

Apply Data Science & AI with Python

LECTURE 4. Data Structure



Outline

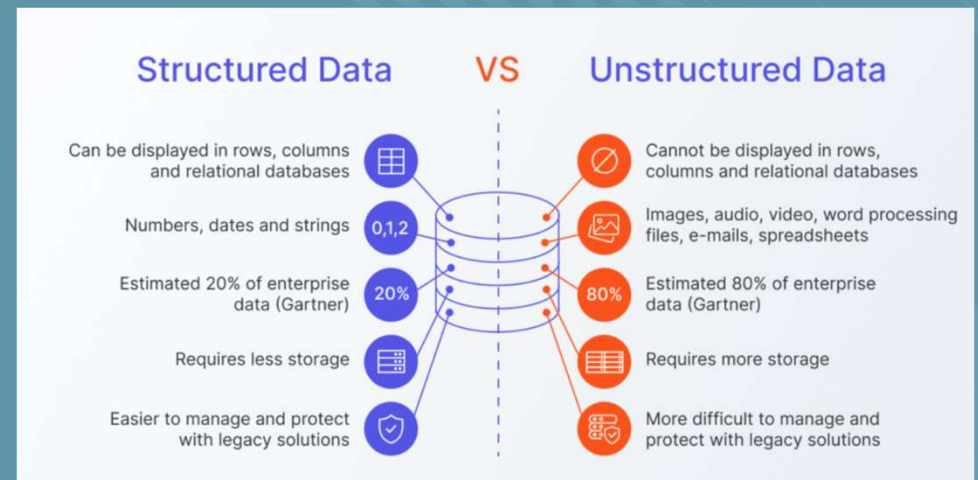
04 Built-in Data Structure

- List
- Tuple
- Dictionary
- Set
- Collections Package



01. Why we need data structure?

- Tổ chức dữ liệu hiệu quả
 - Truy xuất dữ liệu nhanh chóng
 - Tối ưu performance của chương trình
- ➔ Không thể thiếu trong xử lý và phân tích dữ liệu



01. How to use List?

```
✓  
0s [1] 1 L = [10, 20, 30, 40]  
    2  
    3 # Length of a list  
    4 print(len(L)) # Output: 4
```

⇒ 4

```
✓  
1s [7] 1 L = [1, 'two', 3.14, [0, 3, 5]]  
    2 print(L)
```

⇒ [1, 'two', 3.14, [0, 3, 5]]

- Khởi tạo list bằng dấu “[]”
- Hàm len để xác định độ dài của list



01. How to use List?

```
✓ [6] 1 # Append a value to the end
0s    2 L.append(50)
      3 print(L) # Output: [10, 20, 30, 40, 50]
      4
      5 # Addition concatenates lists
      6 new_list = L + [60, 70, 80]
      7 print(new_list) # Output: [10, 20, 30, 40, 50, 60, 70, 80]
      8
      9 # Insert a value at a specific index
     10 L.insert(2, 25) # Inserts 25 at index 2
     11 print(L) # Output: [10, 20, 25, 30, 40, 50]
     12
     13 # Reverse the list in-place
     14 L.reverse()
     15 print(L) # Output: [50, 40, 30, 25, 20, 10]
     16
     17 # Remove a specific value from the list
     18 L.remove(25)
     19 print(L) # Output: [50, 40, 30, 20, 10]

⇒ [50, 40, 30, 20, 10, 50]
   [50, 40, 30, 20, 10, 50, 60, 70, 80]
   [50, 40, 25, 30, 20, 10, 50]
   [50, 10, 20, 30, 25, 40, 50]
   [50, 10, 20, 30, 40, 50]
```

List là một class trong python, cung cấp nhiều build-in methods :

- **Append:** đẩy thêm phần tử mới vào cuối list
- **“+”:** 2 list có thể nối với nhau bằng phép cộng
- **Insert:** đẩy phần tử mới vào vị trí được chỉ định
- **Reverse:** đảo ngược thứ tự list
- **Remove:** xóa phần tử tại vị trí được chỉ định




01. How to use List?

Methods	Description
<code>append()</code>	To add elements at the end of the list
<code>extend()</code>	To add all elements of a list to another list
<code>sort()</code>	To sort items in a list in ascending order
<code>reverse()</code>	To reverse the order of items in the list
<code>pop()</code>	To removes and an element at the given index
<code>remove()</code>	To removes an item from the list
<code>insert()</code>	To insert an item at the defined index
<code>len()</code>	To count the number of items in a list




01. Indexing and Slicing in List?

✓ 0s  1 L = [4, 5, 6, 7, 8]

✓ 0s [14] 1 # Python use zero-based indexing
2 # Select element at index 3
3 L[3]

 7

✓ 0s [15] 1 # Python support negative indexing
2 # -1 indicate the last element
3 L[-1], L[-2]

 (8, 7)



01. Indexing and Slicing in List?

```
✓ 0s [16] 1 # Python support select many values by using slicing  
      2 # Select elements from index 1 to index 2 (<3)  
      3 L[1:3]
```

```
⇒ [5, 6]
```

```
✓ 0s [17] 1 # If the starting index or ending index is not specified  
      2 # use the start and end elements in the list  
      3  
      4 print(L[1:])  
      5 print(L[:4])
```

```
⇒ [5, 6, 7, 8]  
   [4, 5, 6, 7]
```

```
✓ 0s [18] 1 # Provide step to slide over list  
      2 print(L[::2])  
      3 print(L[1:4:2])
```

```
⇒ [4, 6, 8]  
   [5, 7]
```

```
▶ 1 L[::-1]
```

Quizz?

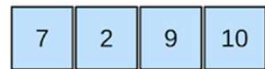


```
1 L[::-1]
```



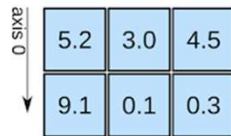
01. Indexing and Slicing in List?

1D array



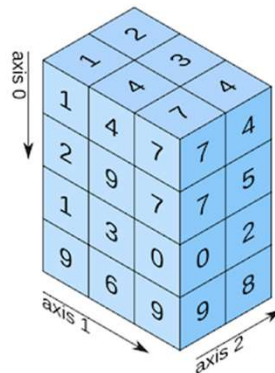
axis 0
shape: (4,)

2D array



axis 1
shape: (2, 3)

3D array

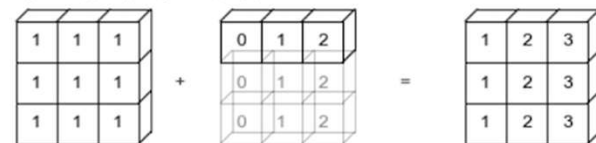


shape: (4, 3, 2)

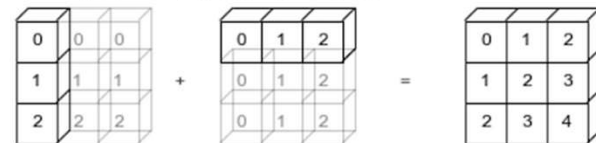
`np.arange(3)+5`



`np.ones((3,3))+np.arange(3)`



`np.arange(3).reshape((3,1))+np.arange(3)`



02. How to use Tuple?

```
[ ] 1 # List
    2 L = [4, 5, 6]
    3
    4 # Tuple
    5 L = (4, 5, 6)
```

```
✓ [19] 1 # List
      2 L = [4, 5, 6]
      3 L[1] = 10
      4 L
```

```
⇒ [4, 10, 6]
```

```
⊘ [20] 1 # Tuple
      2 L = (4, 5, 6)
      3 L[1] = 10
      4 L
```

```
⇒ -----
TypeError                                Traceback (most recent call last)
<ipython-input-20-6c904bf60443> in <cell line: 3>()
      1 # Tuple
      2 L = (4, 5, 6)
----> 3 L[1] = 10
      4 L
```

```
TypeError: 'tuple' object does not support item assignment
```

- Khởi tạo tuple bằng dấu “()”
- Tuple cơ bản sử dụng giống List
- Tuple là một immutable object, không thay đổi được giá trị



02. How to use Tuple?

```
✓ [21] 1 def multiple_return(a, b, c):  
0s      2     return a, b, c  
        3  
        4 multiple_return(5,6,7)
```

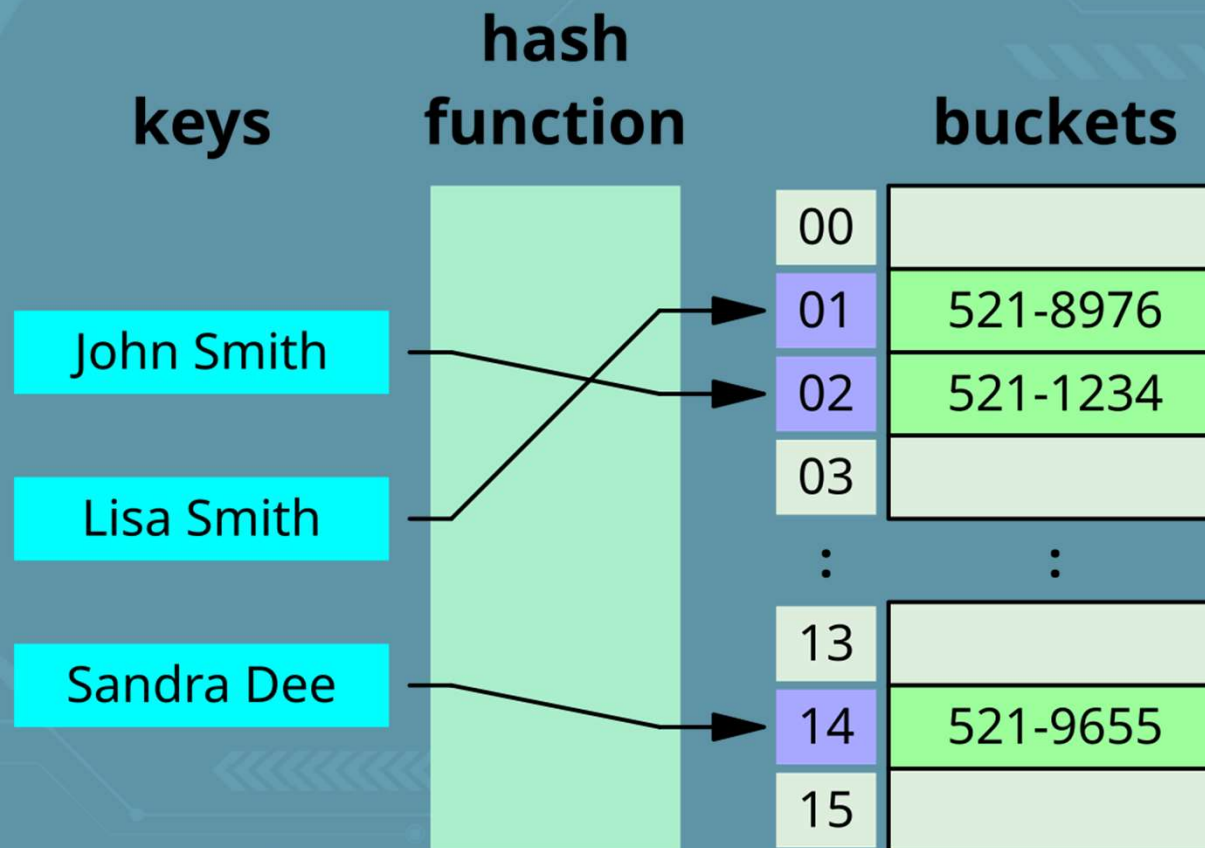
⇒ (5, 6, 7)

```
✓ [22] 1 a, b, c = multiple_return(5,6,7) 💡  
0s
```

- Hàm trả về nhiều giá trị, kết quả được đóng gói thành tuple
- Các giá trị có thể tách ra lần lượt từ trái qua phải



01. How to use Dictionary?



01. How to use Dictionary?

```
✓ 0s ▶ 1 # Creating a dictionary to store student names and their grades
2 grades = {
3     "Alice": 85,
4     "Bob": 92,
5     "Charlie": 78
6 }
7
8 # Access a value by key
9 print(grades["Bob"]) # Output: 92
10
11 # Add a new key-value pair
12 grades["David"] = 88
13 print(grades) # Output: {'Alice': 85, 'Bob': 92, 'Charlie': 78, 'David': 88}
14
15 # Update an existing value
16 grades["Alice"] = 90
17 print(grades) # Output: {'Alice': 90, 'Bob': 92, 'Charlie': 78, 'David': 88}
18
19 # Remove a key-value pair
20 del grades["Charlie"]
21 print(grades) # Output: {'Alice': 90, 'Bob': 92, 'David': 88}
22
23 # Get all keys in the dictionary
24 print(grades.keys()) # Output: dict_keys(['Alice', 'Bob', 'David'])
25
26 # Get all values in the dictionary
27 print(grades.values()) # Output: dict_values([90, 92, 88])
28
29 # Check if a key exists in the dictionary
30 print("Alice" in grades) # Output: True
31
```

```
92
{'Alice': 85, 'Bob': 92, 'Charlie': 78, 'David': 88}
{'Alice': 90, 'Bob': 92, 'Charlie': 78, 'David': 88}
{'Alice': 90, 'Bob': 92, 'David': 88}
dict_keys(['Alice', 'Bob', 'David'])
dict_values([90, 92, 88])
True
```



01. How to use Dictionary?

```
✓ 0s ▶ 1 # Creating a dictionary to store student names and their grades
2 grades = {
3     "Alice": 85,
4     "Bob": 92,
5     "Charlie": 78
6 }
7
8 # Access a value by key
9 print(grades["Bob"]) # Output: 92
10
11 # Add a new key-value pair
12 grades["David"] = 88
13 print(grades) # Output: {'Alice': 85, 'Bob': 92, 'Charlie': 78, 'David': 88}
14
15 # Update an existing value
16 grades["Alice"] = 90
17 print(grades) # Output: {'Alice': 90, 'Bob': 92, 'Charlie': 78, 'David': 88}
18
19 # Remove a key-value pair
20 del grades["Charlie"]
21 print(grades) # Output: {'Alice': 90, 'Bob': 92, 'David': 88}
22
23 # Get all keys in the dictionary
24 print(grades.keys()) # Output: dict_keys(['Alice', 'Bob', 'David'])
25
26 # Get all values in the dictionary
27 print(grades.values()) # Output: dict_values([90, 92, 88])
28
29 # Check if a key exists in the dictionary
30 print("Alice" in grades) # Output: True
31
```

```
92
{'Alice': 85, 'Bob': 92, 'Charlie': 78, 'David': 88}
{'Alice': 90, 'Bob': 92, 'Charlie': 78, 'David': 88}
{'Alice': 90, 'Bob': 92, 'David': 88}
dict_keys(['Alice', 'Bob', 'David'])
dict_values([90, 92, 88])
True
```



01. How to use Set?

```
✓ 0s ▶ 1 # Creating a set with unique values
2 fruits = {"apple", "banana", "cherry"}
3
4 # Add a new element to the set
5 fruits.add("orange")
6 print(fruits) # Output: {'apple', 'orange', 'banana', 'cherry'}
7
8 # Adding a duplicate value (has no effect, as sets ensure uniqueness)
9 fruits.add("banana")
10 print(fruits) # Output: {'apple', 'orange', 'banana', 'cherry'}
11
12 # Removing an element from the set
13 fruits.remove("cherry")
14 print(fruits) # Output: {'apple', 'orange', 'banana'}
15
```

⇨ {'orange', 'banana', 'apple', 'cherry'}
{'orange', 'banana', 'apple', 'cherry'}
{'orange', 'banana', 'apple'}

- Set là một kiểu dữ liệu gần giống dict chỉ có mỗi key, không có value
- Các giá trị trong set là duy nhất
- Set hoạt động giống các tính chất của set trong toán học



01. Other Data Structure?

```
✓ [26] 1 from collections import Counter
0s      2
        3 # Count the frequency of elements in a list
        4 fruits = ["apple", "banana", "apple", "orange", "banana", "apple"]
        5 fruit_counter = Counter(fruits)
        6 print(fruit_counter) # Output: Counter({'apple': 3, 'banana': 2, 'orange': 1})
        7
        8 # Most common element
        9 print(fruit_counter.most_common(1)) # Output: [('apple', 3)]
       10
```

⇨ Counter({'apple': 3, 'banana': 2, 'orange': 1})
[('apple', 3)]

- Hoạt động như một dict
- Key là các giá trị duy nhất, value là số lần xuất hiện các giá trị đó trong list
- Có nhiều methods hữu ích như tìm các phần tử phổ biến nhất



01. Other Data Structure?

```
0s 1 dct = {}  
2 print(dct[5])  
  
-----  
KeyError                                Traceback (most recent call last)  
<ipython-input-28-26f780199d7e> in <cell line: 2>()  
      1 dct = {}  
----> 2 print(dct[5])  
  
KeyError: 5  
  
Next steps: Explain error
```

- Dict sẽ báo lỗi nếu key không tồn tại
- Defaultdict sẽ tự tạo key với giá trị default nếu key không tồn tại

```
✓ [27] 0s 1 from collections import defaultdict  
2  
3 # Default value for missing keys is an empty list  
4 dd = defaultdict(list)  
5 dd["fruits"].append("apple")  
6 dd["fruits"].append("banana")  
7 dd["vegetables"].append("carrot")  
8 print(dd) # Output: defaultdict(<class 'list'>, {'fruits': ['apple', 'banana'], 'vegetables': ['carrot']})  
9  
10 # Accessing a non-existing key returns a default empty list  
11 print(dd["unknown"]) # Output: []  
12  
defaultdict(<class 'list'>, {'fruits': ['apple', 'banana'], 'vegetables': ['carrot']})  
[]
```



01. Other Data Structure?

```
✓ [29] 1 from collections import OrderedDict
0s      2
        3 # Create an OrderedDict
        4 od = OrderedDict()
        5 od["first"] = 1
        6 od["second"] = 2
        7 od["third"] = 3
        8 print(od) # Output: OrderedDict([('first', 1), ('second', 2), ('third', 3)])
        9
       10 # Iterates in the order the keys were added
       11 for key, value in od.items():
       12     print(f"{key}: {value}")
       13 # Output:
       14 # first: 1
       15 # second: 2
       16 # third: 3
       17
```

```
⇒ OrderedDict([('first', 1), ('second', 2), ('third', 3)])
first: 1
second: 2
third: 3
```

- Thứ tự của các key trong dict là ngẫu nhiên
- Thứ tự key trong OrderedDict là theo đúng thứ tự phần tử được thêm vào



01. Other Data Structure?

Table of Contents

collections — Container datatypes

- **ChainMap** objects
 - **ChainMap**
Examples and Recipes
- **Counter** objects
- **deque** objects
 - **deque** Recipes
- **defaultdict** objects
 - **defaultdict**
Examples
- **namedtuple()** Factory Function for Tuples with Named Fields
- **OrderedDict** objects
 - **OrderedDict**
Examples and Recipes
- **UserDict** objects
- **UserList** objects
- **UserString** objects

Previous topic

calendar — General calendar-related functions

collections — Container datatypes

Source code: [Lib/collections/_init_.py](https://docs.python.org/3/library/collections/_init_.py)

This module implements specialized container datatypes providing alternatives to Python's general purpose built-in containers, [dict](#), [list](#), [set](#), and [tuple](#).

namedtuple()	factory function for creating tuple subclasses with named fields
deque	list-like container with fast appends and pops on either end
ChainMap	dict-like class for creating a single view of multiple mappings
Counter	dict subclass for counting hashable objects
OrderedDict	dict subclass that remembers the order entries were added
defaultdict	dict subclass that calls a factory function to supply missing values
UserDict	wrapper around dictionary objects for easier dict subclassing
UserList	wrapper around list objects for easier list subclassing
UserString	wrapper around string objects for easier string subclassing



01. Summary

Loại Dữ Liệu	Đặc Điểm Chính	Chức Năng	Ví Dụ
List (Danh sách)	<ul style="list-style-type: none">- Có thể chứa các phần tử trùng lặp.- Có thể thay đổi (mutable).- Duy trì thứ tự phần tử.	<ul style="list-style-type: none">- Lưu trữ và truy cập các phần tử theo chỉ mục.- Thêm, xóa, sắp xếp các phần tử dễ dàng.	<code>L = [1, 2, 3, 4]</code>
Set (Tập hợp)	<ul style="list-style-type: none">- Chỉ chứa các phần tử duy nhất (không trùng lặp).- Không duy trì thứ tự phần tử.- Có thể thay đổi (mutable).	<ul style="list-style-type: none">- Loại bỏ các phần tử trùng lặp.- Kiểm tra nhanh phần tử có trong tập hợp hay không.- Các phép toán như hợp, giao, hiệu.	<code>s = {1, 2, 3}</code>
Dictionary (Từ điển)	<ul style="list-style-type: none">- Lưu trữ cặp key-value (khóa-giá trị).- Có thể thay đổi (mutable).- Không cho phép khóa trùng lặp, nhưng giá trị có thể trùng lặp.	<ul style="list-style-type: none">- Truy xuất dữ liệu theo khóa (key).- Dễ dàng thêm, xóa, cập nhật cặp key-value.- Tìm kiếm nhanh theo khóa.	<code>d = {'a': 1, 'b': 2}</code>
Tuple (Bộ dữ liệu)	<ul style="list-style-type: none">- Không thể thay đổi (immutable).- Có thể chứa các phần tử trùng lặp.- Duy trì thứ tự phần tử.	<ul style="list-style-type: none">- Lưu trữ dữ liệu bất biến (không thể thay đổi).- Sử dụng khi cần dữ liệu cố định và tránh thay đổi vô tình.	<code>t = (1, 2, 3, 4)</code>

