LIST

Nội Dung

- Cách tạo List
- Các hàm và toán tử trên List

Tạo List

- **List Rỗng:** **[]** hoặc dùng **list()**

```
print("Two standard ways to create an empty list:")
a = [ ]
b = list()
print(type(a), len(a), a)
print(type(b), len(b), b) print(a == b)
```

```
Two standard ways to create an empty list:
<class 'list'> 0 []
<class 'list'> 0 []
True
```

Tạo List

- **List có một phần tử**

```
a = [ "hello" ]
b = [ 42 ]
print(type(a), len(a), a)
print(type(b), len(b), b)
print(a == b)
```

```
<class 'list'> 1 ['hello']
<class 'list'> 1 [42]
False
```

Tạo List

- **List có nhiều phần tử:** các phần tử cách nhau bởi dấu phẩy

```
a = [2, 3, 5, 7]
b = list(range(5))
c = ["mixed types", True, 42]
print(type(a), len(a), a)
print(type(b), len(b), b)
print(type(c), len(c), c)
```

```
<class 'list'> 4 [2, 3, 5, 7]
<class 'list'> 5 [0, 1, 2, 3, 4]
<class 'list'> 3 ['mixed types', True, 42]
```

Tạo List

```
n = 10
a = [0] * n
b = list(range(n))
print(type(a), len(a), a)
print(type(b), len(b), b)
```

```
<class 'list'> 10 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
<class 'list'> 10 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Các Hàm và Toán Tử Thông Dụng

- **len(), min(), max(), sum()**

```
a = [ 2, 3, 5, 2 ]
print("a = ", a)
print("len =", len(a))
print("min =", min(a))
print("max =", max(a))
print("sum =", sum(a))
```

```
a =  [2, 3, 5, 2]
len = 4
min = 2
max = 5
sum = 12
```

Các Hàm và Toán Tử Thông Dụng

- **Chỉ số trong List** : dùng toán tử []

```
a = [2, 3, 5, 7, 11, 13]
print("a =", a)
print("a[0] =", a[0])
print("a[2] =", a[2])
# Chi so am
print("a[-1] =", a[-1])
print("a[-3] =", a[-3])
# mot khoang nam trong list
print("a[0:2] =", a[0:2])
print("a[1:4] =", a[1:4])
print("a[1:6:2] =", a[1:6:2])
```

```
a          = [2, 3, 5, 7, 11, 13]
a[0]       = 2
a[2]       = 5
a[-1]      = 13
a[-3]      = 7
a[0:2]     = [2, 3]
a[1:4]     = [3, 5, 7]
a[1:6:2]   = [3, 7, 13]
```

Các Hàm và Toán Tử Thông Dụng

- **List Aliases: Alias** là khả năng mà tại 1 ô nhớ có nhiều đối tượng cùng trỏ tới

```
# Tao mot list a
a = [ 2, 3, 5, 7 ]
# Tao mot bi danh den list a
b = a
# Co hai tham chieu cung mot list
a[0] = 42
b[1] = 99
print(a)
print(b)
```

```
[42, 99, 5, 7]
[42, 99, 5, 7]
```

```
a = [ 2, 3, 5, 7 ]
b = a
c = [ 2, 3, 5, 7 ]
print("initially:")
print(" a==b :", a==b)
print(" a==c :", a==c)
print(" a is b:", a is b)
print(" a is c:", a is c)
a[0] = 42
print("After changing a[0] to 42")
print(" a=",a)
print(" b=",b)
print(" c=",c)
print(" a==b :", a==b)
print(" a==c :", a==c)
print(" a is b:", a is b)
print(" a is c:", a is c)
```

Thông Dụng

```
  a==b  : True
  a==c  : True
  a is b: True
  a is c: False
After changing a[0] to 42
  a= [42, 3, 5, 7]
  b= [42, 3, 5, 7]
  c= [2, 3, 5, 7]
  a==b  : True
  a==c  : False
  a is b: True
  a is c: False
```

Các Hàm và Toán Tử Thông Dụng

- **Tìm phần tử trong list: in** và **not in**

```
a = [ 2, 3, 5, 2, 6, 2, 2, 7 ]
print("a =", a)
print("2 in a =", (2 in a))
print("4 in a =", (4 in a))
```

```
a = [ 2, 3, 5, 2, 6, 2, 2, 7 ]
print("a =", a)
print("2 not in a =", (2 not in a))
print("4 not in a =", (4 not in a))
```

```
a      = [2, 3, 5, 2, 6, 2, 2, 7]
2 in a = True
4 in a = False
```

```
a          = [2, 3, 5, 2, 6, 2, 2, 7]
2 not in a = False
4 not in a = True
```

Các Hàm và Toán Tử Thông Dụng

- **Đếm số lần xuất hiện: list.count(item)**

```
a = [ 2, 3, 5, 2, 6, 2, 2, 7 ]
print("a =", a)
print("a.count(1) =", a.count(1))
print("a.count(2) =", a.count(2))
print("a.count(3) =", a.count(3))
```

```
a          = [2, 3, 5, 2, 6, 2, 2, 7]
a.count(1) = 0
a.count(2) = 4
a.count(3) = 1
```

Các Hàm và Toán Tử Thông Dụng

- **Tìm chỉ số của một phần tử:**

list.index(item) và **list.index(item, start)**

```
a = [ 2, 3, 5, 2, 6, 2, 2, 7 ]
print("a =", a)
print("a.index(6) =", a.index(6))
print("a.index(2) =", a.index(2))
print("a.index(2,1) =", a.index(2,1))
print("a.index(2,4) =", a.index(2,4))
print("a.index(2,7) =", a.index(2,7))
```

```
a             = [2, 3, 5, 2, 6, 2, 2, 7]
a.index(6)    = 4
a.index(2)    = 0
a.index(2,1) = 3
a.index(2,4) = 5
Traceback (most recent call last):
ValueError: 2 is not in list
```

Các Hàm và Toán Tử Thông Dụng

- **Thêm phần tử hoặc một list vào list**: khi thêm sẽ thay đổi list hoặc tạo list mới.
- Thêm **một phần tử** dùng: **list.append(item)**

```
a = [ 2, 3 ]
a.append(7)

print(a)
```

```
[2, 3, 7]
```

- Thêm **một list vào một list**: **list += list2**

```
a = [ 2, 3 ]
a += [ 11, 13 ]
a += [0]

print(a)
```

```
[2, 3, 11, 13, 0]
```

Các Hàm và Toán Tử Thông Dụng

- Thêm **một list** dùng **list.extend(list2)**

```
a = [ 2, 3 ]
a.extend([ 17, 19 ])
print(a)
```

```
[2, 3, 17, 19]
```

- Thêm **một phần tử tại vị trí cho trước**: dùng insert()

```
a = [ 2, 3, 5, 7, 11 ]
a.insert(2, 42) # at index 2, insert 42
print(a)
```

```
[2, 3, 42, 5, 7, 11]
```

Các Hàm và Toán Tử Thông Dụng

- **Thêm phần tử hoặc list bằng cách tạo list mới.**

```
a = [ 2, 3 ]
b = a + [ 13, 17 ]
print(a)
print(b)
```

```
[2, 3]
[2, 3, 13, 17]
```

```
a = [ 2, 3 ]
b = a[:2] + [5] + a[2:]
print(a)
print(b)
```

```
[2, 3]
[2, 3, 5]
```

```
print("Destructive:")
a = [ 2, 3 ]
b = a
a += [ 4 ]
print(a)
print(b)
print("Non-Destructive:")
a = [ 2, 3 ]
b = a
a = a + [ 4 ]
print(a)
print(b)
```

```
Destructive:
[2, 3, 4]
[2, 3, 4]
Non-Destructive:
[2, 3, 4]
[2, 3]
```

Các Hàm và Toán Tử Thông Dụng

- **Thay đổi các phần tử trong list**

```
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
print(letters)
# thay the
letters[2:5] = ['C', 'D', 'E']
print(letters)
# xoa
letters[2:5] = []
print(letters)
# xóa list bằng cách thay tất cả các phần tử bằng một list rỗng
letters[:] = []
print(letters)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g']
['a', 'b', 'C', 'D', 'E', 'f', 'g']
['a', 'b', 'f', 'g']
[]
```

Các Hàm và Toán Tử Thông Dụng

- Xóa phần tử.
- Xóa **một phần tử** dùng **list.remove(item)**

```
a = [ 2, 3, 5, 3, 7, 6, 5, 11, 13 ]
print("a =", a)
a.remove(5)
print("After a.remove(5), a=", a)
a.remove(5)
print("After another a.remove(5), a=", a)
```

```
a = [2, 3, 5, 3, 7, 6, 5, 11, 13]
After a.remove(5), a= [2, 3, 3, 7, 6, 5, 11, 13]
After another a.remove(5), a= [2, 3, 3, 7, 6, 11, 13]
```

Các Hàm và Toán Tử Thông Dụng

- Xóa **một phần tử dùng chỉ số : list.pop(index)**

```
a = [ 2, 3, 4, 5, 6, 7, 8 ]
print("a =", a)
item = a.pop(3)
print("After item = a.pop(3)")
print(" item =", item)
print(" a =", a)
item = a.pop(3)
print("After another item = a.pop(3)")
print(" item =", item)
print(" a =", a)
# Xoa phan tu cuoi cung cua lít dung list.pop()
item = a.pop()
print("After item = a.pop()")
print(" item =", item)
print(" a =", a)
```

```
a = [2, 3, 4, 5, 6, 7, 8]
After item = a.pop(3)
    item = 5
    a = [2, 3, 4, 6, 7, 8]
After another item = a.pop(3)
    item = 6
    a = [2, 3, 4, 7, 8]
After item = a.pop()
    item = 8
    a = [2, 3, 4, 7]
```

Các Hàm và Toán Tử Thông Dụng

- Xóa **phần từ dùng toán tử del**

```
a = [ 2, 3, 4, 5, 6, 7, 8 ]
del a[2:4]
print("a =", a)
```

```
a = [2, 3, 6, 7, 8]
```

- **Phương thức clear() cũng được dùng để làm rỗng một list**

```
a = [ 2, 3, 4, 5, 6, 7, 8 ]
a.clear()
print("a =", a)
```

```
a = []
```

Các Hàm và Toán Tử Thông Dụng

- **Xóa phần từ dựa vào chỉ số bằng cách tạo list mới**

```
a = [ 2, 3, 5, 3, 7, 5, 11, 13 ]
print("a =", a)
b = a[:2] + a[3:]
print("After b = a[:2] + a[3:]")
print(" a =", a)
print(" b =", b)
```

```
a = [2, 3, 5, 3, 7, 5, 11, 13]
After b = a[:2] + a[3:]
    a = [2, 3, 5, 3, 7, 5, 11, 13]
    b = [2, 3, 3, 7, 5, 11, 13]
```

Các Hàm và Toán Tử Thông Dụng

- Hoán đổi các phần tử (swapping)
- **Failed swap**

```
a = [ 2, 3, 5, 7 ]
print("a =", a)
a[0] = a[1]
a[1] = a[0]
print("After failed swap of a[0] and a[1]:")
print(" a=",a)
```

```
a = [2, 3, 5, 7]
After failed swap of a[0] and a[1]:
   a= [3, 3, 5, 7]
```

Các Hàm và Toán Tử Thông Dụng

- **Swap dùng biến temp**

```
a = [ 2, 3, 5, 7 ]
print("a =", a)
temp = a[0]
a[0] = a[1]
a[1] = temp
print("After swapping a[0] and a[1]:")
print(" a=",a)
```

```
a = [2, 3, 5, 7]
After swapping a[0] and a[1]:
    a= [3, 2, 5, 7]
```

Các Hàm và Toán Tử Thông Dụng

- Swap dùng **parallel assignment**

```
a = [ 2, 3, 5, 7 ]
print("a =", a)
a[0],a[1] = a[1],a[0]
print("After swapping a[0] and a[1]:")
print(" a=",a)
```

```
a = [2, 3, 5, 7]
After swapping a[0] and a[1]:
    a= [3, 2, 5, 7]
```

Các Hàm và Toán Tử Thông Dụng

- Vòng lặp for trong list
- Vòng lặp **dùng phần tử** trong list: **for item in list**

```
a = [ 2, 3, 5, 7 ]
print("Here are the items in a:")
for pt in a:
      print(pt)
```

```
Here are the items in a:
2
3
5
7
```

Các Hàm và Toán Tử Thông Dụng

- Vòng lặp **dùng chỉ số** trong list : **for index in range(len(list))**

```
a = [ 2, 3, 5, 7 ]
print("Here are the items in a with their indexes:")
for i in range(len(a)):
      print("a[", i, "] =", a[i])
```

```
Here are the items in a with their indexes:
a[ 0 ] = 2
a[ 1 ] = 3
a[ 2 ] = 5
a[ 3 ] = 7
```

Các Hàm và Toán Tử Thông Dụng

- **Duyệt ngược** list **dùng chỉ số**

```
a = [ 2, 3, 5, 7 ]
print("And here are the items in reverse:")
for index in range(len(a)):
        revIndex = len(a)-1-index
        print("a[", revIndex, "] =", a[revIndex])
```

```
And here are the items in reverse:
a[ 3 ] = 7
a[ 2 ] = 5
a[ 1 ] = 3
a[ 0 ] = 2
```

- **Duyệt ngược** dùng hàm **reversed()**

```
a = [ 2, 3, 5, 7 ]
print("And here are the items in reverse:")
for item in reversed(a):
        print(item)
print(a)
```

```
And here are the items in reverse:
7
5
3
2
[2, 3, 5, 7]
```

Các Hàm và Toán Tử Thông Dụng

- So sánh lists :

```
# Create some lists
a = [ 2, 3, 5, 3, 7 ]
b = [ 2, 3, 5, 3, 7 ] # same as a
c = [ 2, 3, 5, 3, 8 ] # differs in last elem
d = [ 2, 3, 5 ] # prefix of a
print("a =", a)
print("b =", b)
print("c =", c)
print("d =", d)
print("------------------")
print("a == b", (a == b))
print("a == c", (a == c))
print("a != b", (a != b))
print("a != c", (a != c))
print("------------------")
print("a < c", (a < c))
print("a < d", (a < d))
```

```
a = [2, 3, 5, 3, 7]
b = [2, 3, 5, 3, 7]
c = [2, 3, 5, 3, 8]
d = [2, 3, 5]
------------------
a == b True
a == c False
a != b False
a != c True
------------------
a < c True
a < d False
```

```python
# Create some lists
a = [ 2, 3]
b = [ 2, 3]
c = [ 2, 4]
d = [ 2]
e = [ 3]
f = [ 2 , 3 ,5]
print("--------------------")
print("a < c", (a < c))
print("a < d", (a < d))
print("a <= b", (a <= b))
print("a < b", (a < b))
print("a < d", (a < d))
print("a < e", (a < e))
print("a < f", (a < f))
```

```
--------------------
a < c True
a < d False
a <= b True
a < b False
a < d False
a < e True
a < f True
```

Các Hàm và Toán Tử Thông Dụng

- **Copying list vs. List Aliasses**

```
import copy
# Create a list
a = [ 2, 3 ]
# Try to copy it
b = a # Error! Not a copy, but an alias
c = copy.copy(a) # Ok
print("At first...")
print(" a =", a)
print(" b =", b)
print(" c =", c)
# Now modify a[0]
a[0] = 42
print("But after a[0] = 42")
print(" a =", a)
print(" b =", b)
print(" c =", c)
```

```
At first...
    a = [2, 3]
    b = [2, 3]
    c = [2, 3]
But after a[0] = 42
    a = [42, 3]
    b = [42, 3]
    c = [2, 3]
```

Các Hàm và Toán Tử Thông Dụng

- **Các cách sao chép list**

```
import copy
a = [2, 3]
b = copy.copy(a)
c = a[:]
d = a + [ ]
e = list(a)
f = sorted(a)
a[0] = 42
print(a, b, c, d, e, f)
```

```
[42, 3] [2, 3] [2, 3] [2, 3] [2, 3] [2, 3]
```

Các Hàm và Toán Tử Thông Dụng

- Sắp xếp trên list
- **Sắp xếp và thay đổi list** dùng **list.sort() :**

iterable.sort(key=None, reverse=False)

```
a = [ 7, 2, 5, 3, 5, 11, 7 ]
b = list(a) # copy of a
print("At first, a =", a)
a.sort()
print("After a.sort(), a =",a)
############################
print("At first, b =", b)
b.sort(reverse=True)
print("After b.sort(reverse=True), b =",b)
```

```
At first, a = [7, 2, 5, 3, 5, 11, 7]
After a.sort(), a = [2, 3, 5, 5, 7, 7, 11]
At first, b = [7, 2, 5, 3, 5, 11, 7]
After b.sort(reverse=True), b = [11, 7, 7, 5, 5, 3, 2]
```

Các Hàm và Toán Tử Thông Dụng

- **Sắp xếp nhưng không thay đổi list** (tạo ra list mới) dùng **sorted(list)**

sorted(iterable, key=None, reverse=False)

```
a = [ 7, 2, 5, 3, 5, 11, 7 ]
print("At first")
print(" a =", a)
b = sorted(a)
c = sorted(a,reverse=True)
print("After b = sorted(a)")
print(" a =", a)
print(" b =", b)
print(" c =", c)
```

```
At first
   a = [7, 2, 5, 3, 5, 11, 7]
After b = sorted(a)
   a = [7, 2, 5, 3, 5, 11, 7]
   b = [2, 3, 5, 5, 7, 7, 11]
   c = [11, 7, 7, 5, 5, 3, 2]
```

Các Hàm và Toán Tử Thông Dụng

- Sắp xếp với key function
- **Dùng abs()**

```
a = [ 10, 2, -5, 8, -3, 7, 1 ]
print(sorted(a))

print(sorted(a, key=abs))
```

```
[-5, -3, 1, 2, 7, 8, 10]
[1, 2, -3, -5, 7, 8, 10]
```

Các Hàm và Toán Tử Thông Dụng

- **Sort dựa vào chiều dài của chuỗi**

```
a = [ 'a', 'ab', 'aab', 'ac', 'abccc']
print(sorted(a))
```

```
['a', 'aab', 'ab', 'abccc', 'ac']
```

```
####################
def mylensort(a):
        return len(a)

print(sorted(a, key=mylensort))
```

```
['a', 'ac', 'ab', 'aab', 'abccc']
```

Các Hàm và Toán Tử Thông Dụng

- So sanh thời gian và không gian bộ nhớ sử dụng giữa sort() và sorted()

https://viblo.asia/p/so-sanh-listsort-voi-sortedlist-trong-python-gDVK22MrKLj

Các Hàm và Toán Tử Thông Dụng

- **List và Function:** list được dùng như input của một hàm

```
def countOdds(a):
        count = 0
        for item in a:
                if (item % 2 == 1):
                        count += 1
        return count


print(countOdds([2, 3, 7, 8, 21, 23, 24]))
```

```
4
```

Các Hàm và Toán Tử Thông Dụng

- **Dùng hàm để thay đổi các giá trị của list**

```
def fill(a, value):
      for i in range(len(a)):
            a[i] = value
a = [1, 2, 3, 4, 5]
print("At first, a =", a)
fill(a, 42)
print("After fill(a, 42), a =", a)
```

```
At first, a = [1, 2, 3, 4, 5]
After fill(a, 42), a = [42, 42, 42, 42, 42]
```

Các Hàm và Toán Tử Thông Dụng

- **List comprehension (Cách tạo list mới ngắn gọn):** là một biểu thức đi kèm với lệnh for được đặt trong cặp dấu ngoặc vuông [].

```
cub3 = [3 ** x for x in range(9)]
# Output: [1, 3, 9, 27, 81, 243, 729, 2187, 6561]
print(cub3)
```

Code trên tương đương với:

```
cub3 = []
for x in range (9):
        cub3.append(3**x)
print(cub3)
```

Các Hàm và Toán Tử Thông Dụng

```
cub3 = [3 ** x for x in range(9) if x > 4]
 # Output: [243, 729, 2187, 6561]
print(cub3)

so_le = [x for x in range (18) if x % 2 == 1]
# Output: [1, 3, 5, 7, 9, 11, 13, 15, 17]
print(so_le)

noi_list = [x+y for x in ['Ngôn ngữ ','Lập trình '] for y in ['Python','C++']]
# Output: ['Ngôn ngữ Python', 'Ngôn ngữ C++', 'Lập trình Python', 'Lập trình C++']
print(noi_list)
```

Các Hàm và Toán Tử Thông Dụng

- **Converting Between Lists and Strings**

```
# use list(s) to convert a string to a list of characters
a = list("wahoo!")
print(a) # prints: ['w', 'a', 'h', 'o', 'o', '!']

a = "How are you doing today?".split(" ")
print(a) # prints ['How', 'are', 'you', 'doing', 'today?']

a = ["parsley", " ", "is", " ", "gharsley"]
s = "".join(a)
print(s) # prints: parsley is gharsley
```

2D LIST

Nội Dung

- Cách tạo 2D List
- Các hàm và toán tử trên 2D List

Tạo 2D List

- **Cấp phát tĩnh**

```
# Tao 2D list voi cac gia tri co dinh
a = [ [ 2, 3, 4 ] , [ 5, 6, 7 ] ]
print(a)
```

```
[[2, 3, 4], [5, 6, 7]]
```

Tạo 2D List

- **Cấp phát động**
  - a) **Wrong: Cannot use ***

```
rows = 3
cols = 2
a = [ [0] * cols ] * rows # Error:
# Chỉ tao duy nhat mot dong (unique row), phan con lai la aliases!
print("This SEEMS ok. At first:")
print(" a =", a)
a[0][0] = 42
print("But see what happens after a[0][0]=42")
print(" a =", a)
```

```
This SEEMS ok.  At first:
   a = [[0, 0], [0, 0], [0, 0]]
But see what happens after a[0][0]=42
   a = [[42, 0], [42, 0], [42, 0]]
```

Tạo 2D List

- **Cấp phát động**
- b) **Mở rộng mỗi dòng**

```
rows = 3
cols = 2
a=[]
for row in range(rows):
      a += [[0]*cols]
print("This IS ok. At first:")
print(" a =", a)
a[0][0] = 42
print("And now see what happens after a[0][0]=42")
print(" a =", a)
```

```
This IS ok.  At first:
   a = [[0, 0], [0, 0], [0, 0]]
And now see what happens after a[0][0]=42
   a = [[42, 0], [0, 0], [0, 0]]
```

Tạo 2D List

- **Cấp phát động**
- c) Dùng **list comprehension**

```
This IS ok.  At first:
   a = [[0, 0], [0, 0], [0, 0]]
And now see what happens after a[0][0]=42
   a = [[42, 0], [0, 0], [0, 0]]
```

```
rows = 3
cols = 2
a = [ ([0] * cols) for row in range(rows) ]
print("This IS ok. At first:")
print(" a =", a)
a[0][0] = 42
print("And now see what happens after a[0][0]=42")
print(" a =", a)
```

Các hàm và toán tử trên 2D List

- **Tìm số chiều của 2D List**: dùng **len()**

```
a = [ [ 2, 3, 5] , [ 1, 4, 7 ] ]
print("a = ", a)
# Now find its dimensions
rows = len(a)
cols = len(a[0])
print("rows =", rows)
print("cols =", cols)
```

```
a =  [[2, 3, 5], [1, 4, 7]]
rows = 2
cols = 3
```

Các hàm và toán tử trên 2D List

- **Vòng lặp**

```
a = [ [ 2, 3, 5] , [ 1, 4, 7 ] ]
print("Before: a =", a)
rows = len(a)
cols = len(a[0])
for row in range(rows):
      for col in range(cols):
            a[row][col] += 1
print("After: a =", a)
```

```
Before: a = [[2, 3, 5], [1, 4, 7]]
After:  a = [[3, 4, 6], [2, 5, 8]]
```

Các hàm và toán tử trên 2D List

- **Truy cập 2D List bằng dòng hoặc cột**
  - a) **Truy cập một dòng**

```
# alias (not a copy!); (no new list created)
a = [ [ 1, 2, 3 ] , [ 4, 5, 6 ] ]
row = 1
rowList = a[row]
print(rowList)
rowList[0] = 100
print(rowList)
print(a)
```

```
[4, 5, 6]
[100, 5, 6]
[[1, 2, 3], [100, 5, 6]]
```

Các hàm và toán tử trên 2D List

- **Truy cập 2D List bằng dòng hoặc cột**
- **b) Truy cập một cột**

```
# copy (not an alias!); (new list created)
a = [ [ 1, 2, 3 ] , [ 4, 5, 6 ] ]
col = 1
colList = [ ]
for i in range(len(a)):
      colList += [ a[i][col] ]
print(colList)
```

```
[2, 5]
```

```
# co the dung list comprehension
colList = [ a[i][col] for i in range(len(a)) ]
```

Các hàm và toán tử trên 2D List

- **Số lượng phần tử mỗi dòng có thể khác nhau**

```
# 2d lists do not have to be rectangular
a = [ [ 1, 2, 3 ] ,
      [ 4, 5 ],
      [ 6 ],
      [ 7, 8, 9, 10 ] ]
rows = len(a)
for row in range(rows):
      cols = len(a[row]) # now cols depends on each row
      print("Row", row, "has", cols, "columns: ", end="")
      for col in range(cols):
            print(a[row][col], " ", end="")
      print()
```

```
Row 0 has 3 columns: 1  2  3
Row 1 has 2 columns: 4  5
Row 2 has 1 columns: 6
Row 3 has 4 columns: 7  8  9  10
```

Dictionary

Nội Dung

- Cách tạo Dictionary
- Các hàm và toán tử trên Dictionary

Cách Tạo Dictionary

- Từ điển (Dictionary) được dùng để ánh xạ hoặc liên kết dữ liệu bạn cần lưu trữ (**value**) và khóa (**key**) bạn cần để lấy ra dữ liệu đó.
- Từ điển trong Python được định nghĩa gồm hai thành phần là khóa (key) và giá trị (value).
  - Khóa là đối tượng mang tính duy nhất.
  - Giá trị được lưu có thể là 1D List hoặc 2D List, một chuỗi, một số, một đối tượng bất kỳ trong python ...

```
Dict = {'Name': 'Tim', 'Age': 18, ..}
```

Cách Tạo Dictionary

- **Tạo một từ điển rỗng**: dùng dict() hoặc { }

```
d = dict()
print(d) # prints {}
```

```
d = { }
print(d) # prints {}
```

- **Tạo từ điển từ một list các cặp (key, value)**

```
pairs = [("cow", 5), ("dog", 98), ("cat", 1)]
d = dict(pairs)
print(d)
```

```
{'cow': 5, 'dog': 98, 'cat': 1}
```

Cách Tạo Dictionary

- **Tạo từ điển bằng cấp phát tĩnh**

```
d = { "cow":5, "dog":98, "cat":1 }
print(d)
```

```
{'cow': 5, 'dog': 98, 'cat': 1}
```

- **Các ví dụ khác**:

```
d1 = {1: 'Quantrimang.com',2: 'Công nghệ'}
d2 = {'tên': 'QTM', 1: [1, 3, 5]}
d3 = dict({1:'apple', 2:'ball'})
d4 = dict([(1,'QTM'), (2,'CN')])
```

Cách Tạo Dictionary

- Các tính chất của từ điển:
- Cần lưu ý khi sử dụng khóa trong từ điển:
  - Một khóa không thể xuất hiện hai lần (khóa không được trùng nhau).
  - Giá trị được lưu trong từ điển có thể thuộc bất kỳ kiểu nào trong khi khóa phải là kiểu bất biến như số, tuple hoặc chuỗi.
  - Khóa sử dụng trong từ điển có phân biệt chữ hoa chữ thường - Cùng tên khóa nhưng tên khóa viết hoa và viết thường sẽ được coi là các khóa khác nhau.

Cách Tạo Dictionary

- Các tính chất của từ điển:
- **Từ điển ánh xạ một khóa đến một giá trị**

```
ages = dict()
key = “tom"
value = 38
ages[key] = value # “tom" is the key, 38 is the value
print(ages)
print(ages[key])
```

```
{'tom': 38}
38
```

Cách Tạo Dictionary

- **Các từ khóa biểu diễn theo kiểu tập hợp**
  - **Không có thứ tự**

```
d = dict()
d[2] = 100
d[4] = 200
d[8] = 300

print(d) # unpredictable order
```

```
{2: 100, 4: 200, 8: 300}
```

  - **Duy nhất**

```
d = dict()
d[2] = 100
d[2] = 200
d[2] = 400

print(d) # { 2:400 }
```

```
{2: 400}
```

Cách Tạo Dictionary

- **Các từ khóa không thể thay đổi**

```
d = dict()
a = [1] # lists are mutable, so...
d[a] = 42 # Error: unhashable type: 'list'
```

- **Các giá trị có thể thay đổi**

```
d = dict()
a = [1,2]
d["fred"] = a
print(d)
print(d["fred"])
a += [3]
print(d["fred"])
# but keys may not be mutable
d[a] = 42 # TypeError: unhashable type: 'list'
```

```
{'fred': [1, 2]}
[1, 2]
[1, 2, 3]
Traceback (most recent call last):
TypeError: unhashable type: 'list'
```

Các Hàm và Toán Tử Trên Dictionary

- **Các phép toán trên từ điển**
- **Tìm chiều dài: len()**

```
d = { 1:[1,2,3,4,5], 2:"abcd" }
print(len(d))  # 2
```

- **Tạo bản sao : copy()**

```
d1 = { 1:"a" }
d2 = d1.copy()
d1[2] = "b"
print(d1)
print(d2)
```

```
{1: 'a', 2: 'b'}
{1: 'a'}
```

Các Hàm và Toán Tử Trên Dictionary

- **Xóa tất cả phần tử trong từ điển: clear()**

```
d = { 1:"a", 2:"b" }
d.clear()
print(d, len(d))
```

```
{} 0
```

- **Vòng lặp trên từ điển:**

```
d = { 1:"a", 2:"b" }
for key in d:
    print(key, d[key])
```

```
1 a
2 b
```

Các Hàm và Toán Tử Trên Dictionary

- **Các phép toán trên từ điển và khóa**
- **Toán tử in và not in:**

```
d = { 1:"a", 2:"b" }
print(0 in d)  # False
print(1 in d)  # True
print("a" in d) # surprised?  False
```

```
d = { 1:"a", 2:"b" }
print(0 not in d)  # True
print(1 not in d)  # False
print("a" not in d) # True
```

Các Hàm và Toán Tử Trên Dictionary

- **Toán tử [key] :** trả về giá trị tương ứng với khóa

```
d = { 1:"a", 2:"b" }
print(d[1])  # a
print(d[3]) # crash!
```

- **Toán tử gán** = : gán giá trị cho từ khóa tương ứng

```
d = { 1:"a", 2:"b" }
print(d[1])   # a
d[1] = 42
print(d[1])   # 42
```

Các Hàm và Toán Tử Trên Dictionary

- **Hàm get()**: get(key,default) trả về giá trị tương ứng với key hoặc nếu không tồn tại trả về default (None nếu default không được dùng)

```
d = { 1:"a", 2:"b" }
print(d.get(1)) # tuong duong d[1]
print(d.get(1, 42)) # default không được dùng
print(d.get(0)) # doesn't crash! Không bị lỗi
print(d.get(0, 42)) # default được dùng
```

```
a
a
None
42
```

Các Hàm và Toán Tử Trên Dictionary

- **Toán tử del:** xóa một từ khóa khỏi từ điển

```
d = { 1:"a", 2:"b" }
print(1 in d)  # True
del d[1]
print(1 in d)  # False
del d[1] # crash! ERROR
```

Các Hàm và Toán Tử Trên Dictionary

- **Thay đổi giá trị và thêm phần tử cho từ điển**

```
d2 = {1: 'Quantrimang.com','quantrimang': 'Công nghệ'}
d2['quantrimang'] = 'Quản trị mạng'
#output: {1: 'Quantrimang.com', 'quantrimang': 'Quản trị mạng'}
 print(d2)

d2[2] = 'Python‘
 #output: {1: 'Quantrimang.com', 'quantrimang': 'Quản trị mạng', 2: 'Python'}
print(d2)
```

Các Hàm và Toán Tử Trên Dictionary

- Trả về danh sách đối tượng dùng **items()** hoặc từ khóa dùng **keys()**

```
d = { 1:"a", 2:"b" }
print(d)
print(list(d.items())) # [(1, 'a'), (2, 'b')]
print(list(d.keys())) # [1, 2]
```

```
{1: 'a', 2: 'b'}
[(1, 'a'), (2, 'b')]
[1, 2]
```

Các Hàm và Toán Tử Trên Dictionary

- **Sắp xếp các từ khóa trong từ điển**

```
Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}
Students = list(Dict.keys())
Students.sort()
print(Students) # ['Charlie', 'Robert', 'Tiffany', 'Tim']
for S in Students:
    print(":".join((S,str(Dict[S]))))
```

```
['Charlie', 'Robert', 'Tiffany', 'Tim']
Charlie:12
Robert:25
Tiffany:22
Tim:18
```

Các Hàm và Toán Tử Trên Dictionary

- **Cập nhật từ điển dùng: update()**

```
d1 = { 1:"a", 2:"b" }
d2 = { 2:"c", 3:"d" }
d1.update(d2)
d2[4] = "e"
print(d1)
print(d2)
```

```
{1: 'a', 2: 'c', 3: 'd'}
{2: 'c', 3: 'd', 4: 'e'}
```

Các Hàm và Toán Tử Trên Dictionary

- **Dictionary comprehension**

```
lap_phuong = {x: x*x*x for x in range(6)}
 # Output: {0: 0, 1: 1, 2: 8, 3: 27, 4: 64, 5: 125}
print(lap_phuong)
```

```
lap_phuong_chan = {x: x*x*x for x in range (10) if x%2==0}
# output: {0: 0, 2: 8, 4: 64, 6: 216, 8: 512}
print(lap_phuong_chan)
```

Chuyển Đổi Giữa Các Kiểu Dữ Liệu

```
print(float(11))  # 11.0
print(int(18.6))  # 18
print(set([2,4,6])) # set({2, 4, 6})
print(tuple({3,5,7})) # (3 , 5, 7)
print(list('hello')) # ['h', 'e', 'l', 'l', 'o']
print(dict([[2,4],[1,3]])) # {1: 3, 2: 4}
print(dict([(3,9),(4,16)])) # {3: 9, 4: 16}
```

Tuples

Nội Dung

- Cách tạo Tuple
- Các hàm và toán tử trên Tuple

Cách Tạo Tuple

- **Tuple** (hay còn gọi là danh sách bất biến - **Immutable List**): là một chuỗi các phần tử có thứ tự giống như list. Sự khác biệt giữa list và tuple là chúng ta **không thể thay đổi các phần tử trong tuple khi đã gán**, nhưng trong list thì các phần tử có thể thay đổi.

```
t = (1, 2, 3)
print(type(t), len(t), t)
a = [1, 2, 3]
t = tuple(a)
print(type(t), len(t), t)
```

```
<class 'tuple'> 3 (1, 2, 3)
<class 'tuple'> 3 (1, 2, 3)
```

Cách Tạo Tuple

- **Không thể thay đổi giá trị trong tuple khi đã gán** (tương tự như chuỗi)

```
t = (1, 2, 3)
print(t[0])
t[0] = 42 # crash!
print(t[0])
```

```
1
Traceback (most recent call last):
TypeError: 'tuple' object does not support item assignment
```

Cách Tạo Tuple

```
t = (10, "quan tri mang", 2j)
#t[0:2] = (10, 'quan tri mang')
print("t[0:2] = ", t[0:2])
# Tuple rỗng
# Output: ()
my_tuple = ()
print(my_tuple)
# tuple số nguyên
# Output: (2, 4, 16, 256)
my_tuple = (2, 4, 16, 256)
print(my_tuple)
```

```
# tuple có nhiều kiểu dữ liệu
# Output: (10, "Quantrimang.com", 3.5)
my_tuple = (10, "Quantrimang.com", 3.5)
print(my_tuple)
# tuple lồng nhau
# Output: ("QTM", [2, 4, 6], (3, 5, 7))
my_tuple = ("QTM", [2, 4, 6], (3, 5, 7))
print(my_tuple)
# tuple có thể được tạo mà không cần dấu ()
# còn gọi là đóng gói tuple
 # Output: (10, "Quantrimang.com", 3.5)
my_tuple = 10, "Quantrimang.com", 3.5
print(my_tuple)
# mở gói (unpacking) tuple cũng có thể làm được
# Output:
# 10
# Quantrimang.com
# 3.5
a, b, c = my_tuple
print(a)
print(b)
print(c)
```

Cách Tạo Tuple

- Tuple có những lợi thế nhất định so với list:
  - Tuple chứa những phần tử không thay đổi, có thể được sử dụng như key cho dictionary. Với list, điều này không thể làm được.
  - Nếu có dữ liệu không thay đổi việc triển khai nó như một tuple sẽ đảm bảo rằng dữ liệu đó **được bảo vệ chống ghi** (write-protected).

Các Hàm và Toán Tử Trên Tuple

- **Parallel (tuple) assignment**

```
(x, y) = (1, 2)
print(x)
print(y)
(x, y) = (y, x)
print(x)
print(y)
```

```
1
2
2
1
```

Các Hàm và Toán Tử Trên Tuple

- **Singleton tuple syntax :** **Tạo tuple chỉ có một phần tử** hơi phức tạp chút, nếu tạo theo cách thông thường là cho phần tử đó vào trong cặp dấu () là chưa đủ, cần phải thêm dấu phẩy để chỉ ra rằng, đây là tuple.

```
t = (42)
print(type(t), t*5)
t = (42,)
print(type(t), t*5)
t = (42,2)
print(type(t), t*2)
```

```
<class 'int'> 210
<class 'tuple'> (42, 42, 42, 42, 42)
<class 'tuple'> (42, 2, 42, 2, 42, 2, 42, 2, 42, 2)
```

Các Hàm và Toán Tử Trên Tuple

```
t = [42]
print(type(t), t)
t = (42)
print(type(t), t)
t = (42)
print(type(t), t*5)
t = (42,)
print(type(t), t*5)
t = [42]
print(type(t), t*5)
```

```
<class 'list'> [42]
<class 'int'> 42
<class 'int'> 210
<class 'tuple'> (42, 42, 42, 42, 42)
<class 'list'> [42, 42, 42, 42, 42]
```

Các Hàm và Toán Tử Trên Tuple

- **Truy cập vào các phần tử của tuple (tương tự list): dùng index [n]** hoặc **[a:b]** hoặc **[-k]**

```
# tuple lồng nhau
n_tuple = ("Quantrimang.com", [2, 6, 8], (1, 2, 3))
# index lồng nhau
# Output: 'r'
print(n_tuple[0][5])
# index lồng nhau
# Output: 8
print(n_tuple[1][2])
```

Các Hàm và Toán Tử Trên Tuple

- **Thay đổi một tuple:** Không giống như list, **tuple không thể thay đổi.**
  - Điều này có nghĩa là các phần tử của một tuple không thể thay đổi một khi đã được gán.
  - Nhưng, nếu **bản thân phần tử đó là một kiểu dữ liệu có thể thay đổi** (như list chẳng hạn) thì **các phần tử lồng nhau có thể được thay đổi**. Chúng ta cũng có thể gán giá trị khác cho tuple (gọi là gán lại - **reassignment**).

```
my_tuple = (1, 3, 5, [7, 9])
#không thể thay đổi phần tử của tuple
#Bạn sẽ nhận được lỗi:
#TypeError: 'tuple' object does not support item assignment
my_tuple[1] = 9
```

Các Hàm và Toán Tử Trên Tuple

```
my_tuple = (1, 3, 5, [7, 9])
# Nhưng phần tử có index 3 trong tuple là list
# list có thể thay đổi, nên phần tử đó có thể thay đổi
# Output: (1, 3, 5, [8, 9])
my_tuple[3][0] = 8
print(my_tuple)
```

Các Hàm và Toán Tử Trên Tuple

```
my_tuple = (1, 3, 5, [7, 9])
# Nếu cần thay đổi tuple hãy gán lại giá trị cho nó
# Output: ('q', 'u', 'a', 'n', 't', 'r', 'i', 'm', 'a', 'n', 'g')
my_tuple = ('q', 'u', 'a', 'n', 't', 'r', 'i', 'm', 'a', 'n', 'g‘)
print(my_tuple)
```

Các Hàm và Toán Tử Trên Tuple

- Nối tuple (+) và lặp lại tuple (*)

```
# Nối 2 tuple
# Output: (2, 4, 6, 3, 5, 7)
print((2, 4, 6) + (3, 5, 7))
# Lặp lại tuple
# Output: ('Quantrimang.com', 'Quantrimang.com', 'Quantrimang.com')
print(("Quantrimang.com",) * 3)
```

Các Hàm và Toán Tử Trên Tuple

- **Xóa tuple:** Các phần tử trong tuple không thể thay đổi nên chúng ta cũng **không thể xóa, loại bỏ phần tử khỏi tuple**.
- Nhưng việc xóa hoàn toàn một tuple có thể thực hiện được với từ khóa del như dưới đây

```
QTM = ['q','u','a','n','t','r','i','m','a','n','g','.','c','o','m']
# Không thể xóa phần tử của tuple
# TypeError: 'tuple' object doesn't support item deletion
del QTM[3]
# Có thể xóa toàn bộ tuple
del QTM
```

Các Hàm và Toán Tử Trên Tuple

- **count(x)**: Đếm số phần tử x trong tuple.
- **index(x)**: Trả về giá trị index của phần tử x đầu tiên mà nó gặp trong tuple.

```
QTM = ['q','u','a','n','t','r','i','m','a','n','g','.','c','o','m']
# Count
# Output: 2
print(QTM.count('m'))
# Index
# Output: 3
print(QTM.index('n'))
```

Các Hàm và Toán Tử Trên Tuple

- Toán tử: in và not in

```
QTM = ['q','u','a','n','t','r','i','m','a','n','g','.','c','o','m']
# Kiểm tra phần tử
 # Output: True
print('a' in QTM)
# Output: False
 print('b' in QTM)
# Output: False
print('g' not in QTM)
```

Các Hàm và Toán Tử Trên Tuple

- **Vòng lặp và tuple**

```
for ngon_ngu in ('Python','C#,'Web'):
        print(“ Lap trinh",ngon_ngu)
```