



## KIỂU DỮ LIỆU: CHUỖI, LIST, TUPLE, DICTIONARY, SETS

- 01** Xử lý chuỗi
- 02** Danh sách
- 03** Tuples
- 04** Kiểu từ điển (Dictionaries)
- 05** Kiểu tập hợp (Sets)
- 06** Thảo luận

1




## 1. Xử lý chuỗi




- Chuỗi được định nghĩa nằm trong dấu nháy đơn (') hoặc dấu nháy kép (") hoặc 3 dấu nháy kép (""").
- Sử dụng các hàm thông qua đối tượng object . method name ( parameter list )

Tên hàm	Mô tả
upper, lower	Xử lý in Hoa, in thường
rjust	Căn lề phải
ljust	Căn lề trái
center	Căn giữa
strip	Xóa khoảng trắng dư thừa
startswith	Kiểm tra Chuỗi có phải bắt đầu là ký tự ?
endswith	Kiểm tra Chuỗi có phải kết thúc là ký tự ?
count	Đếm số lần xuất hiện trong Chuỗi
find	Tìm kiếm Chuỗi con
format	Định dạng Chuỗi
_len_()	Trả về số lượng ký tự trong chuỗi, dùng index để lấy ký tự ra: str[index]

2




# 1. Xử lý chuỗi




- Index
- Slicing
- Stride
- Concatenate
- Escape sequences
- String operations

3

3



# 1. Xử lý chuỗi



```

'''
Chuỗi
Thao tác trên chuỗi
'''

#Khai báo và xuất
hoten="Nguyen Lan Huong"
print('Ho ten: %s'%(hoten))
print(type(hoten))

#index, negative index
ten="Huong"#index: 0 1 2 3 4, negative index: -1 -2 -3 -4 -5
print("Ten: %s"% ten)
print("Chieu dai chuoi '%s' là %d"%(ten, len(ten)))
print("Ten[0]: %s, Ten[-1]: %s" %(ten[0], ten[-1]))
print("Ten[4]: %s, Ten[-5]: %s" %(ten[4], ten[-5]))

```


```

Ho ten: Nguyen Lan Huong
<class 'str'>
Ten: Huong
Chieu dai chuoi 'Huong' là 5
Ten[0]: H, Ten[-1]: g
Ten[4]: g, Ten[-5]: H


```

4

4



# 1. Xử lý chuỗi



```
#Slicing [start:end]: get from start to end index
hoten="Nguyen Lan Huong"
print("Ho ten: %s"%(hoten))
print("hoten[0:6]: '%s'%hoten[0:6])
print("hoten[7:10]: '%s'%hoten[7:10])

#Stride [start:end:step]
print("hoten[::3]: %s"%hoten[::3])
print("hoten[::3]: %s"%hoten[0:6:3])

#Concatenate
hoten=hoten + ", tuoi: 20. "
print(hoten)
print("Xuat 3 lan: %s"%(3*hoten))
```

Ho ten: Nguyen Lan Huong  
hoten[0:6]: 'Nguyen'  
hoten[7:10]: 'Lan'  
hoten[::3]: Ny nug  
hoten[::3]: Ny

Nguyen Lan Huong, tuoi: 20.  
Xuat 3 lan: Nguyen Lan Huong, tuoi: 20. Nguyen Lan Huong, tuoi: 20. Nguyen Lan Huong, tuoi: 20.

5

5



# 1. Xử lý chuỗi



```
hoten='Nguyen Lan Huong'
hoten=hoten.upper() #Chuyen chuoi In hoa
print(hoten)
hoten=hoten.lower() #Chuyen chuoi thuong
print(hoten)


hoten='Nguyen Lan Huong'
hoten=hoten.replace("Nguyen", "Tran") #Thay the chuoi "Nguyen" -> "Tran"
print(hoten)
vitri=hoten.find("Lan") #Tim vi tri xuat hien chuoi
print("Vi tri xuat hien 'Lan' la %s"%vitri)
vitri=hoten.find("Hung")
print("Vi tri xuat hien 'Hung' la %s"%vitri)
```

NGUYEN LAN HUONG  
nguyen lan huong


Tran Lan Huong  
Vi tri xuat hien 'Lan' la 5  
Vi tri xuat hien 'Hung' la -1

6

6




## 2. Danh sách




1. Khai báo và sử dụng List
2. Duyệt List
3. Gán giá trị cho các phần tử trong List
4. Phương thức insert
5. Phương thức append
6. Phương thức remove
7. Phương thức reverse
8. Phương thức sort
9. Slicing
10. List đa chiều

7

7

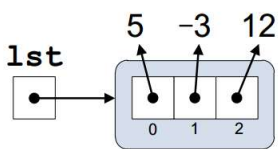


## 2.1. Cách khai báo và sử dụng List



- List trong Python là một đối tượng dùng để lưu tập các đối tượng khác. List có thể chứa bất kỳ kiểu dữ liệu nào. Tuy nhiên ta nên thống nhất một kiểu để dễ dàng trong quá trình xử lý.
- Khai báo list rỗng: **lst = []**
- Khai báo list có các giá trị: **lst = [2, -3, 0, 4, -1]**
- Khai báo list có 10 phần tử với giá trị mặc định là 0: **lst=[0]\*10**
- Khai báo list có 10 phần tử với giá trị mặc định là 0.5: **lst=[0.5]\*10**

lst=[5,-3,12]



```
print(lst[0])
print(lst)
print(len(lst))
```

8

8

## 2.2. Duyệt danh sách

- Có nhiều cách để duyệt, ví dụ như:
  - Duyệt theo collection
  - Duyệt theo chỉ số index

```

print("""*35)
lst=[5,7,2,9,6,3,10,17,16]
for x in lst:
    print(x,end='\t')
print()
print("""*35)
for i in range(len(lst)):
    x=lst[i]
    print(x,end='\t')
print()
print("""*35)
for i in range(len(lst)-1,-1,-1):
    x=lst[i]
    print(x,end='\t')

```

\*\*\*\*\*

5   7   2   9   6   3   10   17   16

\*\*\*\*\*

5   7   2   9   6   3   10   17   16

\*\*\*\*\*

16   17   10   3   6   9   2   7   5

9

9

## 2.3. Gán giá trị cho các phần tử trong List

- `lst = [2, 4, 6, 8]` → `lst` tham chiếu tới List
- `lst[2]` → tham chiếu tới phần tử thứ 2 (giá trị =6)

`a = [10, 20, 30, 40]`

`b[2] = 35`

`b = [10, 20, 30, 40]`

10

10

## 2.3. Gán giá trị cho các phần tử trong List

■ Minh họa gán tham chiếu:

`a = [10, 20, 30, 40]`

`b = a`

`b[2] = 35`

11

11

## 2.4. Phương thức insert

Phương thức insert trong list: chèn vào vị trí thích hợp

*`list.insert(vị trí muốn chèn, giá trị muốn chèn)`*

```
lst=[1,2,3]
print(lst)
lst.insert(2,9)
print(lst)
lst.insert(0,17)
print(lst)
```

➔

```
[1, 2, 3]
[1, 2, 9, 3]
[17, 1, 2, 9, 3]
```

12

12

## 2.5. Phương thức append

Phương thức append trong list: chèn giá trị vào cuối danh sách

*list.append(giá trị muốn chèn)*

```
lst=[1,2,3]
lst.append(-113)
print(lst)
```

→ [1, 2, 3, -113]

13

13

## 2.6. Phương thức remove

Phương thức remove trong list: xóa giá trị đầu danh sách

*list.remove(giá trị muốn xóa)*

```
lst=[2,0,1,8,0]
lst.remove(0)
print(lst)
```

→ [2, 1, 8, 0]

`del lst[0]`

14

14

## 2.7. Phương thức reverse

Phương thức **reverse** trong list: đảo danh sách

`list.reverse()`

```
lst=[8,1,0,2]
print(lst)
lst.reverse()
print(lst)
```

→

```
[8, 1, 0, 2]
[2, 0, 1, 8]
```

15

15

## 2.8. Phương thức sort

Phương thức **sort** trong list: sắp xếp danh sách

`list.sort(reverse=True|False, key=myFunc)`

Tùy chọn:

- `reverse=True`: sắp giảm
- `key`: hàm chỉ ra trường giá trị cần sắp xếp

```
lst=[8,1,0,2]
print(lst)
lst.sort()
print(lst)
```

→

```
[8, 1, 0, 2]
[0, 1, 2, 8]
```

Hoặc  
`lst=sorted(lst)`

16

16





## 2.8. Phương thức sort



Ví dụ:

```
# A function that returns the length of the
value:
def myFunc(e):
    return len(e)
```

```
cars = ['Ford', 'Mitsubishi', 'BMW', 'VW']
```

```
cars.sort(key=myFunc)
```

```
# A function that returns the length of the
value:
```

```
def myFunc(e):
    return len(e)
```

```
cars = ['Ford', 'Mitsubishi', 'BMW', 'VW']
```

```
cars.sort(reverse=True, key=myFunc)
```

```
# A function that returns the 'year' value:
def myFunc(e):
    return e['year']
```

```
cars = [
    {'car': 'Ford', 'year': 2005},
    {'car': 'Mitsubishi', 'year': 2000},
    {'car': 'BMW', 'year': 2019},
    {'car': 'VW', 'year': 2011}
]
```

```
cars.sort(key=myFunc)
```

17

17



## 2.9. Slicing



Slicing dùng để trích lọc list

*list [ begin : end : step ]*

list: là danh sách

begin: Vị trí bắt đầu cắt

end: Vị trí cuối cùng cắt

step: bước nhảy

```
lst = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]
print(lst) # [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]
print(lst[0:3]) # [10, 20, 30]
print(lst[4:8]) # [50, 60, 70, 80]
print(lst[2:5]) # [30, 40, 50]
print(lst[-5:-3]) # [80, 90]
print(lst[:3]) # [10, 20, 30]
print(lst[4:]) # [50, 60, 70, 80, 90, 100, 110, 120]
print(lst[:]) # [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]
print(lst[-100:3]) # [10, 20, 30]
print(lst[4:100]) # [50, 60, 70, 80, 90, 100, 110, 120]
print(lst[2:-2:2]) # [30, 50, 70, 90]
print(lst[::2]) # [10, 30, 50, 70, 90, 110]
print(lst[::-1]) # [120, 110, 100, 90, 80, 70, 60, 50, 40, 30, 20, 10]
```

18

18

## 2.10. List đa chiều

```

matrix = [
    [100, 14, 8, 22, 71],
    [ 0, 243, 68, 1, 30],
    [ 90, 21, 7, 67, 112],
    [115, 200, 70, 150, 8]
]

print(matrix)

for row in matrix: # duyệt từng dòng
    for elem in row: # Lấy từng phần tử trên mỗi dòng
        print('{:>4}'.format(elem), end='')
    print()

```

19

19

## 2.10. List đa chiều

■ Khởi tạo

```

row=5
column=3
lst=[0]*column]*row
print(lst)
↓
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]

```

```

arr2D = []
rowsize=5
columnsize=3
for i in range(rowsize):
    onerow = []
    for j in range(columnsize):
        onerow.append(randrange(-100,100))
    arr2D.append(onerow)

for i in range(len(arr2D)):
    for j in range(len(arr2D[i])):
        print(arr2D[i][j],end='\t')
    print()

for row in arr2D:
    for column in row:
        print(column,end='\t')
    print()

```

20

20

## Ví dụ: List

```

#List in python
#Create
list1=['CNTT', 100, 3.14, True]
print("type(list1)=", type(list1))
print("len(list1)=%s"%len(list1))
#Indexing
print("list1[0]=", list1[0], ", list1[-2]=", list1[-3])
#Content
list1=['CNTT', 100, 3.14,(2,'tuple'), True, [2, 0, 10, 2]]
print("type(list1)=",type(list1))
print("type(list1[3])=",list1[3])
print("type(list1[5])=",type(list1[5]))
print("list1[5][1]=%s"%list1[5][1])

type(list1)= <class 'list'>
len(list1)=4
list1[0]= CNTT , list1[-2]= 100
type(list1)= <class 'list'>
type(list1[3])= (2, 'tuple')
type(list1[5])= <class 'list'>
list1[5][1]=0

```

21

21

## Ví dụ: List

```

#Operations

list1=['CNTT', 100, 3.14,(2,'tuple'), True, [2, 0, 10, 2]]
print("list1[2:5]", list1[2:5])
#Extend
list1.extend([2022, "extend"])
print("extend:",list1)
#append
list1.append([9, 99, "append"])
print("Append:",list1)

list1[0]=2000
print("change list1[0]:",list1)
del(list1[0])
print("del(list1[0]): ",list1)

str1='10,a,b,200'
str1=str1.split(',')
print(str1, type(str1))

list1[2:5] [3.14, (2, 'tuple'), True]
extend: ['CNTT', 100, 3.14, (2, 'tuple'), True, [2, 0, 10, 2], 2022, 'extend']
Append: ['CNTT', 100, 3.14, (2, 'tuple'), True, [2, 0, 10, 2], 2022, 'extend', [9, 99, 'append']]
change list1[0]: [2000, 100, 3.14, (2, 'tuple'), True, [2, 0, 10, 2], 2022, 'extend', [9, 99, 'append']]
del(list1[0]): [100, 3.14, (2, 'tuple'), True, [2, 0, 10, 2], 2022, 'extend', [9, 99, 'append']]
['10', 'a', 'b', '200'] <class 'list'>

```

22

22

## Ví dụ: List

```
#Copy and clone
list1=[100, 200, 'abc']
list2=list1
list1[0]='CNTT'
print("list1:", list1)
print("list2:", list2)

list1=[100, 200, 'abc']
list3=list1[:]
list1[0]=999
print("list1:", list1)
print("list3:", list3)
list3[0]='CNTT'
print("list1:", list1)
print("list3:", list3)
```

```
list1: ['CNTT', 200, 'abc']
list2: ['CNTT', 200, 'abc']

list1: [999, 200, 'abc']
list3: [100, 200, 'abc']
list1: [999, 200, 'abc']
list3: ['CNTT', 200, 'abc']
```

23

23

## 3. Tuples

- Tạo
- Index
- Nói
- Slicing
- Sắp xếp
- Nested

24

24

## 3. Tuples

```

#Tạo Tuple
tup1=('Xin chao', 100,True, 3.14)
print(type(tup1))
#Indexing
print("tup1[0]='%s"%tup1[0]+", type(tup1[0])=%s"%type(tup1[0]))
print("tup1[2]='%s"%tup1[2]+", type(tup1[2])=%s"%type(tup1[2]))
print("tup1[3]='%s"%tup1[3]+", type(tup1[3])=%s"%type(tup1[3]))
#Nối tuples
tup2=(10, False)
tup3=tup1+tup2
print(tup3)
#Chiều dài của tuple
print("Chieu dai cua tup3:%s"%len(tup3))
#Slicing
print(tup3[1:5])

```

```

<class 'tuple'>
tup1[0]='Xin chao, type(tup1[0])=<class 'str'>
tup1[2]='True, type(tup1[2])=<class 'bool'>
tup1[3]='3.14, type(tup1[3])=<class 'float'>
('Xin chao', 100, True, 3.14, 10, False)
Chieu dai cua tup3:6
(100, True, 3.14, 10)

```

25

25

## 3. Tuples

```

tup4=(10, 0, -1, 8, 20, 15)
print("tup4=",tup4,"type=%s"%type(tup4))
#Sort
tup4=sorted(tup4)
print("tup4=%s"%tup4 + " type=%s"%type(tup4))
#Nested
nestedtup=('Xin chao', 100, (10, 0, 20, 15), True, 3.14)
print("nestedtup=",nestedtup)
print("nestedtup[2]=",nestedtup[2], " type(nestedtup[2])=", type(nestedtup[2]))
print("nestedtup[2][0]=",nestedtup[2][0])

```


```

tup4= (10, 0, -1, 8, 20, 15) type=<class 'tuple'>
tup4=[-1, 0, 8, 10, 15, 20] type=<class 'list'>
nestedtup= ('Xin chao', 100, (10, 0, 20, 15), True, 3.14)
nestedtup[2]= (10, 0, 20, 15) type(nestedtup[2])= <class 'tuple'>
nestedtup[2][0]= 10


```

26

26




# 4. Dictionaries




- Create
- Get value from key
- Get all keys
- Get all values
- Verify key
- Add
- Delete

27

27



# 4. Dictionaries



```
#Dictionary

#Create
dict1={'k1':100,
      'k2':['CNTT', 'IT'],
      'k3':(10, 'CT', True),
      (1, 2, 4):'value'
}
print(type(dict1))
print(dict1)


#Get value from key
print(dict1['k3'])
print(type(dict1['k3']))
#Get all keys
print(dict1.keys())
#Get all values
print(dict1.values())
#Verify key
print(('k1' in dict1))
print(((1, 6, 4) in dict1))
```

```
<class 'dict'>
{'k1': 100, 'k2': ['CNTT', 'IT'], 'k3': (10, 'CT', True), (1, 2, 4): 'value'}
(10, 'CT', True)
<class 'tuple'>
dict_keys(['k1', 'k2', 'k3', (1, 2, 4)])
dict_values([100, ['CNTT', 'IT'], (10, 'CT', True), 'value'])
True
False
```



28

28





## 4. Dictionaries

```
dict1={'k1':100,
      'k2':['CNTT', 'IT'],
      'k3':(10, 'CT', True),
      (1, 2, 4):'value'
}
print(dict1)
#Add entry
dict1['k4']='value 4'
print("After add \'k4\' :", dict1)

#Delete entry
del(dict1['k4'])
print("After delete \'k4\' :", dict1)
```


{'k1': 100, 'k2': ['CNTT', 'IT'], 'k3': (10, 'CT', True), (1, 2, 4): 'value'}

After add 'k4': {'k1': 100, 'k2': ['CNTT', 'IT'], 'k3': (10, 'CT', True), (1, 2, 4): 'value', 'k4': 'value 4'}



After delete 'k4': {'k1': 100, 'k2': ['CNTT', 'IT'], 'k3': (10, 'CT', True), (1, 2, 4): 'value'}

29

29




## 5. Sets



- Gán dữ liệu
- Chuyển list sang set
- Thao tác trên tập hợp
- Thao tác logic trên tập hợp

30

30



## 5. Sets





```
#Sets
#Set content
s={3, 8, 'CNTT', 3} #3 is not unique in database
print(s)



#Convert list to set
list1=[1, 2, 3, 1]
s=set(list1)
print(s,type(s))
#add to set
s.add(100)
print(s)
```

31

31



## 5. Sets

```
#Set logic operations
s1={1, 20, 15}
s2={20, 8, 10}
print(s1&s2)
print(s1.intersection(s2))
print(s1.union(s2))
print(s1.difference(s2))
s3={1, 15}
print(s3.issubset(s1))
s4={100,1,50, 20, 15}
print(s4.issubset(s1))
```

{20}

{20}

{1, 20, 8, 10, 15}

{1, 15}

True

False

32

32



# THẢO LUẬN

33

33

## Question?

34