



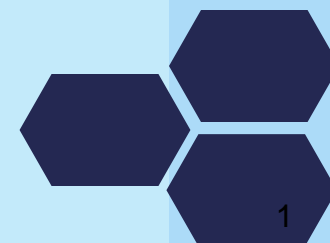
Bài giảng

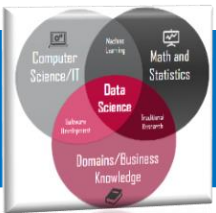
NHẬP MÔN

KHOA HỌC DỮ LIỆU

(Data Science)

ThS. Vương Xuân Chí
0903.270 567
vxchi@ntt.edu.vn

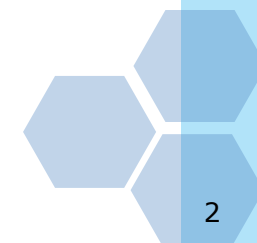


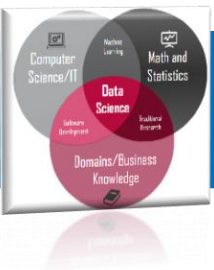


CHƯƠNG

2

Ngôn ngữ Python

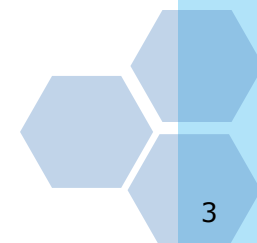


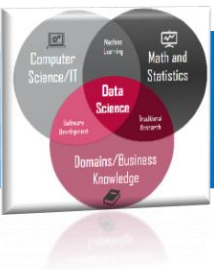


Bài 2.4

Tập hợp

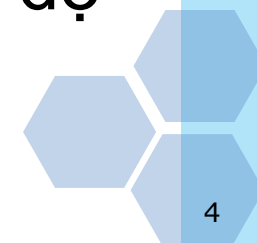
- 1. Set (tập hợp) và Frozenset (tập hợp tĩnh)***
- 2. Dictionary (từ điển)***
- 3. Module và Package***

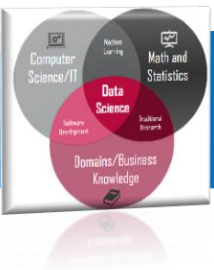




Ôn lại

- **Kiểu dữ liệu tuần tự**: là kiểu dữ liệu cho phép xử lý dữ liệu bằng cách xử lý từng-phần-tử-con-một
- **Danh sách (list)**: dãy các phần tử, khai báo bên trong cặp ngoặc vuông, nội dung có thể thay đổi
- **Hàng (tuple)**: dãy các phần tử, khai báo bên trong cặp ngoặc tròn, nội dung cố định (không thay đổi)
- **Range (miền)**: có thể xem như một dạng tuple đặc biệt gồm các số nguyên, chuyên dùng cho lặp for
- **Chuỗi (str)**: một dạng tuple đặc biệt gồm nhiều chuỗi có độ dài 1 ký tự



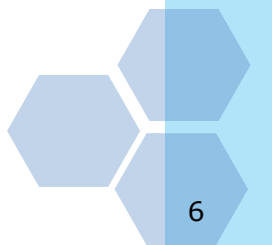


Ôn lại

- Các kiểu dữ liệu này có chung đặc điểm:
 - *Bản chất là các đối tượng, được viết một cách tự nhiên*
 - *Rất nhiều phương thức hỗ trợ việc xử lý*
 - *Sử dụng chung 2 hệ thống chỉ mục (âm và dương)*
 - *Sử dụng chung kỹ thuật cắt lát (bằng chỉ mục)*
 - *Sử dụng chung 3 phép toán: **+**, *****, **in***
- Chuỗi có rất nhiều kỹ thuật định dạng nội dung
- List và Tuple có thể được tạo bằng comprehension
- Nhiều hàm dựng sẵn (built-in) xử lý các kiểu dữ liệu này: **len**, **max**, **min**, **all**, **any**, **filter**, **sorted**, **sum**, **zip**, ...



Set (tập hợp) và Frozenset (tập hợp tĩnh)





Giới thiệu và khởi tạo

- **Set** = tập hợp các đối tượng (không trùng nhau)
- Khai báo trực tiếp bằng cách liệt kê các phần tử con đặt trong cặp ngoặc nhọn {}, ngăn cách bởi phẩy.

Ví dụ: `basket = {'apple', 'orange', 'apple', 'pear'}`
`print(basket)`
`#{'orange', 'pear', 'apple'}` : xóa trùng nhau

Ví dụ: `a = {5, 2, 3, 1, 4}`
`print("a=", a)` `#a = {1, 2, 3, 4, 5}`

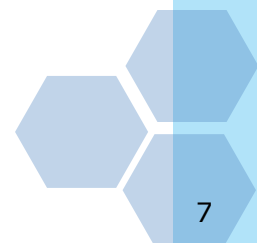
- Tạo set bằng constructor

`s1 = set([1, 2, 3, 4])` `# {1, 2, 3, 4}`

`s2 = set((1, 1, 1))` `# {1}`

`s3 = s1 - s2` `# {2, 3, 4}`

`s4 = set(range(1, 100))` `# {1, 2, 3, ..., 98, 99}`





Khởi tạo Set

- **Tạo set** bằng **set comprehension**

```
# a = {'r', 'd'}
```

```
a = {x for x in 'abracadabra' if x not in 'abc'}
```

- **Set** không thể chứa những đối tượng mutable (có thể bị thay đổi), mặc dù chính set lại có thể thay đổi

```
a = set([1,2], [2,3]) # lỗi list
```

```
a = set((1,2), (2,3)) # {(1, 2), (2, 3)}
```

```
a.add("abc") # {(1, 2), "abc", (2, 3)}
```

- **Frozenset** giống set, nhưng không thể bị thay đổi

```
b = frozenset((1,2), (2,3)) # {(1, 2), (2, 3)}
```

```
b.add("abc") # lỗi
```




Set

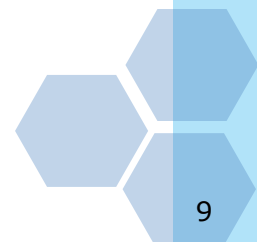
■ Set Hỗn hợp

Ví dụ:

```
my_set = {8.0, "Sinh viên", (1, 2, 3)}  
print("ketqua_Set=", my_set)  
#Output: ketqua_Set= {'Sinh viên', 8.0, (1, 2, 3)}
```

Ví dụ:

```
# Khởi tạo my_set  
my_set = {1, 3}  
my_set.add(2)    #thêm  
my_set.update([2, 3, 4]) # Cập nhật  
my_set.update([4, 5], {1, 6, 8}) # Cập nhật set  
print(my_set)  
  
# Output: {1, 2, 3, 4, 5, 6, 8}
```





Các phép toán trên set

```
a = set('abracadabra') # {'d', 'r', 'c', 'b', 'a'}
```

```
b = set('alacazam') # {'z', 'c', 'm', 'l', 'a'}
```

Phép Hiệu: thuộc a nhưng không thuộc b

```
print(a - b) # {'r', 'd', 'b'}
```

Phép Hợp: thuộc a hoặc b

```
# {'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
```

```
print(a | b)
```

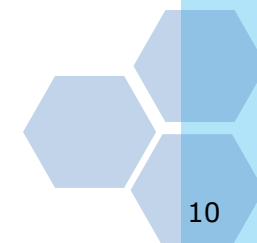
Phép Giao: thuộc cả a và b

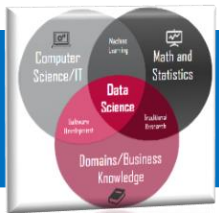
```
print(a & b) # {'a', 'c'}
```

Phép Xor: thuộc hoặc a, hoặc b nhưng không phải cả 2

```
# {'r', 'd', 'b', 'm', 'z', 'l'}
```

```
print(a ^ b)
```

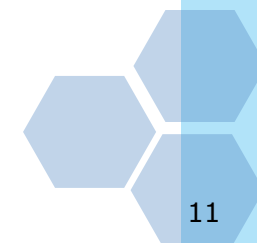


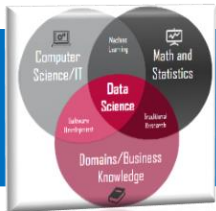


Các phương thức của set

Một số phương thức thường hay sử dụng

- **add(e)** : thêm e vào tập hợp
- **clear()** : xóa mọi phần tử trong tập hợp
- **copy()** : tạo một bản sao của tập hợp
- **difference(x)** : tương đương với phép trừ đi x
- **difference_update(x)** : loại bỏ những phần tử trong x khỏi tập
- **discard(e)** : bỏ e khỏi tập
- **remove(e)** : bỏ e khỏi tập, báo lỗi nếu không tìm thấy e
- **union(x)** : tương đương với phép hợp với x
- **intersection(x)** : tương đương với phép giao với x





Các phương thức của set

Một số phương thức thường hay sử dụng

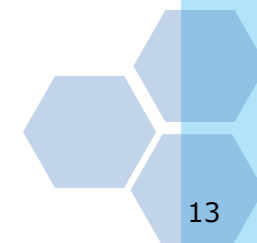
- **`isdisjoint(x)`** : trả về True nếu tập không có phần chung nào với **`x`**
- **`issubset(x)`** : trả về True nếu tập là con của **`x`**, tương đương với phép so sánh **`<=x`**
- **`issuperset(x)`** : trả về True nếu **`x`** là tập con của tập, tương đương với phép so sánh **`>=x`**
- **`pop()`** : lấy một phần tử ra khỏi tập (không biết trước)
- **`symmetric_difference(x)`** : tương đương với phép **`^x`**

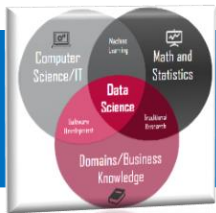


Các phương thức của set

Các hàm thường dùng trên set bao gồm **all()**, **any()**, **enumerate()**, **len()**, **max()**, **min()**, **sorted()**, **sum()**. Chức năng của những hàm này khá giống list, tuple.

```
num = {1, 3, 5, 6, 8, 20, 7, 8, 29, 33}  
print("Số lớn nhất là :", max(num) )  
#Số lớn nhất là : 33
```





Vòng lặp for của set()

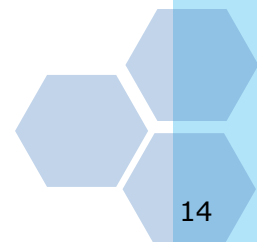
- Không thể truy cập các phần tử trong một Set bằng cách sử dụng chỉ mục, vì Set không lưu trữ các phần tử theo thứ tự nhập ban đầu.
- Sử dụng vòng lặp for để truy cập các phần tử của một Set.

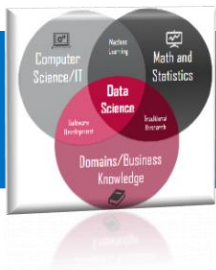
Ví dụ:

```
setFruits = { 'lemon', 'orange', 'apple', 'pear' }  
for x in setFruits:  
    print(x)
```

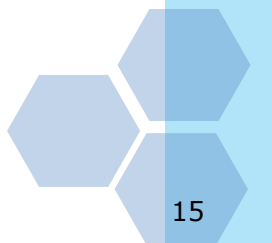
Ví dụ:

```
for letter in set("chivuong") :  
    print(letter)
```





Dictionary (từ điển)





Dictionary

- Từ điển là một danh sách các từ (key) và định nghĩa của nó (value)
 - Yêu cầu các key không được trùng nhau, như vậy có thể xem từ điển như một loại set

- Từ điển có thể khai báo theo cú pháp của set

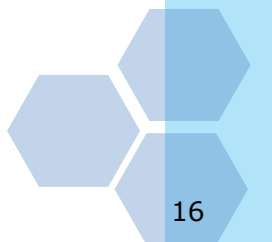
```
dict = {1: 'one', 2: 'two', 3: 'three' }
```

```
print(dict[1]) # 'one'
```

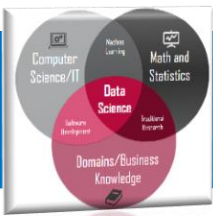
```
dict[4]='four'
```

```
print(dict)
```

```
{1: 'one', 2: 'two', 3: 'three', 4: 'four' }
```



Dictionary



Python Dictionary

py_dict = { 1: 'Apple', 2: 'OnePlus' }

key value key value

↓ ↓ ↓ ↓

└──────────┘ └──────────┘

Item 1 Item 2



Dictionary

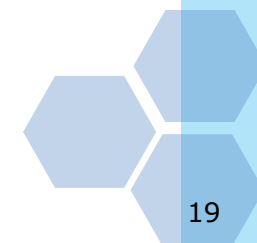
- Chú ý: chỉ những loại dữ liệu immutable (không thể thay đổi) mới có thể dùng làm key của từ điển
`dict = { (1,2,3) : "abc", 3.1415 : "abc" }`
`dict = { [1,2,3] : "abc" } # lỗi`
- Một số phép toán / phương thức thường dùng
 - `len(d)` : trả về độ dài của từ điển (số cặp key-value)
 - `del d[k]` : xóa key k (và value tương ứng)
 - `k in d` : trả về True nếu có key k trong từ điển
 - `k not in d` : trả về True nếu không có key k trong từ điển
 - `pop(k)` : trả về value tương ứng với k và xóa cặp này đi
 - `popitem()` : trả về (và xóa) một cặp (key, value) tùy ý



Dictionary

Một số phép toán / phương thức thường dùng

- **get (k)** : lấy về value tương ứng với key k
 - Khác phép `[]` ở chỗ `get` trả về `None` nếu `k` không phải là key
- **update (w)** : ghép các nội dung từ từ điển `w` vào từ điển hiện tại (nếu key trùng thì lấy value từ `w`)
- **items ()** : trả về list các cặp (key, value)
- **keys ()** : trả về các key của từ điển
- **values ()** : trả về các value của từ điển
- **pop (k)** : trả về value tương ứng với `k` và xóa cặp này đi
- **popitem ()** : trả về (và xóa) một cặp (key, value) tùy ý





Dictionary (từ điển)

Dùng zip để ghép 2 list thành từ điển

```
l1 = ["a", "b", "c"]
```

```
l2 = [1, 2, 3]
```

```
c = zip(l1, l2)
```

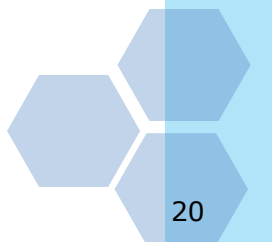
```
for i in c:
```

```
    print(i)
```

```
('a', 1)
```

```
('b', 2)
```

```
('c', 3)
```





Dictionary (từ điển)

❖ Tạo mới từ điển

- Ví dụ: Tạo 1 từ điển

tạo một dict với key là các số nguyên

```
Dict = {1: 'Nguyen', 2: 'Van', 3: 'An' }
```

```
print(Dict)
```

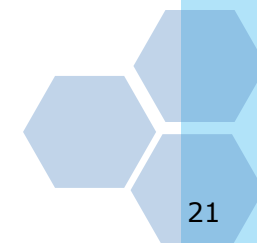
```
# {1: 'Nguyen', 2: 'Van', 3: 'an'}
```

- # tạo một dict với key hỗn hợp

```
Dict = { 'Name': 'dict', 1: [1, 2, 3, 4] }
```

```
print(Dict)
```

```
# {'Name': 'dict', 1: [1, 2, 3, 4]}
```





Dictionary (từ điển)

❖ Thêm các giá trị vào từ điển

▪ Ví dụ:

tạo một từ điển

```
Dict = { }
```

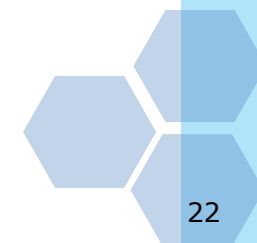
thêm các giá trị

```
Dict[0] = 'Xin chào'
```

```
Dict[1] = 'lớp học Khoa học dữ liệu'
```

```
print(Dict)
```

{0: ' Xin chào ', 1: ' Khoa học dữ liệu'}





Dictionary (từ điển)

❖ Truy cập giá trị của từ điển

▪ Ví dụ:

tạo một dict mới

```
Dict = {1: 'Lớp', 2: 'Khoa học', 3: 'Dữ liệu'}
```

truy cập theo cách thông thường

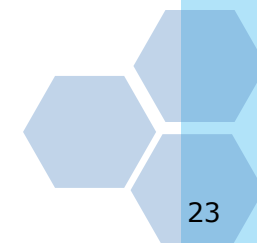
```
print(Dict[1])
```

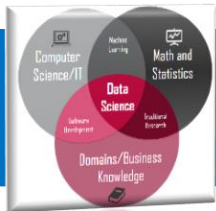
Lớp

truy cập bằng phương thức get()

```
print(Dict.get(3))
```

Dữ liệu





Dictionary (từ điển)

❖ Chuyển từ điển thành chuỗi: str()

```
Dict = { 'Ten' : 'An' , 'Tuoi' : 20 } ;  
print("Chuỗi tương đương là :", % str (Dict))  
#Chuỗi tương đương là : { 'Ten' : 'An' , 'Tuoi' : 20 }
```

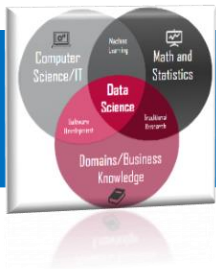
❖ Cập nhật 1 giá trị của từ điển: update()

Hàm update() sẽ thêm vào một giá trị mới, hoặc sẽ cập nhật một giá trị nếu có các khóa trùng nhau.

```
Dict = { 'Cúc' : 18 , 'Lan' : 19 , 'Hồng' : 20 , 'Mai' : 27 }  
Dict.update({ 'Lan' : 21 }) #cập nhật tuổi cho Lan  
Dict.update({ 'Đào' : 25 }) #thêm một phần tử mới là Đào  
print(Dict)  
#{ 'Cúc' : 18 , 'Lan' : 21 , 'Hồng' : 20 , 'Mai' : 27 , 'Đào' : 25 }
```

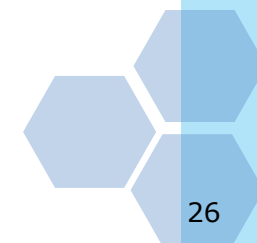



Module và Package



Module

- Một file mã nguồn trong python được xem là một **module**
 - Có phần mở rộng **.py**
 - Mọi hàm, biến, kiểu trong file là các thành phần của module
- **Sử dụng module:**
 - Khai báo import module đó: **import <tên-module>**
 - Có thể khai báo import cùng lúc nhiều module cách nhau bởi *dấu phẩy*
 - Nếu muốn sử dụng các hàm, biến trong module thì cần viết trước tên module đó
 - Hoặc có thể import riêng một hàm hoặc nhiều hàm, cú pháp:
from <tên-module> import func1, func2, ..., funcN





Package

- **Package** = Thư mục các module (lưu trữ vật lý)
 - `import numpy`
 - `A = array([1, 2, 3])` # lỗi
 - `A = numpy.array([1, 2, 3])` # ok
 - `import numpy as np`
 - `B = np.array([1, 2, 3])` # ok
 - `from numpy import array`
 - `C = array([1, 2, 3])` # ok
- **Module** và **Package** giúp quản lý tốt hơn mã nguồn
- **Gom, nhóm** các hàm, biến, lớp xử lý cùng một chủ đề, giúp phân cấp và sử dụng dễ dàng hơn



Thank You!

ThS. Vương Xuân Chí
0903.270 567
vxchi@ntt.edu.vn

