2, 9, 1, 5)

print(min(numbers))  # 1
print(max(numbers))  # 9
print(sum(numbers))  # 27

# Với strings
words = ("apple", "banana", "cherry")
print(min(words))  # apple (alphabetically)
print(max(words))  # cherry
```

---

8. Tuple Dùng Làm Dict Keys

8.1. Tuple là Hashable

```
# ☑ Tuple có thể làm dict key
locations = {
    (0, 0): "Origin",
    (1, 2): "Point A",
    (3, 4): "Point B"
}
```

```
print(locations[(1, 2)])  # Point A

# ❌ List KHÔNG thể làm dict key (unhashable)
try:
    invalid = {[1, 2]: "value"}
except TypeError as e:
    print(e)  # unhashable type: 'list'
```

8.2. Ứng Dụng: Coordinates Map

```
# Chess board positions
chess_board = {
    ('a', 1): 'Rook',
    ('b', 1): 'Knight',
    ('c', 1): 'Bishop',
    ('d', 1): 'Queen',
    ('e', 1): 'King'
}

piece = chess_board[('d', 1)]
print(f"Piece at d1: {piece}")  # Queen

# Grid-based game
grid = {
    (0, 0): 'Player',
    (1, 2): 'Enemy',
    (3, 4): 'Treasure'
}

if (1, 2) in grid:
    print(f"Found: {grid[(1, 2)]}")
```

8.3. ⚠️ Tuple Chứa Mutable → Unhashable

```
# Tuple chứa list → unhashable
try:
    invalid_key = {(1, [2, 3]): "value"}
except TypeError as e:
    print(e)  # unhashable type: 'list'

# ☑️ Tuple chỉ chứa immutable → hashable
valid_key = {(1, (2, 3)): "value"}  # OK
```

---

9. Named Tuples (Advanced)

9.1. Vấn Đề Với Tuple Thường

```
# Tuple thường - khó nhớ thứ tự
student = ("An", 20, "Hà Nội", 8.5)

# Phải nhớ index
name = student[0]
age = student[1]
city = student[2]
score = student[3]

# Dễ nhầm lẫn!
```

9.2. Named Tuple - Giải Pháp

```
from collections import namedtuple

# Định nghĩa named tuple
Student = namedtuple('Student', ['name', 'age', 'city', 'score'])

# Tạo instance
student = Student("An", 20, "Hà Nội", 8.5)

# Truy cập bằng tên (rõ ràng!)
print(student.name)   # An
print(student.age)    # 20
print(student.city)   # Hà Nội
print(student.score)  # 8.5

# Vẫn có thể dùng index
print(student[0])  # An

# Vẫn immutable
try:
    student.age = 21
except AttributeError as e:
    print("Cannot modify")
```

9.3. Ứng Dụng Named Tuple

```
from collections import namedtuple

# Coordinates
Point = namedtuple('Point', ['x', 'y'])
p1 = Point(10, 20)
print(f"Point: ({p1.x}, {p1.y})")

# RGB Color
Color = namedtuple('Color', ['red', 'green', 'blue'])
red = Color(255, 0, 0)
```

```
print(f"RGB: ({red.red}, {red.green}, {red.blue})")

# Employee
Employee = namedtuple('Employee', ['id', 'name', 'department', 'salary'])
emp = Employee('E001', 'Bình', 'IT', 5000)
print(f"{emp.name} - {emp.department}: ${emp.salary}")
```

---

10. Bài Tập Thực Hành

Bài 1: Swap nhiều biến

```
def rotate_values(a, b, c):
    """
    Xoay vòng: a → b → c → a

    Returns:
        tuple: (b, c, a)
    """
    return b, c, a

# Test
x, y, z = 1, 2, 3
print(f"Before: x={x}, y={y}, z={z}")

x, y, z = rotate_values(x, y, z)
print(f"After: x={x}, y={y}, z={z}")
# Before: x=1, y=2, z=3
# After: x=2, y=3, z=1
```

Bài 2: Return min và max

```
def min_max(numbers):
    """
    Tìm min và max trong 1 lần duyệt

    Returns:
        tuple: (min, max)
    """
    if not numbers:
        return None, None

    min_val = max_val = numbers[0]

    for num in numbers[1:]:
        if num < min_val:
            min_val = num
        if num > max_val:
```

```
            max_val = num

    return min_val, max_val

# Test
numbers = [3, 7, 2, 9, 1, 5]
minimum, maximum = min_max(numbers)
print(f"Min: {minimum}, Max: {maximum}")  # Min: 1, Max: 9
```

Bài 3: Coordinates distance

```
import math

def distance(point1, point2):
    """
    Tính khoảng cách Euclidean giữa 2 điểm

    Args:
        point1 (tuple): (x1, y1)
        point2 (tuple): (x2, y2)

    Returns:
        float: Khoảng cách
    """
    x1, y1 = point1
    x2, y2 = point2

    return math.sqrt((x2 - x1)**2 + (y2 - y1)**2)

# Test
p1 = (0, 0)
p2 = (3, 4)
d = distance(p1, p2)
print(f"Distance: {d}")  # 5.0
```

Bài 4: Tuple to dict

```
def tuples_to_dict(tuples_list):
    """
    Chuyển list of tuples thành dictionary

    Args:
        tuples_list (list): [(key, value), ...]

    Returns:
        dict: {key: value}
    """
```

```
    return dict(tuples_list)

# Hoặc dùng dict comprehension
def tuples_to_dict_v2(tuples_list):
    return {k: v for k, v in tuples_list}

# Test
data = [("name", "An"), ("age", 25), ("city", "Hà Nội")]
result = tuples_to_dict(data)
print(result)
# {'name': 'An', 'age': 25, 'city': 'Hà Nội'}
```

Bài 5: Enumerate với tuples

```
def index_words(sentence):
    """
    Tạo list of tuples: (index, word)

    Returns:
        list: [(index, word), ...]
    """
    words = sentence.split()
    return list(enumerate(words))

# Test
sentence = "Python is awesome"
indexed = index_words(sentence)
print(indexed)
# [(0, 'Python'), (1, 'is'), (2, 'awesome')]

# Duyệt qua
for index, word in indexed:
    print(f"{index}: {word}")
```

Bài 6: Filter tuples

```
def filter_students_by_score(students, min_score):
    """
    Lọc students có điểm >= min_score

    Args:
        students (list): [(name, score), ...]
        min_score (float): Điểm tối thiểu

    Returns:
        list: Students thỏa điều kiện
    """
```

```
    return [student for student in students if student[1] >= min_score]

# Hoặc dùng unpacking rõ ràng hơn
def filter_students_by_score_v2(students, min_score):
    return [(name, score) for name, score in students if score >= min_score]

# Test
students = [
    ("An", 8.5),
    ("Bình", 7.0),
    ("Chi", 9.0),
    ("Dũng", 6.5)
]

high_scorers = filter_students_by_score(students, 8.0)
print(high_scorers)
# [('An', 8.5), ('Chi', 9.0)]
```

Bài 7: Merge sorted tuples

```
def merge_sorted_tuples(tuple1, tuple2):
    """
    Merge 2 sorted tuples thành 1 sorted tuple

    Returns:
        tuple: Merged sorted tuple
    """
    # Cách 1: Đơn giản
    combined = tuple1 + tuple2
    return tuple(sorted(combined))

# Cách 2: Merge O(n) như merge sort
def merge_sorted_tuples_v2(tuple1, tuple2):
    result = []
    i = j = 0

    while i < len(tuple1) and j < len(tuple2):
        if tuple1[i] < tuple2[j]:
            result.append(tuple1[i])
            i += 1
        else:
            result.append(tuple2[j])
            j += 1

    result.extend(tuple1[i:])
    result.extend(tuple2[j:])

    return tuple(result)

# Test
```

```
t1 = (1, 3, 5, 7)
t2 = (2, 4, 6, 8)
merged = merge_sorted_tuples(t1, t2)
print(merged)  # (1, 2, 3, 4, 5, 6, 7, 8)
```

Bài 8: Find duplicates in tuples

```
def find_duplicates(items):
    """
    Tìm các phần tử xuất hiện > 1 lần

    Returns:
        tuple: Các phần tử trùng lặp (unique)
    """
    seen = set()
    duplicates = set()

    for item in items:
        if item in seen:
            duplicates.add(item)
        else:
            seen.add(item)

    return tuple(duplicates)

# Test
numbers = (1, 2, 3, 2, 4, 1, 5, 3)
dupes = find_duplicates(numbers)
print(dupes)  # (1, 2, 3) - thứ tự không đảm bảo
```

---

11. Best Practices

11.1. ☑ DO

```
# ☑ Dùng tuple cho data không đổi
DAYS_OF_WEEK = ("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
RGB_RED = (255, 0, 0)

# ☑ Return multiple values với tuple
def get_stats(numbers):
    return min(numbers), max(numbers), sum(numbers)

# ☑ Unpacking rõ ràng
name, age, city = person_tuple

# ☑ Dùng _ cho giá trị không dùng
name, _, city = ("An", 25, "Hà Nội")
```

```
# ☑ Named tuple cho clarity
from collections import namedtuple
Point = namedtuple('Point', ['x', 'y'])
p = Point(10, 20)

# ☑ Tuple làm dict key
positions = {(0, 0): "start", (10, 10): "end"}
```

11.2. ❌ DON'T

```
# ❌ Dùng tuple khi cần modify
# data = (1, 2, 3)
# data.append(4)  # Lỗi!

# ☑ Dùng list thay thế
data = [1, 2, 3]
data.append(4)

# ❌ Quên dấu phẩy cho tuple 1 element
single = (5)  # int, không phải tuple!

# ☑ Nhớ dấu phẩy
single = (5,)

# ❌ Unpacking không khớp số lượng
# a, b = (1, 2, 3)  # ValueError

# ☑ Dùng * hoặc khớp đúng
a, b, c = (1, 2, 3)
# hoặc
a, *rest = (1, 2, 3)

# ❌ Tuple chứa mutable làm dict key
# key = (1, [2, 3])
# d = {key: "value"}  # Lỗi!

# ☑ Chỉ dùng immutable
key = (1, (2, 3))
d = {key: "value"}
```

---

12. Tổng Kết và Checklist

☑ Kiến thức cần nắm vững

Cơ bản:

- ☐ Tạo tuple: literals `()`, `tuple()`, packing
- ☐ Tuple 1 element: cần dấu phẩy `(5,)`

- [ ] Truy cập: indexing, slicing (giống list)
- [ ] **Immutable** - không thể sửa/xóa elements

Methods:

- [ ] `.count()` - đếm xuất hiện
- [ ] `.index()` - tìm vị trí

Packing/Unpacking:

- [ ] Packing: `coords = 10, 20`
- [ ] Unpacking: `x, y = coords`
- [ ] Extended unpacking: `a, *b, c = tuple`
- [ ] Swap: `a, b = b, a`

Operations:

- [ ] Concatenation: `+`
- [ ] Repetition: `*`
- [ ] Membership: `in`, `not in`
- [ ] Functions: `len()`, `min()`, `max()`, `sum()`

Advanced:

- [ ] Tuple làm dict key (hashable)
- [ ] Named tuples (collections.namedtuple)
- [ ] Return multiple values từ function

🎯 Lưu ý quan trọng

1. **Tuple IMMUTABLE** - không thể sửa sau khi tạo
2. **Tuple 1 element** - phải có dấu phẩy: `(5,)`
3. **Tuple không có comprehension** - dùng generator + tuple()
4. **Nested mutable** - tuple chứa list, list vẫn mutable
5. **Tuple nhanh hơn list** - nhẹ hơn, performance tốt
6. **Unpacking powerful** - swap, return multiple values
7. **Dict keys** - tuple hashable, list không

---

13. Câu Hỏi Thường Gặp (FAQ)

Q1: Khi nào dùng tuple vs list?

A:

Dùng Tuple khi:

- Data **không đổi** (coordinates, config, constants)
- Return **nhiều values** từ function
- Dict **keys**
- Performance quan trọng

Dùng List khi:

- Data **thay đổi** thường xuyên
- Cần `.append()`, `.remove()`, etc.
- Không quan tâm immutability

Q2: Tại sao `(5)` không phải tuple?

A: Python coi `()` là toán tử nhóm, không phải tuple.

```
# ❌ Chỉ là int trong ngoặc
x = (5)
print(type(x))  # <class 'int'>

# ☑ Dấu phẩy tạo tuple
x = (5,)
print(type(x))  # <class 'tuple'>

# Hoặc không cần ngoặc
x = 5,
print(type(x))  # <class 'tuple'>
```

Q3: Làm sao để "sửa" tuple?

A: Không thể sửa trực tiếp, nhưng có thể tạo tuple mới.

```
original = (1, 2, 3, 4, 5)

# Cách 1: Convert → list → sửa → tuple
temp_list = list(original)
temp_list[0] = 10
modified = tuple(temp_list)
print(modified)  # (10, 2, 3, 4, 5)

# Cách 2: Slicing + concatenation
modified = (10,) + original[1:]
print(modified)  # (10, 2, 3, 4, 5)

# Cách 3: Comprehension
modified = tuple(10 if i == 0 else x for i, x in enumerate(original))
print(modified)  # (10, 2, 3, 4, 5)
```

Q4: Tuple có thể chứa list không?

A: Có, nhưng tuple sẽ **không hashable** (không làm dict key).

```
# ☑ Tuple chứa list - OK
data = (1, 2, [3, 4])
print(data)  # (1, 2, [3, 4])

# Sửa list bên trong - OK
data[2].append(5)
print(data)  # (1, 2, [3, 4, 5])

# ❌ Nhưng không dùng làm dict key
try:
    d = {data: "value"}
except TypeError as e:
    print("Unhashable!")  # Unhashable type: 'list'
```

Q5: Unpacking có bắt buộc khớp số lượng không?

A: Có, trừ khi dùng *.

```
# ❌ Không khớp
try:
    a, b = (1, 2, 3)
except ValueError as e:
    print("Too many values")

# ☑ Khớp chính xác
a, b, c = (1, 2, 3)

# ☑ Hoặc dùng * lấy phần còn lại
a, *b = (1, 2, 3)
print(a)  # 1
print(b)  # [2, 3]
```
