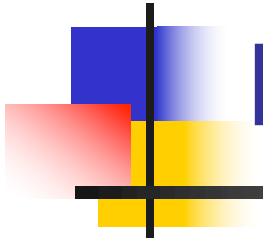
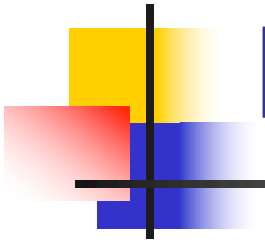


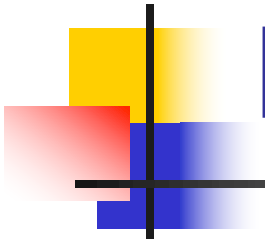
Bài 8: Các kiểu dữ liệu chuẩn

TS. Trịnh Tuấn Đạt
Viện CNTT-TT, ĐHBK Hà Nội



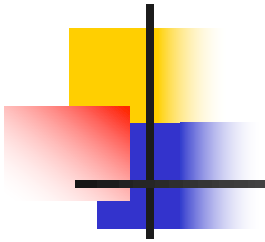
Nội dung

1. Kiểu dữ liệu Number
2. Kiểu dữ liệu String
3. Kiểu dữ liệu List
4. Kiểu dữ liệu Tuple
5. Kiểu dữ liệu Dictionary



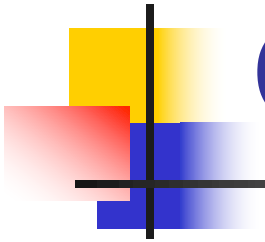
Nội dung

1. **Kiểu dữ liệu Number**
2. Kiểu dữ liệu String
3. Kiểu dữ liệu List
4. Kiểu dữ liệu Tuple
5. Kiểu dữ liệu Dictionary



1. Kiểu dữ liệu Number

- Chuyển đổi kiểu dữ liệu số
- Một số hàm toán học trong Python
- Hàm xử lý số ngẫu nhiên



Chuyển đổi kiểu dữ liệu số

- **int(x):** chuyển đổi số x thành số thuần nguyên
- **float(x):** chuyển đổi số x thành số thực
- **complex(x):** chuyển đổi số x thành số phức với phần thực là x và phần ảo là 0
- **complex(x, y):** chuyển đổi số x và y thành số phức với phần thực là x và phần ảo là y

Một số hàm toán học trong Python

Hàm	Miêu tả
Hàm abs(x)	Trị tuyệt đối của x
Hàm ceil(x)	Số nguyên nhỏ nhất mà không nhỏ hơn x
Hàm cmp(x, y)	Trả về -1 nếu x < y, trả về 0 nếu x == y, hoặc 1 nếu x > y
Hàm exp(x)	Trả về e^x
Hàm fabs(x)	Giá trị tuyệt đối của x
Hàm floor(x)	Số nguyên lớn nhất mà không lớn hơn x
Hàm log(x)	Trả về lnx, với x> 0
Hàm log10(x)	Trả về log10(x), với x> 0 .
Hàm max(x1, x2,...)	Trả về số lớn nhất
Hàm min(x1, x2,...)	Trả về số nhỏ nhất
Hàm modf(x)	Trả về phần nguyên và phần thập phân của x. Cả hai phần có cùng dấu với x và phần nguyên được trả về dưới dạng một số thực
Hàm pow(x, y)	Trả về giá trị của x**y.
Hàm round(x [,n])	Làm tròn x về n chữ số sau dấu thập phân. Python làm tròn theo cách sau: round(0.5) là 1.0 và round(-0.5) là -1.0
Hàm sqrt(x)	Trả về căn bậc hai của x, với x > 0

Một số hàm toán học trong Python

```
import math   # This will import math module

print (math.ceil(45.17))
print (math.floor(100.12))
```

```
46
100
```

Hàm xử lý số ngẫu nhiên

Hàm	Miêu tả
Hàm choice(seq)	Một item ngẫu nhiên trong một list, tuple, hoặc một string
Hàm randrange ([start,] stop [,step])	Một phần tử được lựa chọn một cách ngẫu nhiên từ dãy (start, stop, step)
Hàm random()	Một số thực ngẫu nhiên r trong dãy 0<= r <1
Hàm seed([x])	Thiết lập giá trị nguyên bắt đầu mà được sử đụng trong bộ sinh số ngẫu nhiên. Bạn nên gọi hàm này trước khi gọi bất cứ hàm ngẫu nhiên nào khác. Hàm này trả về None
Hàm shuffle(lst)	Sắp xếp các item trong list một cách ngẫu nhiên
Hàm uniform(x, y)	Một số thực ngẫu nhiên r trong dãy x<= r <y

Hàm xử lý số ngẫu nhiên

```
import random

l = [11, 17, 10, 19]
print(random.choice(l))
print(random.choice('HELLO WORLD'))
print(random.choice(range(10)))
```

```
19
L
3
```

```
17
R
1
```

```
11
L
7
```

```
import random
print(random.uniform(5,7))
```

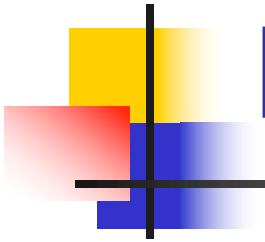
```
6.687918880989755
```

```
5.989161836184021
```

Hàm xử lý số ngẫu nhiên

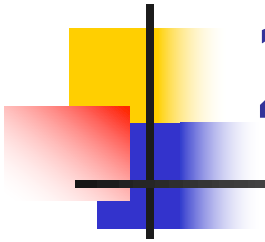
- Hàm randrange ([start,] stop [,step]): sinh số ngẫu nhiên từ start đến stop-1 với bước nhảy là step
- start, step: tùy chọn. Không đưa vào, mặc định start = 0, step = 1

```
import random

# Sinh so le ngau nhien tu 1-99
print (random.randrange(1, 100, 2))

# Sinh so ngau nhien tu 0-99
print (random.randrange(100))
```



Nội dung

1. Kiểu dữ liệu Number
2. **Kiểu dữ liệu String**
3. Kiểu dữ liệu List
4. Kiểu dữ liệu Tuple
5. Kiểu dữ liệu Dictionary



2. Kiểu dữ liệu String

- Truy cập giá trị trong String
- So sánh String
- Cập nhật giá trị String
- Một số phương thức làm việc với String

Truy cập giá trị trong String

- Nếu chỉ số dương, đánh số từ trái sang phải, bắt đầu từ 0
- Nếu chỉ số âm, đánh số từ phải sang trái, bắt đầu từ -1

```
var1 = 'Hello World!'
var2 = "Python Programming"

print("var1[0]: ", var1[0])
print("var2[1:5]: ", var2[1:5])
print("var1[-2]: ", var1[-1])
print("var2[-5:-1]: ", var2[-5:-1])
```

```
var1[0]:  H
var2[1:5]:  ytho
var1[-2]:  !
var2[-5:-1]:  mmin
```

So sánh String

- Tất cả các toán tử quan hệ (như <,>, <=, >=, ==, !=) cũng có thể áp dụng cho các String. Các chuỗi được so sánh dựa trên giá trị ASCII hoặc Unicode.
- Ví dụ:

```
print("HOANG"=="HOANG")
print("afsha">='Afsha')
print("Z"!="z")

True
True
True
```

Cập nhật String

- VD: đổi chữ 'h' và 'w' trong xâu 'hello world!' sang chữ hoa

```
var1 = 'hello world!'
var1[0] = 'H'
var1[6] = 'W'
print (var1)
```

```
var1 = 'hello world!'
var1 = 'H' + var1[1:6] + 'W' +
var1[7:]
print (var1)
```

Một số ký tự đặc biệt

- Ký tự xuống dòng: \n
- Ký tự tab: \t
- Ký tự ": \"
- Ký tự ': \'

```
print ('123\n456\t789')
```

```
123
456    789
```

```
print ('Hello everyone. My name's Alice')
```

```
print ('Hello everyone. My name\'s Alice')
print ("Hello everyone. My name's Alice")
print ("Hello everyone. My name\'s Alice")
```

```
Hello everyone. My name's Alice
Hello everyone. My name's Alice
Hello everyone. My name's Alice
```

Hàm và phương thức

- Hàm và phương thức là tương tự nhau: đoạn code thực hiện công việc xác định, nhận vào các tham số và trả về kết quả gì đó, giúp tăng tính tái sử dụng
- Phương thức: lập trình hướng đối tượng. Cần gọi phương thức từ **một đối tượng** nào đó.
- Cú pháp gọi hàm

```
function1(arg1, arg2)      module.function2(arg1, arg2)
```

- Cú pháp phương thức

```
object.function1(arg1, arg2)
```

Một số phương thức làm việc với xâu

- Phương thức string.capitalize(): Viết hoa chữ cái đầu tiên của chuỗi string

```
s1 = "xin chao cac ban"
print (s1.capitalize())

Xin chao cac ban
```

Một số phương thức làm việc với xâu

- Phương thức string.count(str, beg= 0, end=len(string)): đếm số lần xuất hiện của chuỗi str trong chuỗi string hoặc chuỗi con của string
- Tham số beg, end là tùy chọn, nếu không đưa vào, mặc định tìm trong toàn bộ xâu string

```
s1 = "Xin chao cac ban. Hom nay cac ban khoe khong"
print (s1.count("cac"))
print (s1.count("cac", 15))
print (s1.count("cac", 15, 20))
print (s1.count("cac", 15, len(s1)))
```

```
2
1
0
1
```

Một số phương thức làm việc với xâu

- Phương thức string.endswith(str, beg=0, end=len(string)): kiểm tra chuỗi string hoặc chuỗi con của nó có kết thúc bằng chuỗi str (True hoặc False)

```
s1 = "Xin chao cac ban. Hom nay cac ban khoe khong"
print (s1.endswith("khong"))
print (s1.endswith("Xin", 0, 3))
```

```
True
True
```

Một số phương thức làm việc với xâu

- Phương thức string.startswith(str, beg=0, end=len(string)): kiểm tra chuỗi string hoặc chuỗi con của nó có bắt đầu bằng chuỗi str (True hoặc False)

```
s1 = "Xin chao cac ban. Hom nay cac ban khoe khong"
print (s1.startswith("Xin"))
print (s1.startswith("chao", 4, 15))
```

```
True
True
```

Một số phương thức làm việc với xâu

- Phương thức string.find(str, beg=0 end=len(string)): tìm vị trí xuất hiện của chuỗi str trong chuỗi string hoặc chuỗi con của string. Nếu không tìm thấy, trả về -1

```
s1 = "0123456789"
print (s1.find("567"))
print (s1.find("123"))
print (s1.find("13"))
```

```
5
1
-1
```

Một số phương thức làm việc với xâu

- Phương thức string.isalpha(): trả về True nếu chuỗi string có ít nhất 1 ký tự và tất cả ký tự là chữ cái. Ngược lại, trả về False

```
s1 = ""                          False
s2 = "123"                       False
s3 = "abc"                       True
s4 = "12a"                       False
print (s1.isalpha())
print (s2.isalpha())
print (s3.isalpha())
print (s4.isalpha())
```

Một số phương thức làm việc với xâu

- Phương thức string.isdigit(): trả về True nếu chuỗi string có ít nhất 1 ký tự và tất cả ký tự là chữ số. Ngược lại, trả về False

```
s1 = ""                    False
s2 = "123"                 True
s3 = "abc"                 False
s4 = "12a"                 False
print (s1.isdigit())
print (s2.isdigit())
print (s3.isdigit())
print (s4.isdigit())
```

Một số phương thức làm việc với xâu

- Phương thức string.islower(): Trả về True nếu xâu string chứa ít nhất một chữ cái, và tất cả các chữ cái là ký tự viết thường. Nếu không trả về False

```
s1 = ""                          False
s2 = "abc"                       True
s3 = "123"                       False
s4 = "12%-.*a"                   True
s5 = "12A"                       False
print (s1.islower())
print (s2.islower())
print (s3.islower())
print (s4.islower())
print (s5.islower())
```

Một số phương thức làm việc với xâu

- Phương thức string.isupper(): Trả về True nếu xâu string chứa ít nhất một chữ cái, và tất cả các chữ cái là ký tự viết hoa. Nếu không trả về False

```
s1 = ""                          False
s2 = "abc"                       False
s3 = "123"                       False
s4 = "12%-.*a"                   False
s5 = "12%*A"                     True
print (s1.isupper())
print (s2.isupper())
print (s3.isupper())
print (s4.isupper())
print (s5.isupper())
```

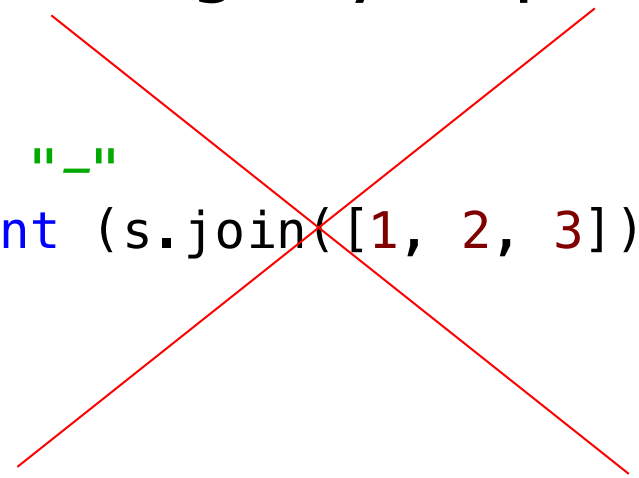
Một số phương thức làm việc với xâu

- Phương thức string.join(seq): Dùng xâu string làm ký hiệu kết nối, nối các phần tử trong dãy seq thành một chuỗi

```
s = "-"
print (s.join(["1", "2", "3"]))
```

```
1-2-3
```

```
s = "-"
print (s.join([1, 2, 3]))
```



```
s = ","
print (s.join(("1", "2", "3")))
```

```
1,2,3
```

Một số phương thức làm việc với xâu

- Phương thức string.upper(): Chuyển đổi tất cả chữ thường có trong chuỗi string sang kiểu chữ hoa

```
s = "!@#$1234abcdAbAbABCD"          !@#$1234ABCDABABABCD
print(s.upper())
```

- Phương thức string.lower(): Chuyển đổi tất cả chữ hoa có trong chuỗi string sang kiểu chữ thường

```
s = "!@#$1234abcdAbAbABCD"          !@#$1234abcdababababcd
print(s.lower())
```

Một số phương thức làm việc với xâu

- Phương thức string.replace(old, new [, max]): Thay thế tất cả sự xuất hiện của xâu old trong chuỗi string thành xâu new. Tham số max là tùy chọn, chỉ ra số lần thay thế tối đa

```
s1 = "Xin chao cac ban. Hom nay cac ban khoe khong"
print (s1.replace("cac", "Cac"))
print (s1.replace("cac", "Cac", 2))
print (s1.replace("cac", "Cac", 1))
```

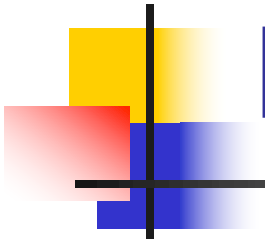
```
Xin chao Cac ban. Hom nay Cac ban khoe khong
Xin chao Cac ban. Hom nay Cac ban khoe khong
Xin chao Cac ban. Hom nay cac ban khoe khong
```

Một số phương thức làm việc với xâu

- Phương thức string.split(str=" ", num=string.count(str)): Chia chuỗi string theo ký tự phân tách là str (mặc định là space nếu không được cung cấp) và trả về List các chuỗi con. Nếu cung cấp tham số num, chỉ dùng num ký tự phân tách

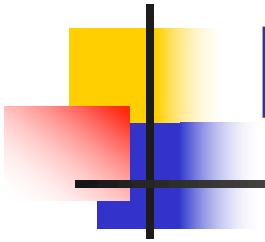
```
s = "Xin chao-cac ban nhe"
print(s.split())
print(s.split(" "))
print(s.split("-"))
print(s.split("cac"))
print(s.split(" ", 1))
print(s.split(" ", 2))
```

```
['Xin', 'chao-cac', 'ban', 'nhe']
['Xin', 'chao-cac', 'ban', 'nhe']
['Xin chao', 'cac ban nhe']
['Xin chao-', ' ban nhe']
['Xin', 'chao-cac ban nhe']
['Xin', 'chao-cac', 'ban nhe']
```



Nội dung

1. Kiểu dữ liệu Number
2. Kiểu dữ liệu String
3. **Kiểu dữ liệu List**
4. Kiểu dữ liệu Tuple
5. Kiểu dữ liệu Dictionary



Kiểu dữ liệu List

- Truy cập các phần tử List
- Thay đổi giá trị một phần tử trong List
- Xóa phần tử trong List
- Các thao tác cơ bản với List
- Hàm xử lý cơ bản với List
- Phương thức xử lý cơ bản với List

Truy cập các phần tử của List

- Nếu chỉ số dương, đánh số từ trái sang phải, bắt đầu từ 0
- Nếu chỉ số âm, đánh số từ phải sang trái, bắt đầu từ -1

```
list1 = ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5, 6, 7 ]
list3 = [7, 6, 5, 4, 3, 2, 1 ]

print ("list1[0]: ", list1[0])
print ("list2[1:5]: ", list2[1:5])
print ("list3[-1]: ", list3[-1])
print ("list3[-5:-1]: ", list3[-5:-1])
```

```
list1[0]:  physics
list2[1:5]:  [2, 3, 4, 5]
list3[-1]:  1
list3[-5:-1]:  [5, 4, 3, 2]
```

Thay đổi giá trị một phần tử trong list

- Có thể thay đổi các phần tử của List. Cú pháp:

```
<ten_list>[index]=<giatri>
```

- Ví dụ:

```
list = ['vatly', 'hoahoc', 1997, 2000];

print("Gia tri co san tai chi muc thu 2:",
list[2])
list[2] = 2001;
print("Gia tri moi tai chi muc thu 2:", list[2])

Gia tri co san tai chi muc thu
2: 1997
Gia tri moi tai chi muc thu 2:
2001
```

Xóa phần tử trong List

- Để xóa một phần tử trong List, sử dụng lệnh del hoặc phương thức remove(obj) với obj là phần tử muốn xóa. Sẽ bị lỗi nếu obj không có trong List

```
list1 = ['vatly', 'hoahoc', 1997, 2000];

del list1[2];
print("List sau khi xoa phan tu chi so 2:", list1)
list1.remove(2000);
print("List sau khi xoa phan tu 2000:", list1)
```

```
List sau khi xoa phan tu chi so 2: ['vatly', 'hoahoc', 2000]
List sau khi xoa phan tu 2000: ['vatly', 'hoahoc']
```

Xóa phần tử trong List

- Có thể xóa nhiều phần tử, hoặc xóa toàn bộ list như sau

```
list1 = ['vatly', 'hoahoc', 1997, 2000];

del list1[1:3];
print(list1)
del list1
```

```
['vatly', 2000]
```

Các thao tác cơ bản với list

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 3	['Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1,2,3] : print (x,end = ' ')	1 2 3	Iteration

Hàm xử lý cơ bản với list

- Hàm len(list): Trả về độ dài của list
- Hàm max(list): Trả về phần tử có giá trị lớn nhất trong list
- Hàm min(list): Trả về phần tử có giá trị nhỏ nhất trong list
- Hàm list(seq): Chuyển đổi một tuple thành list

```
list1 = [2, 5, 10, 7]
tuple1 = (2, 3, 4)
print(len(list1))              4
print(max(list1))              10
print(min(list1))              2
print(tuple1)                  (2, 3, 4)
print(list(tuple1))            [2, 3, 4]
```

Các phương thức cơ bản với List

- Phương thức list.count(obj): Trả về số lần obj xuất hiện trong list

```
aList = [123, 'xyz', 'hoang', 'abc', 123];

print("So lan 123 xuat hien:", aList.count(123))
print("So lan hoang xuat hien:",
aList.count('hoang'))
```

```
So lan 123 xuat hien: 2
So lan hoang xuat hien: 1
```

Các phương thức cơ bản với List

- Phương thức list.extend(seq): Chèn thêm các phần tử trong seq vào cuối list

```
list1 = [1, 2, 3]
list2 = [4, 5]
tuple1 = (6, 7)
list1.extend(list2)
print("Sau khi them list2:", list1)
list1.extend(tuple1)
print("Sau khi them tuple1:", list1)
```

```
Sau khi them list2: [1, 2, 3, 4, 5]
Sau khi them tuple1: [1, 2, 3, 4, 5, 6, 7]
```

Các phương thức cơ bản với List

- Phương thức list.index(obj): Trả về vị trí xuất hiện đầu tiên của obj trong list. Nếu không xuất hiện, trả về ngoại lệ (exception) – bị lỗi

```
aList = [123, 'xyz', 'hoang', 'abc'];

print("Chi muc cua xyz la:", aList.index('xyz'))
print("Chi muc cua hoang la:", aList.index('hoang'))
```

```
Chi muc cua xyz la: 1
Chi muc cua hoang la: 2
```

Các phương thức cơ bản với List

- Phương thức list.append(item) thêm một phần tử item vào cuối list.
- Ví dụ:

```
list1=[10,"hoang",'z']
print("Cac phan tu cua List la:", list1)
list1.append(10.45)
print("Cac phan tu cua List sau khi phu them la:", list1)
```

```
Cac phan tu cua List la: [10, 'hoang', 'z']
Cac phan tu cua List sau khi phu them la: [10, 'hoang', 'z', 10.45]
```

Các phương thức cơ bản với List

- Phương thức list.insert(index, obj): Chèn đối tượng obj vào trong list tại vị trí index đã cho

```
aList = [123, 'xyz', 'hoang', 'abc']
aList.insert( 3, 2015)
print("List sau khi da chen bao gom:", aList)
```

```
List sau khi da chen bao gom: [123, 'xyz', 'hoang', 2015, 'abc']
```

Các phương thức cơ bản với List

- Phương thức list.reverse(): Đảo ngược thứ tự các đối tượng trong list

```
aList = [123, 'xyz', 'hoang', 'abc', 'xyz'];
aList.reverse();
print("List:", aList)
```

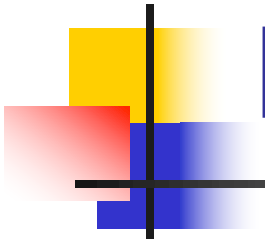
```
List: ['xyz', 'abc', 'hoang', 'xyz', 123]
```

Các phương thức cơ bản với List

- Phương thức list.sort([func]): Sắp xếp các đối tượng của list, sử dụng hàm so sánh func nếu được cung cấp

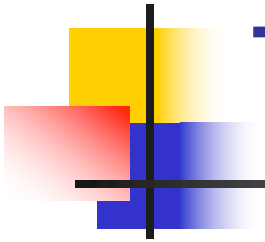
```
aList = [6, 2, 8, 3, 10, 9]
aList.sort()
print("List:", aList)
```

```
List: [2, 3, 6, 8, 9, 10]
```



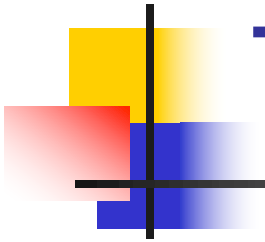
Nội dung

1. Kiểu dữ liệu Number
2. Kiểu dữ liệu String
3. Kiểu dữ liệu List
4. **Kiểu dữ liệu Tuple**
5. Kiểu dữ liệu Dictionary



Tuple

- Các thao tác cơ bản với Tuple
- Các hàm cơ bản với Tuple

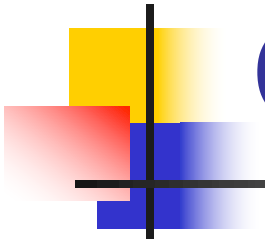


Tuple

- Tuple tương tự như List. Tuple dùng dấu ngoặc tròn, và không cập nhật (thay đổi) được Tuple
- Xử lý Tuple nhanh và an toàn hơn List

```
tup1 = ('vatly', 'hoahoc', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
tup3 = (7, 6, 5, 4, 3, 2, 1);

print(tup1[0])
print(tup2[1:5])
print(tup3[-3:-1])
```

```
vatly
(2, 3, 4, 5)
(3, 2)
```



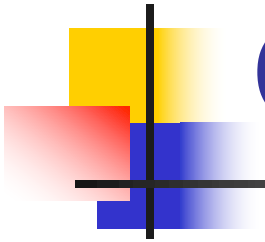
Các thao tác cơ bản với Tuple

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1,2,3) : print (x, end = ' ')	1 2 3	Iteration

Xóa phần tử trong Tuple

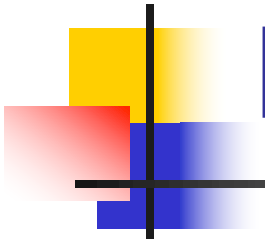
- Không thể xóa các phần tử trong Tuple. Chỉ có thể xóa toàn bộ Tuple với lệnh del
- Cũng không thể cập nhật giá trị một phần tử trong Tuple

```
data=(10,20,'hoang',40.6,'z')
print(data)
del data       #se xoa du lieu cua tuple
print(data)    #se hien thi mot error boi vi tuple da bi xoa


data=(10,20,30)
data[0]=100. #se hien thi error
print(data)
```



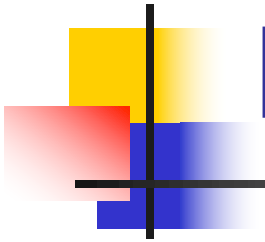
Các hàm cơ bản với Tuple

- Hàm len(tuple): Trả về độ dài của tuple
- Hàm max(tuple): Trả về item có giá trị lớn nhất từ một tuple đã cho
- Hàm min(tuple): Trả về item có giá trị nhỏ nhất từ một tuple đã cho
- Hàm tuple(seq): Chuyển đổi một dãy thành tuple



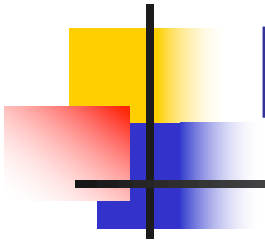
Nội dung

1. Kiểu dữ liệu Number
2. Kiểu dữ liệu String
3. Kiểu dữ liệu List
4. Kiểu dữ liệu Tuple
5. **Kiểu dữ liệu Dictionary**



Dictionary

- Truy cập các giá trị trong Dictionary
- Cập nhật Dictionary
- Xóa phần tử trong Dictionary
- Một số phương thức làm việc với Dictionary



Dictionary

- Dictionary là tập hợp các cặp key và value (không quan tâm tới thứ tự), được bao quanh bởi dấu ngoặc nhọn {}.
- Mỗi cặp key-value được xem là một item.
- Key phải là duy nhất, nhưng value có thể có giá trị bất kỳ
- Key phải là một kiểu dữ liệu không thay đổi (immutable) như chuỗi, số hoặc tuple
- Key và value được tách biệt bằng dấu hai chấm (:)
- Các item tách biệt nhau bởi dấu phảy (,).

```
data={100: 'Hoang', 101: 'Nam', 'a': 'Alice', 100: 'Hung'}
print(data)
```

```
{100: 'Hung', 101: 'Nam', 'a': 'Alice'}
```

Truy cập các giá trị trong Dictionary

- Các giá trị trong Dictionary có thể được truy cập thông qua các key của chúng. Cú pháp:

```
<ten_dictionary>[key]
```

- Ví dụ

```
data={100: 'Hoang', 101: 'Nam', 'a': 'Alice'}
print(data[100])
print(data[101])
print(data['a'])
```

```
Hoang
Nam
Alice
```

- Lưu ý: Truy cập key không có trong Dictionary sẽ bị lỗi

Cập nhật Dictionary

- Cập nhật value cho key cũ, thêm key-value mới:

```
data1={'Id':100, 'Ten':'Thanh', 'Nghenghiep':'Developer'}
data1['Nghenghiep'] = 'Manager'
data1['Tuoi'] = 25
print(data1)
```

```
{'Id': 100, 'Ten': 'Thanh', 'Nghenghiep': 'Manager', 'Tuoi': 25}
```

Xóa phần tử trong Dictionary

- Có thể xóa một phần tử hoặc xóa toàn bộ một Dictionary với lệnh del

```
del ten_dictionary[key]
del  ten_dictionary
```

- Ví dụ

```
data={100: 'Hoang', 101: 'Thanh', 102: 'Nam'}
del data[102]
print(data)
del data
print(data)   #se bi error vi Dictionary da bi xoa.
```

```
{100: 'Hoang', 101: 'Thanh'}
```

Một số phương thức làm việc với Dictionary

- Phương thức dict.clear(): Xóa tất cả phần tử của dict

```
dict = {'Ten': 'Hoang', 'Tuoi': 20};

print("Do dai ban dau:", len(dict))
dict.clear()
print("Do dai sau khi xoa:", len(dict))
```

```
Do dai ban dau: 2
Do dai sau khi xoa: 0
```

Một số phương thức làm việc với Dictionary

- Phương thức dict.get(key, default=None): Trả về giá trị của key đã cho. Nếu key không tồn tại, phương thức trả về giá trị default. Giá trị default là tùy chọn, nếu không chỉ ra, nhận giá trị mặc định là None

```
dict = {'Ten': 'Hoang', 'Tuoi': 27}

print(dict.get('Tuoi'))
print(dict.get('Gioitinh'))
print(dict.get('Gioitinh', "Never"))
```

```
27
None
Never
```

Một số phương thức làm việc với Dictionary

- Phương thức dict.items(): trả về tất cả các cặp (key-value) của một Dictionary
- Phương thức dict.keys(): trả về tất cả các key của một Dictionary
- Phương thức dict.values(): trả về tất cả các value của một Dictionary

```
dict = {'Ten': 'Hoang', 'Tuoi': 20}
keys = dict.keys()
print (keys)
print (list(keys))
```

```
dict_keys(['Ten', 'Tuoi'])
['Ten', 'Tuoi']
```