Python Programming for Data Science

Basic Data Structures: Lists, Tuples, Sets, Dictionaries

Quanskill

Nội dung bài học

Quanskill

Giới thiệu về Data Structures

Lists - Danh sách

Tuples - Bộ dữ liệu

Sets - Tập hợp

Dictionaries - Từ điển

So sánh và Lựa chọn Data Structure

Kết luận

Giới thiệu về Data Structures

Tại sao Data Structures quan trọng?

Quanskill

- **Lưu trữ dữ liệu**: Tổ chức và quản lý datasets lớn
- **Xử lý dữ liệu**: Thao tác hiệu quả với nhiều loại dữ liệu khác nhau
- **Machine Learning**: Chuẩn bị features và labels cho mô hình
- **Data Analysis**: Phân tích, thống kê và visualization

Trong bài này chúng ta sẽ học

4 cấu trúc dữ liệu cơ bản nhất của Python được sử dụng hàng ngày trong Data Science

Python Data Structures Overview

Quanskill

Data Structure	Ordered	Mutable	Duplicates
List	o	o	o
Tuple	o	x	o
Set	x	o	x
Dictionary	o	o	x (keys)

- **Ordered**: Thứ tự các phần tử được duy trì
- **Mutable**: Có thể thay đổi sau khi tạo
- **Duplicates**: Cho phép các phần tử trùng lặp

Lists - Danh sách

Lists - Khái niệm cơ bản

Quanskill

List là cấu trúc dữ liệu có thứ tự, có thể thay đổi và cho phép duplicate values.

Tạo Lists

```
# Create empty list
empty_list = []
print(empty_list)  # Output: []

# Create list with data
student_scores = [85, 92, 78, 96, 88]
print(student_scores)  # Output: [85, 92, 78, 96, 88]

# Mixed data types
mixed_data = ["Alice", 25, 85.5, True]
print(mixed_data)  # Output: ['Alice', 25, 85.5, True]
```

Giải thích: Lists sử dụng dấu ngoặc vuông [ ] và các phần tử cách nhau bởi dấu phẩy.

Lists - Accessing Elements (Indexing)

Quanskill

Index và Slicing

```python
# Dataset of house prices (in thousands)
house_prices = [250, 180, 320, 150, 280, 400, 200]

# Access by index (starts from 0)
print(house_prices[0])   # Output: 250 (first element)
print(house_prices[3])   # Output: 150 (fourth element)
print(house_prices[-1])  # Output: 200 (last element)

# Slicing [start:end:step]
print(house_prices[1:4])   # Output: [180, 320, 150]
print(house_prices[:3])    # Output: [250, 180, 320]
print(house_prices[4:])    # Output: [280, 400, 200]
print(house_prices[::2])   # Output: [250, 320, 280, 200]
```

Giải thích: Index âm đếm từ cuối về. Slicing giúp lấy một phần của list.

Lists - Adding và Removing Elements

Quanskill

Thao tác với Lists

```
# Customer data
customers = ["John", "Mary", "Bob"]

# Adding elements
customers.append("Alice")          # Add at end
print(customers)  # Output: ['John', 'Mary', 'Bob', 'Alice']

customers.insert(1, "David")       # Insert at position 1
print(customers)  # Output: ['John', 'David', 'Mary', 'Bob', 'Alice']

# Removing elements
customers.remove("Bob")            # Remove first occurrence
print(customers)  # Output: ['John', 'David', 'Mary', 'Alice']

removed = customers.pop()          # Remove and return last
print(removed)    # Output: Alice
print(customers) # Output: ['John', 'David', 'Mary']

del customers[1]                   # Delete by index
print(customers) # Output: ['John', 'Mary']
```

Phân tích dữ liệu bán hàng

```python
# Daily sales data for a week
daily_sales = [1200, 800, 1500, 900, 1800, 2200, 1600]
days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]

# Basic statistics
total_sales = sum(daily_sales)
average_sales = total_sales / len(daily_sales)
max_sales = max(daily_sales)
min_sales = min(daily_sales)

print(f"Total sales: ${total_sales}")       # Output: Total sales: $10000
print(f"Average sales: ${average_sales}")   # Output: Average sales: $1428.57
print(f"Max sales: ${max_sales}")           # Output: Max sales: $2200
print(f"Min sales: ${min_sales}")           # Output: Min sales: $800

# Find best selling day
best_day_index = daily_sales.index(max_sales)
print(f"Best day: {days[best_day_index]}")  # Output: Best day: Sat
```

Tuples - Bộ dữ liệu

Tuples - Khái niệm cơ bản

Quanskill

Tuple là cấu trúc dữ liệu có thứ tự, **KHÔNG thể thay đổi** và cho phép duplicate values.

Tạo Tuples

```python
# Create empty tuple
empty_tuple = ()
print(empty_tuple)  # Output: ()

# Create tuple with data
coordinates = (10.5, 20.3)
print(coordinates)  # Output: (10.5, 20.3)

# Single element tuple (note the comma!)
single_tuple = (42,)
print(single_tuple)  # Output: (42,)

# Without parentheses (tuple packing)
student_info = "Alice", 20, "Computer Science"
print(student_info)  # Output: ('Alice', 20, 'Computer Science')
print(type(student_info))  # Output: <class 'tuple'>
```

Giải thích: Tuples sử dụng dấu ngoặc tròn ( ) hoặc chỉ cần dấu phẩy.

Quanskill

Truy cập và Unpacking

```python
# RGB color values
red_color = (255, 0, 0)
green_color = (0, 255, 0)
blue_color = (0, 0, 255)

# Access by index (same as lists)
print(red_color[0])     # Output: 255
print(red_color[1:])    # Output: (0, 0)

# Tuple unpacking (very useful!)
r, g, b = red_color
print(f"Red: {r}, Green: {g}, Blue: {b}")
# Output: Red: 255, Green: 0, Blue: 0

# Multiple assignment using tuples
name, age, grade = "Bob", 22, "A"
print(f"Student: {name}, Age: {age}, Grade: {grade}")
# Output: Student: Bob, Age: 22, Grade: A

# Swapping variables
x, y = 10, 20
x, y = y, x  # Swap using tuple unpacking
print(f"x: {x}, y: {y}")  # Output: x: 20, y: 10
```

Quanskill

Dữ liệu tọa độ GPS và Database records

```python
# GPS coordinates for store locations
store_locations = [
    ("Store A", 40.7128, -74.0060),  # New York
    ("Store B", 34.0522, -118.2437), # Los Angeles
    ("Store C", 41.8781, -87.6298)   # Chicago
]

# Processing GPS data
for store_data in store_locations:
    name, lat, lon = store_data  # Unpack tuple
    print(f"{name}: Latitude {lat}, Longitude {lon}")

# Output:
# Store A: Latitude 40.7128, Longitude -74.006
# Store B: Latitude 34.0522, Longitude -118.2437
# Store C: Latitude 41.8781, Longitude -87.6298

# Function returning multiple values as tuple
def calculate_stats(numbers):
    return min(numbers), max(numbers), sum(numbers)/len(numbers)

data = [10, 20, 30, 40, 50]
min_val, max_val, avg_val = calculate_stats(data)
print(f"Min: {min_val}, Max: {max_val}, Average: {avg_val}")
# Output: Min: 10, Max: 50, Average: 30.0
```

Sets - Tập hợp

Sets - Khái niệm cơ bản

Quanskill

Set là collection **không có thứ tự**, có thể thay đổi và **KHÔNG cho phép duplicate**.

Tạo Sets

```python
# Create empty set
empty_set = set()  # Note: {} creates empty dict, not set!
print(empty_set)   # Output: set()

# Create set with data
unique_ages = {25, 30, 35, 25, 30}  # Duplicates will be removed
print(unique_ages)  # Output: {25, 30, 35}

# Convert list to set (remove duplicates)
customer_ids = [101, 102, 101, 103, 102, 104]
unique_customers = set(customer_ids)
print(unique_customers)  # Output: {101, 102, 103, 104}

# Create set from string
letters = set("hello")
print(letters)  # Output: {'h', 'e', 'l', 'o'}
```

Giải thích: Sets tự động loại bỏ duplicates và không duy trì thứ tự.

Quanskill

Thao tác cơ bản với Sets

```python
# Website visitors
monday_visitors = {"user1", "user2", "user3", "user4"}
tuesday_visitors = {"user3", "user4", "user5", "user6"}

# Add and remove elements
monday_visitors.add("user7")
print(monday_visitors)  # Output: {'user1', 'user2', 'user3', 'user4', 'user7'}

monday_visitors.remove("user1")  # Raises error if not found
print(monday_visitors)  # Output: {'user2', 'user3', 'user4', 'user7'}

monday_visitors.discard("user999")  # No error if not found
print(monday_visitors)  # Output: {'user2', 'user3', 'user4', 'user7'}

# Check membership
print("user2" in monday_visitors)    # Output: True
print("user5" in monday_visitors)    # Output: False

# Set length
print(len(monday_visitors))  # Output: 4
```

Quanskill

Phép toán tập hợp

```python
# Customer segments
premium_customers = {"Alice", "Bob", "Charlie", "David"}
active_customers = {"Bob", "Diana", "Eve", "Charlie"}

# Union - all customers
all_customers = premium_customers | active_customers
print(all_customers)
# Output: {'Alice', 'Bob', 'Charlie', 'David', 'Diana', 'Eve'}

# Intersection - customers in both segments
both_segments = premium_customers & active_customers
print(both_segments)  # Output: {'Bob', 'Charlie'}

# Difference - premium customers who are not active
premium_only = premium_customers - active_customers
print(premium_only)   # Output: {'Alice', 'David'}

# Symmetric difference - customers in only one segment
exclusive_customers = premium_customers ^ active_customers
print(exclusive_customers)  # Output: {'Alice', 'David', 'Diana', 'Eve'}
```

Quanskill

Phân tích dữ liệu khách hàng

```python
# E-commerce customer behavior analysis
email_subscribers = {"cust1", "cust2", "cust3", "cust4", "cust5"}
purchase_customers = {"cust2", "cust4", "cust6", "cust7", "cust8"}
mobile_app_users = {"cust1", "cust3", "cust6", "cust9", "cust10"}

# Find customers who subscribed but never purchased
subscribed_no_purchase = email_subscribers - purchase_customers
print(f"Subscribed but no purchase: {subscribed_no_purchase}")
# Output: Subscribed but no purchase: {'cust1', 'cust3', 'cust5'}

# Find highly engaged customers (all three activities)
highly_engaged = email_subscribers & purchase_customers & mobile_app_users
print(f"Highly engaged customers: {highly_engaged}")
# Output: Highly engaged customers: set()

# Total unique customers across all platforms
total_customers = email_subscribers | purchase_customers | mobile_app_users
print(f"Total unique customers: {len(total_customers)}")
# Output: Total unique customers: 10
```

Dictionaries - Từ điển

Dictionary lưu trữ dữ liệu dưới dạng key-value pairs, có thứ tự (Python 3.7+) và có thể thay đổi.

Tạo Dictionaries

```python
# Create empty dictionary
empty_dict = {}
print(empty_dict)  # Output: {}

# Create dictionary with data
student_grades = {
    "Alice": 85,
    "Bob": 92,
    "Charlie": 78,
    "Diana": 96
}
print(student_grades)
# Output: {'Alice': 85, 'Bob': 92, 'Charlie': 78, 'Diana': 96}

# Using dict() constructor
product_prices = dict(laptop=999, mouse=25, keyboard=75)
print(product_prices)  # Output: {'laptop': 999, 'mouse': 25, 'keyboard': 75}
```

Giải thích: Keys phải unique và immutable (string, number, tuple). Values có thể là bất kỳ data type nào.

Truy cập và thay đổi dữ liệu

```python
# Sales data by region
sales_by_region = {
    "North": 15000,
    "South": 12000,
    "East": 18000,
    "West": 14000
}

# Access values by key
print(sales_by_region["North"])  # Output: 15000
print(sales_by_region.get("North"))  # Output: 15000
print(sales_by_region.get("Central", 0))  # Output: 0 (default value)

# Modify existing value
sales_by_region["North"] = 16000
print(sales_by_region["North"])  # Output: 16000

# Add new key-value pair
sales_by_region["Central"] = 10000
print(sales_by_region)
# Output: {'North': 16000, 'South': 12000, 'East': 18000, 'West': 14000, 'Central':
    10000}

# Remove key-value pair
del sales_by_region["Central"]
removed_value = sales_by_region.pop("West", 0)
print(f"Removed West sales: {removed_value}")  # Output: Removed West sales: 14000
```

Quanskill

Các method quan trọng

```python
# Employee information
employees = {
    "emp001": {"name": "Alice", "dept": "IT", "salary": 75000},
    "emp002": {"name": "Bob", "dept": "Sales", "salary": 65000},
    "emp003": {"name": "Charlie", "dept": "IT", "salary": 80000}
}

# Get all keys, values, items
print(employees.keys())
# Output: dict_keys(['emp001', 'emp002', 'emp003'])

print(list(employees.values())[0])
# Output: {'name': 'Alice', 'dept': 'IT', 'salary': 75000}

# Iterate through dictionary
for emp_id, info in employees.items():
    print(f"ID: {emp_id}, Name: {info['name']}, Salary: ${info['salary']}")

# Output:
# ID: emp001, Name: Alice, Salary: $75000
# ID: emp002, Name: Bob, Salary: $65000
# ID: emp003, Name: Charlie, Salary: $80000
```

Phân tích dữ liệu bán hàng

```python
# Product sales data
sales_data = {
    "products": ["laptop", "mouse", "keyboard", "monitor"],
    "quantities": [50, 200, 150, 75],
    "prices": [999, 25, 75, 299],
    "categories": ["electronics", "accessories", "accessories", "electronics"]
}

# Calculate total revenue by product
for i in range(len(sales_data["products"])):
    product = sales_data["products"][i]
    revenue = sales_data["quantities"][i] * sales_data["prices"][i]
    print(f"{product}: ${revenue}")

# Output:
# laptop: $49950
# mouse: $5000
# keyboard: $11250
# monitor: $22425

# Count products by category
category_count = {}
for category in sales_data["categories"]:
    category_count[category] = category_count.get(category, 0) + 1

print(category_count)  # Output: {'electronics': 2, 'accessories': 2}
```

So sánh và Lựa chọn Data Structure

Khi nào sử dụng Data Structure nào?

Quanskill

List - Sử dụng khi

- Cần thứ tự và cho phép duplicates
- Thường xuyên thêm/xóa elements
- Time series data, sequences

Tuple - Sử dụng khi

- Dữ liệu không thay đổi (coordinates, RGB values)
- Return multiple values từ functions
- Dictionary keys

Khi nào sử dụng Data Structure nào?

Quanskill

Set - Sử dụng khi

- Loại bỏ duplicates
- Membership testing (in operator)
- Set operations (union, intersection)

Dictionary - Sử dụng khi

- Key-value relationships
- Fast lookups by key
- Grouping và counting data

Kết hợp tất cả Data Structures

```python
# Customer transaction data
customers = {
    "customer001": {
        "name": "Alice Johnson",
        "transactions": [150, 200, 75, 300],  # List of amounts
        "location": (40.7128, -74.0060),      # Tuple for coordinates
        "purchase_categories": {"electronics", "clothing", "books"}  # Set
    },
    "customer002": {
        "name": "Bob Smith",
        "transactions": [100, 250, 180],
        "location": (34.0522, -118.2437),
        "purchase_categories": {"electronics", "sports", "books"}
    }
}

# Analysis
for customer_id, data in customers.items():
    name = data["name"]
    total_spent = sum(data["transactions"])    # List operation
    avg_transaction = total_spent / len(data["transactions"])
    lat, lon = data["location"]                # Tuple unpacking
    num_categories = len(data["purchase_categories"])  # Set size

    print(f"Customer: {name}")
    print(f"Total spent: ${total_spent}, Average: ${avg_transaction:.2f}")
    print(f"Location: ({lat}, {lon}), Categories: {num_categories}")
    print("---")
```

Kết luận

Tóm tắt bài học

Những gì chúng ta đã học

- **Lists**: Ordered, mutable, allows duplicates - dùng cho sequences
- **Tuples**: Ordered, immutable, allows duplicates - dùng cho fixed data
- **Sets**: Unordered, mutable, no duplicates - dùng cho unique values
- **Dictionaries**: Ordered, mutable, unique keys - dùng cho key-value pairs

Tips cho Data Science

- Sử dụng Lists cho time series và numerical data
- Tuples cho coordinates, fixed records
- Sets cho unique values và set operations
- Dictionaries cho data grouping và fast lookups

Questions?