

Ngôn Ngữ Lập Trình Python

CẤU TRÚC VÀ CÚ PHÁP

Trịnh Tân Đạt

Đại Học Sài Gòn

trinhtandat@sgu.edu.vn

<http://sites.google.com/site/ttdat88>

Biến, Từ Khóa, Định Danh, Kiểu Dữ Liệu

Nội Dung

- Từ Khóa
- Định Danh
- Chú Thích
- Biến
- Nhập dữ liệu từ bàn phím/ Xuất dữ liệu
- Câu lệnh / Khối lệnh
- Toán Tử Trong Python
- Kiểu số

Từ Khóa Trong Python

- Những từ chỉ dành riêng cho Python
- Trong Python, ngoại trừ **True**, **False** và **None** được viết hoa ra thì các keyword khác đều được viết dưới dạng chữ thường, đây là điều bắt buộc.

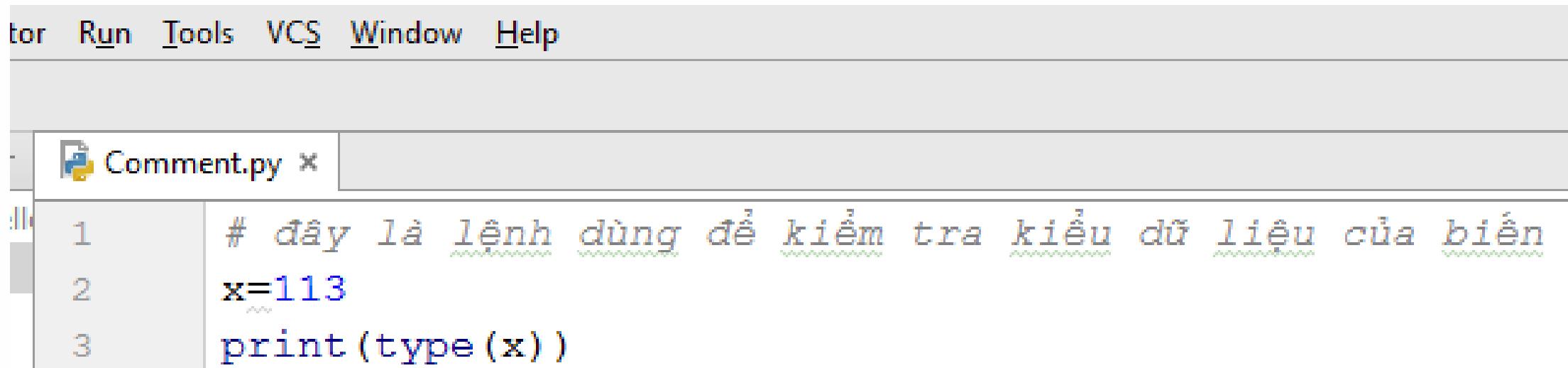
• False	• class	• finally	• is	• return
• None	• continue	• for	• lambda	• try
• True	• def	• from	• nonlocal	• while
• and	• del	• global	• not	• with
• as	• elif	• if	• or	• yield
• assert	• else	• import	• pass	
• break	• except	• in	• raise	

Định Danh Trong Python

- Định danh là tên được đặt cho các thực thể như class, function, biến,... trong Python giúp phân biệt thực thể này với thực thể khác.
- Quy tắc viết định danh trong Python:
 - Định danh có thể là sự kết hợp giữa các chữ cái viết thường (**từ a đến z**) hoặc viết hoa (**A đến Z**) hoặc các số (**từ 0 đến 9**) hoặc dấu gạch dưới (_). Định danh hợp lệ sẽ giống như thế này: bien_1, tinh_tong_0_9, firstClass.
 - Định danh **không thể bắt đầu bằng một chữ số**, ví dụ 1bien là không hợp lệ, nhưng bien1 thì đúng.
 - Định danh phải **khác các keyword**. Bạn thử nhập and = 1 rồi chạy sẽ xuất hiện thông báo lỗi "SyntaxError: invalid syntax".
 - Python **không hỗ trợ các ký tự đặc biệt** như !, @, #, \$, %,... trong định danh. Nếu cố tình gán các ký tự này trong định danh sẽ nhận được thông báo lỗi "SyntaxError: invalid syntax" hoặc "NameError:..."
 - Python là ngôn ngữ lập trình **phân biệt chữ hoa, chữ thường**, nghĩa là bien và Bien là không giống nhau.

Chú Thích (comment)

- Python sử dụng kí tự # để chú thích các đoạn code
- Tất cả các nội dung sau kí tự # sẽ không được dịch



```
Comment.py
1 # đây là lệnh dùng để kiểm tra kiểu dữ liệu của biến
2 x=113
3 print(type(x))
```

Chú Thích (comment)

- Ghi chú nhiều dòng: Dùng """ """(3 cặp nháy đôi) hoặc ''' '''(3 cặp nháy đơn)

```
1 """
2 Giải phương trình bậc 1: ax+b=0
3 Có 3 trường hợp để biện luận
4 Nếu hệ số a =0 và hệ số b=0 ==>vô số nghiệm
5 Nếu hệ số a =0 và hệ số b !=0 ==>vô nghiệm
6 Nếu hệ số a !=0 ==> có nghiệm -b/a
"""
7
8 a = 0
9 b = 113
10 if a == 0 and b == 0:
11     print("Vô số nghiệm")
12 elif a == 0 and b != 0:
13     print("Vô nghiệm")
14 else:
15     print("Có № X=", -b/a)
```

Chú Thích (comment)

- Ví dụ dùng 3 cặp nháy đơn:

```
1   ...
2   Đây là lệnh kiểm tra năm nhuận year
3   Năm nhuận là năm chia hết cho 4 nhưng không chia hết cho 100 hoặc
4   ...
5   year=2016
6   if (year % 4==0 and year %100 !=0) or year % 400 ==0:
7       print(year," Là năm nhuận")
8   else:
9       print(year, " KO là năm nhuận")
```

Biến Trong Python

- Một biến Python là một khu vực bộ nhớ được dành riêng để lưu trữ các giá trị.
- Mỗi giá trị trong Python có một kiểu dữ liệu. Các kiểu dữ liệu khác nhau trong Python là
 - kiểu số (number)
 - kiểu danh sách (list)
 - kiểu bộ (tuple)
 - kiểu chuỗi (string)
 - kiểu từ điển (dictionary)
 - ...

Biến trong Python

- Biến trong python:
 - Có tên, phân biệt chữ hoa/thường
 - Không cần khai báo trước
 - Không cần chỉ ra kiểu dữ liệu
 - Có thể thay đổi sang kiểu dữ liệu khác
 - Nên gán giá trị ngay khi bắt đầu xuất hiện
 - Không được trùng với từ khóa
- **Cách khai báo và sử dụng biến**
 - Ví dụ : **n = 12** # biến n là kiểu nguyên
n = n + 0.1 # biến n chuyển sang kiểu thực
print(n)

Biến trong Python

```
1 # Declare a variable and initialize it
2 f = 0
3 print(f)
# re-declaring the variable works
4 f = 'guru99'
5 print(f)
6
7
8
9
10
11
12
```

Run Python5.1

0 guru99

"C:\Users\DK\PycharmProjects\Python Test\Py

Bạn có thể khai báo lại một biến dù đã khai báo trước đó. Điều này hoàn toàn được chấp nhận.

Biến trong Python

- Tất cả mọi biến trong python đều là các đối tượng, vì thế nó có kiểu (type) và vị trí trong bộ nhớ (id)

```
C:\Users\sony>python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Ana
conda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 10
>>> b = 1.535
>>> c = "hello"
>>>
>>> id(a)
8791359468656
>>> id(a), id(b), id(c)
(8791359468656, 4366984, 43086488)
>>> type(a), type(b), type(c)
(<class 'int'>, <class 'float'>, <class 'str'>)
>>>
```

Biến trong Python

Gán nhiều giá trị:

- Trong Python bạn có thể thực hiện gán nhiều giá trị trong một lệnh như sau:
s, x, f = “Vang”, 3, 5.5
- Nếu muốn gán giá trị giống nhau cho nhiều biến thì có thể viết lệnh như sau:
s, x, f = 3

Biến trong Python

Biến toàn cục và biến cục bộ

- Trong Python khi muốn sử dụng **cùng một biến cho toàn bộ chương trình hoặc mô-đun**, cần khai báo nó dưới dạng biến toàn cục.
- Trường hợp muốn sử dụng biến **trong một hàm hoặc phương thức cụ thể**, hãy sử dụng biến cục bộ.

Biến trong Python

```
# Declare a variable and initialize it  
f = 101  
print(f)  
# Global vs. local variables in functions  
def someFunction():  
    f = 'I am learning Python'  
    print(f)  
someFunction()  
print(f)
```

f là biến cục bộ được khai báo bên trong hàm

```
Python5.2.py x  
1 # Declare a variable and initialize it  
2 f = 101 ①  
3 print(f)  
4  
5 # Global vs. local variables in functions  
6 def someFunction():  
7     # global f  
8     f = 'I am learning Python' ②  
9     print(f)  
10    someFunction()  
11    print(f) ③  
12  
13  
Run Python5.2  
C:\Users\DK\Desktop\Python code\Python Test\Python 5\PythonCode:  
5\PythonCode5\Python5.2.py"  
101 ①  
I am learning Python ②  
101 ③
```

Biến trong Python

global f

The screenshot shows a Python code editor with the following code:

```
1 f = 101; print(f) ①
2
3
4 # Global vs. local variables in functions
5 def someFunction():
6     global f
7     print(f)
8     f = "changing global variable" ②
9
10 someFunction()
11 print(f) ③
12
13
14
```

The code uses a global variable `f`. At step 1, it prints the value 101. At step 2, it changes the value of `f` to a string "changing global variable". At step 3, it prints the modified value.

A callout box with an orange border contains the text: "Giờ chúng ta đang truy xuất và thay đổi giá trị của biến toàn cục f".

The code editor interface includes a vertical toolbar on the left, a status bar at the bottom, and a run configuration bar at the bottom left.

Biến trong Python

- Dữ liệu kiểu chuỗi rất quan trọng trong lập trình python
- Khai báo dữ liệu kiểu chuỗi có thể nằm bên trong cặp nháy đơn ('), hoặc nháy kép ("") hoặc 3 dấu nháy kép liên tiếp (""""")
- Ví dụ:

```
S_name= 'AN'
```

```
S_sentence = "Hello Hello"
```

```
# chuỗi có nội dung nằm trên 2 dòng
```

```
S_twoline= """This string has  
multiple lines in it"""
```

Xuất Dữ Liệu

- Sử dụng hàm **print** để in dữ liệu ra màn hình

```
>>> print(42)
42
>>> print("a = ", a)
a = 3.564
>>> print("a = \n", a)
a =
3.564
>>> print("a","b")
a b
>>> print("a","b",sep="")
ab
>>> print(192,168,178,42,sep=". ")
192.168.178.42
>>> print("a","b",sep=":-)")
a:-)b
```

Nhập Dữ Liệu

- Sử dụng hàm **input** để nhập dữ liệu từ bàn phím

```
name = input("What's your name? ")
print("Nice to meet you " + name + "!")
age = input("Your age? ")
print("You are already " + age + " years old, " + name + "!")
```

```
In [5]: runfile('C:/Users/sony/Desktop/
python_co_ban/test/Test1.py', wdir='C:/Users/
sony/Desktop/python_co_ban/test')
```

```
What's your name? Dat
Nice to meet you Dat!
```

```
Your age? 30
You are already 30 years old, Dat!
```

Ví Dụ Minh Họa

```
a = float(input("A = "))
b = float(input("B = "))
c = float(input("C = "))
delta = b*b-4*a*c
if delta==0:
    print("Nghiem kep: x = ", str(-b/2/a))
if delta<0:
    print("Phuong trinh vo nghiem")
if delta>0:
    print("X1 = " + str((-b+delta**0.5)/2/a))
    print("X2 = " + str((-b-delta**0.5)/2/a))
```

Nhập a,b,c kiểu số thực và tính delta

Biện luận các trường hợp của delta

Các khối lệnh con được viết thụt vào so với khối cha

Tính căn bậc 2 bằng phép lũy thừa 0.5

Câu lệnh trong Python

- Trong Python, một câu lệnh được kết thúc bằng ký tự dòng mới, nghĩa là một câu lệnh sẽ kết thúc khi bạn xuống dòng.
- Chúng ta có thể mở rộng câu lệnh trên nhiều dòng với ký tự tiếp tục dòng lệnh (\).

```
sum = 1 + 3 + 5 + \
      7 + 9 + 11 + \
      13 + 15 + 17
```

Câu lệnh trong Python

- Trong Python, viết tiếp câu lệnh thường sử dụng dấu ngoặc đơn (), ngoặc vuông [] hoặc ngoặc nhọn {}

```
sum = (1 + 3 + 5 +  
       7 + 9 + 11 +  
       13 + 15 + 17)
```

Dấu () ở đây ngầm ý tiếp tục dòng lệnh

```
mau_sac = {"vang",  
           "xanh",  
           "cam"}
```

- Bạn cũng có thể đặt nhiều lệnh trên cùng một dòng, phân cách nhau bởi dấu **chấm phẩy** ;

```
a = 1; b = 2; c = 3
```

Canh lề trong Python

- C, C++ hay Java bạn sẽ biết rằng những ngôn ngữ lập trình này sử dụng { } để xác định các khối code.
- Trong Python thì khác, những khối lệnh sẽ được nhận biết thông qua canh lề. Đó là lý do vì sao canh lề trong Python vô cùng quan trọng, nếu bạn thừa hoặc thiếu khoảng trắng là sẽ bị báo lỗi ngay.
 - Một khối code (thường là khối lệnh của hàm, vòng lặp, if ...) bắt đầu với canh lề và kết thúc với dòng đầu tiên không canh lề.

```
for i in range(1,11):
    print(i)
    if i == 5:
        break
    print("hello")
```

Canh lề trong Python

- Nếu lệnh trên được viết thành:

```
for i in range(1,11):
    print(i)
        if i == 5:
            break
```

- Lỗi sẽ hiện trước lệnh print(i).

```
30
31 for i in range(1,11):
32     print(i)
33         if i == 5:
34             break
```

```
File "C:/Users/sony/Desktop/python_co_ban/
test/Test1.py", line 32
    print(i)
               ^
IndentationError: expected an indented block
```

Toán Tử Trong Python

- Python hỗ trợ nhiều phép toán số, logic, gán, so sánh và phép toán bit
- **Toán tử số học:** $+, -, *, /, \%, **$ ($\%$: phép chia lấy dư ; $**$: lũy thừa)
Ví dụ : $2^{**}3 = 8$; $3^{**}3 = 27$
- Python có 2 phép chia:
 - chia đúng ($/$): $10/3$ # 3.3333333333333335
 - chia nguyên ($//$): $10//3$ # 3 (nhanh hơn phép $/$)
- **Các phép so sánh:** $<$, \leq , $>$, \geq , \neq , \equiv

```
In [2]: runfile('C:/Users/sony/Desktop/python_co_ban/test/Test1.py', wdir='C:/Users/sony/Desktop/python_co_ban/test')
x > y is False
```

```
x = 4
y = 5
print('x > y is', x>y)
```

Toán Tử Trong Python

- **Toán tử gán:** Các toán tử gán khác nhau được sử dụng trong Python là (=, +=, -=, *=, /=, ...)

```
num1 = 4
num2 = 5

print("Line 1 - Value of num1 : ", num1)
print("Line 2 - Value of num2 : ", num2)
```

```
num1 = 4
num2 = 5

res = num1 + num2
res += num1

print("Line 1 - Result of + is ", res)
```

```
In [3]: runfile('C:/Users/sony/Desktop/
python_co_ban/test/Test1.py', wdir='C:/Users/
sony/Desktop/python_co_ban/test')
('Line 1 - Result of + is ', 13)
```

Toán Tử Trong Python

- Toán tử logic: **and, or, not**

```
a = True  
b = False  
  
print('a and b is',a and b)  
print('a or b is',a or b)  
print('not a is',not a)
```

```
In [4]: runfile('C:/Users/sony/Desktop/python_co_ban/test/Test1.py'  
('a and b is', False)  
('a or b is', True)  
('not a is', False)
```

Toán Tử Trong Python

- **Toán tử thành viên :** Các toán tử này kiểm tra tư cách thành viên trong một tập như danh sách, chuỗi hoặc tuple. Có hai toán tử thành viên được sử dụng trong Python là (**in, not in**). Kết quả trả về phụ thuộc vào việc biến có tồn tại trong chuỗi hoặc tập cho trước hay không.

```
x = 4
y = 8
list = [1, 2, 3, 4, 5 ];
if ( x in list ):
    print("Line 1 - x is available in the given list")
else:
    print("Line 1 - x is not available in the given list")
if ( y not in list ):
    print("Line 2 - y is not available in the given list")
else:
    print("Line 2 - y is available in the given list")
```

```
In [5]: runfile('C:/Users/sony/Desktop/python_co_ban/test/Test1.py', wdir='C:/Users/sony/Desktop/python_co_ban/test')
Line 1 - x is available in the given list
Line 2 - y is not available in the given list
```

Kiểu Dữ Liệu: Number

- Python hỗ trợ số nguyên, số thập phân và số phức, chúng lần lượt được định nghĩa là các lớp int, float, complex trong Python
- Số nguyên trong Python không bị giới hạn độ dài, số thập phân bị giới hạn đến 16 số sau dấu thập phân.

Tiền tố hệ số cho các số Python:

Hệ thống số	Tiền tố
Hệ nhị phân	'0b' hoặc '0B'
Hệ bát phân	'0o' hoặc '0O'
Hệ thập lục phân	'0x' hoặc '0X'

(Bạn đặt tiền tố nhưng không có dấu ' ')

Kiểu Dữ Liệu: Number

- Python viết số nguyên theo nhiều hệ cơ số

- A = 1234 # hệ cơ số 10
- B = 0xAF1 # hệ cơ số 16
- C = 0o772 # hệ cơ số 8
- D = 0b1001 # hệ cơ số 2

```
A = 1234
print("A:", type(A), A)
B = 0x1A
print("B:", type(B), B)
C = 0o720
print("C:", type(C), C)
D = 0b1010
print("D:", type(D), D)
```

```
In [26]: runfile('C:/Users/son
python_co_ban/test/Test1.py',
sony/Desktop/python_co_ban/tes
A: <class 'int'> 1234
B: <class 'int'> 26
C: <class 'int'> 464
D: <class 'int'> 10
```

Kiểu Dữ Liệu: Number

- Chuyển đổi từ số nguyên thành string ở các hệ cơ số khác nhau

- K = str(1234) # chuyển thành str ở hệ cơ số 10
- L = hex(1234) # chuyển thành str ở hệ cơ số 16
- M = oct(1234) # chuyển thành str ở hệ cơ số 8
- N = bin(1234) # chuyển thành str ở hệ cơ số 2

```
K = str(1234)
print("K:", type(K), K)
L = hex(1234)
print("L:", type(L), L)
M = oct(1234)
print("M:", type(M), M)
N = bin(1234)
print("M:", type(N), N)
```

```
In [13]: runfile('C:/Users/sony/[...]
python_co_ban/test/Test1.py', wd:
sony/Desktop/python_co_ban/test')
K: <class 'str'> 1234
L: <class 'str'> 0x4d2
M: <class 'str'> 0o2322
N: <class 'str'> 0b10011010010
```

Kiểu Dữ Liệu: Number

- Python hỗ trợ kiểu số phức, với chữ j đại diện cho phần ảo

- A = 3+4j
- B = 2-2j
- print(A+B)

sẽ in ra (5+2j)

Kiểu Dữ Liệu: Number

Chuyển đổi giữa các kiểu số trong Python

- Ví dụ: Nếu bạn thực hiện phép cộng giữa số nguyên là 2 và số thập phân là 3.0, thì 2 sẽ chuyển thành số thập phân 2.0 và kết quả trả về sẽ là số thập phân 5.0.

```
>>> 2 + 3.0  
5.0
```

```
print(2+3.0)
```

Kiểu Dữ Liệu: Number

- Có thể sử dụng các hàm Python tích hợp sẵn như **int()**, **float()** và **complex()** để chuyển đổi giữa các kiểu số một cách rõ ràng. Những hàm này thậm chí có thể chuyển đổi từ các chuỗi.

```
>>> int(3.6)
3
>>> int(-1.2)
-1
>>> float(7)
7.0
>>> complex('2+8j')
(2+8j)
```

Cấu Trúc Rẽ Nhánh, Vòng Lặp

Nội Dung

- Cấu trúc rẽ nhánh
- Vòng lặp while
- Vòng lặp for
- Lệnh break, continue
- Function
- Bài tập

Cấu trúc rẽ nhánh if-else

```
if expression:  
    # If-block
```

```
if expression:  
    # If-block  
elif 2-expression:  
    # 2-if-block  
elif 3-expression:  
    # 3-if-block  
...  
elif n-expression:  
    # n-if-block
```

```
if expression:  
    # If-block  
else:  
    # else-block
```

```
if expression:  
    # If-block  
elif 2-expression:  
    # 2-if-block  
...  
elif n-expression:  
    # n-if-block  
else:  
    # else-block
```

Cấu trúc rẽ nhánh if-else

- Chú ý: python nhạy cảm với việc viết khối mã

```
num = 3
if num > 0:
    print(num, "là số dương.")
print("Thông điệp này luôn được in.")

num = -1
if num > 0:
    print(num, "là số dương.")
print("Thông điệp này cũng luôn được in.")
```

Cấu trúc rẽ nhánh if-else

```
# Kiem tra xem so am hay duong  
# Va hien thi thong bao phu hop  
  
num = 3  
  
if num >= 0:  
    print("So duong hoac bang 0")  
else:  
    print("So am")  
  
num1=-1  
  
if num1 >= 0:  
    print("So duong hoac bang 0")  
else:  
    print("So am")
```

Cấu trúc rẽ nhánh if-else

```
x = int(input("Nhập một số: "))

if x < 0:
    x = 0
    print('Số âm')
elif x == 0:
    print('Số 0')
elif x == 1:
    print('Số 1')
else:
    print('Số dương')
```

Cấu trúc rẽ nhánh if-else

- Chú ý: python nhạy cảm với việc viết khối mã

```
num = float(input("Nhập một số: "))

if num >= 0:
    if num == 0:
        print("Số 'Không'")
    else:
        print("Số 'dương'")
else:
    print("Số 'âm'")
```

Vòng lặp while

```
while biểu_thức:  
    câu_lệnh
```

```
x=0  
#define a while loop  
while(x <4):  
    print(x)  
    x = x+1
```

Vòng lặp while

- Sử dụng while để tính tổng các số từ 1 đến n

```
n = int(input("Nhập n: ")) #Nhập số n tùy ý
tong = 0 #khai báo và gán giá trị cho tong
i = 1 #khai báo và gán giá trị cho biến đếm i

while i <= n:
    tong = tong + i
    i = i+1 # cập nhật biến đếm

print("Tổng là", tong)
```

Vòng lặp while

- Kết hợp while với else

```
dem = 0  
while dem < 3:  
    print("Đang ở trong vòng lặp while")  
    dem = dem + 1  
else:  
    print("Đang ở trong else")
```

Vòng lặp for

```
for variable_1, variable_2, ... variable_n in sequence:
```

```
# for-block
```

- Dùng hàm **range(a, b)** để tạo danh sách gồm các số từ **a** đến **b-1**, hoặc tổng quát hơn là **range(a, b, c)** trong đó **c** là bước nhảy

```
for d in range(10,20):          # in các số từ 10 đến 19
    print(d)
for d in range(20,10,-1):        # in các số từ 20 đến 11
    print(d)
```

Vòng lặp for

- Ví dụ :

```
x=0  
#Define a for loop  
for x in range(2,7):  
    print(x)
```

Vòng lặp for

- Ví dụ : Cách sử dụng vòng lặp for cho chuỗi

```
Months = ["Jan", "Feb", "Mar", "April", "May", "June"]
for m in Months:
    print(m)
```

Lệnh break

- Lệnh break: cho phép bạn ngắt hoặc chấm dứt việc thực hiện vòng lặp.

```
for var in sequence:
```

```
#khỏi code bên trong vòng lặp for
```

```
if dieu_kien:
```

```
    break
```

```
#code khác bên trong vòng lặp for
```

```
#code bên ngoài vòng lặp for
```

Khi break được thực thi thì "#code khác bên trong vòng lặp for" sẽ bị bỏ qua và chuyển đến "#code bên ngoài vòng lặp for".

Lệnh break

```
while dieu_kien_kiem_tra:  
    #code bên trong vòng lặp while  
  
    if dieu_kien:  
        break  
  
    #code khác bên trong vòng lặp while  
  
#code bên ngoài vòng lặp while
```

Lệnh break

```
# use the break and continue statements
for x in range (10,20):
    if (x == 15): break
    print(x)
```

```
for val in "string":
    if val == "i":
        break
    print(val)

print("Kết thúc!")
```

Lệnh continue

- Lệnh continue sẽ khiến việc lặp không thực thi ở vị trí lặp hiện tại, NHƯNG sẽ tiếp tục thực hiện các vòng lặp sau đó.

```
for var in sequence:
```

```
#khỏi code bên trong vòng lặp for
```

```
if dieu_kien:
```

```
    continue
```

```
#code khác bên trong vòng lặp for
```

```
#code bên ngoài vòng lặp for
```

Khi continue được thực thi thì "#code khác bên trong vòng lặp for" bị bỏ qua và quay trở lại "#Khỏi code bên trong vòng lặp for"

Lệnh continue

```
while dieu_kien_kiem_tra:  
    #code bên trong vòng lặp while  
  
    if dieu_kien:  
  
        continue  
  
    #code khác bên trong vòng lặp while  
  
#code bên ngoài vòng lặp while
```

Lệnh continue

```
for x in range (10,20):
    if (x % 5 == 0) : continue
    print(x)
```

```
for val in "string":
    if val == "i":
        continue
    print(val)

print("Kết thúc!")
```

Hàm enumerate

- **Cách sử dụng hàm "enumerate" cho "vòng lặp For"**
- Hàm enumerate trong “vòng lặp for” thực hiện hai điều:
 - Nó trả về giá trị chỉ số cho các phần tử
 - Và phần tử trong tập đang được xét.

```
#use a for loop over a collection
Months = ["Jan", "Feb", "Mar", "April", "May", "June"]
for i, m in enumerate (Months):
    print(i,m)
```

Khi chương trình được thực thi, đầu ra của hàm enumerate trả về tên tháng với chỉ số của tháng trong danh sách như (0-Jan), (1- Feb), (2 – Mar), v.v.

Function

- Cú pháp khai báo hàm

```
def <tên-hàm>(danh-sách-tham-số):  
    <lệnh 1>        def tich(a, b):  
    ...                return a*b  
    <lệnh n>
```

- Ví dụ: hàm tính tích 2 số

```
def tich(a, b):  
    return a*b
```

Hàm trả về kết quả bằng lệnh **return**, nếu không trả về thì coi như trả về **None**

Function

- Hàm có thể chỉ ra giá trị mặc định của tham số
 - Chú ý: các tham số có giá trị mặc định phải đứng cuối danh sách tham số

```
def tich(a, b = 1):  
    return a*b
```

```
print(tich(10, 20))      # 200  
print(tich(10))          # 10  
print(tich(a=5))         # 5  
print(tich(b=6, a=5))    # 30
```

Thực Hành

1. Viết chương trình nhập số A và kiểm tra xem A có phải là số nguyên tố hay không?
2. Viết chương trình in ra tất cả số chẵn trong khoảng (M,N) . N, M nhập từ bàn phím.

Nội Dung

- Cách tạo chuỗi
- Cách truy cập vào phần tử của chuỗi
- Các toán tử trên chuỗi
- Vòng lặp và kiểm tra phần tử của chuỗi
- Các hàm thông dụng trên chuỗi

Cách Tạo String

- Trong Python, chuỗi (string) là một dãy các ký tự Unicode.
- Chuỗi được để trong dấu nháy đơn ('...') hoặc kép ("...") với cùng một kết quả
- \ được sử dụng để "trốn (escape)" 2 dấu nháy này

```
a = 'hello world'  
print(a)  
print("tin hoc")  
print('doesn\'t') # sử dụng \' để viết dấu nháy đơn...  
print("\\"Yes, \" he said.") # sử dụng \" để viết dấu nháy doi...
```

Kết quả

hello world
tin hoc
doesn't
"Yes, " he said.

Cách Tạo String

```
s = 'First line.\nSecond line.' # \n nghĩa là dòng mới  
print(s)
```

```
First line.  
Second line.
```

```
ss = 'First line.\nSecond line.' # \n bo di ky tu xuong dong  
print(ss)
```

```
First line.\nSecond line.
```

Cách Tạo String

- Nếu không muốn các ký tự được thêm vào bởi \ được trình thông dịch hiểu là **ký tự đặc biệt** thì sử dụng chuỗi raw bằng cách **thêm r** vào trước dấu nháy đầu tiên:

```
print('C:\some\name') # ở đây \n là dòng mới!
```

```
C:\some
```

```
ame
```

```
print(r'C:\some\name') # thêm r trước dấu nháy
```

```
C:\some\name
```

Cách Tạo String

```
a = 'hello world'  
print(a)  
print("tin hoc")  
print('doesn\'t') # sử dụng \' để viết dấu nháy đơn...  
print("\\"Yes," he said.") # sử dụng \" để viết dấu nháy doi...
```

```
print("\n")  
s = 'First line.\nSecond line.' # \n nghĩa là dòng mới  
print(s)  
ss = 'First line.\n\nSecond line.' # \n bo di ky tu xuong dong  
print(ss)  
print("\n")
```

```
print('C:\some\name') # ở đây \n là dòng mới!  
print(r'C:\some\name') # thêm r trước dấu nháy
```

Cách Tạo String

- Chuỗi ký tự dạng chuỗi có thể viết trên nhiều dòng bằng cách sử dụng 3 dấu nháy: """...""" hoặc "...".
- Kết thúc dòng tự động bao gồm trong chuỗi, nhưng có thể ngăn chặn điều này bằng cách **thêm \ vào cuối dòng**.

```
print("""\
Usage: thingy [OPTIONS]
-h Display this usage message
-H hostname Hostname to connect to
""")
```

```
Usage: thingy [OPTIONS]
-h Display this usage message
-H hostname Hostname to connect to
```

Cách Tạo String

```
print("""\nUsage: thingy [OPTIONS] \\\n    -h Display this usage message\n    -H hostname Hostname to connect to\n""")
```

```
Usage: thingy [OPTIONS]           -h Display this usage message\n      -H hostname Hostname to connect to
```

Các

Đây là danh sách tất cả các ký tự thoát (escape sequence) được Python hỗ trợ:

Escape Sequence	Mô tả
\newline	Dấu gạch chéo ngược và dòng mới bị bỏ qua
\\\	Dấu gạch chéo ngược
\'	Dấu nháy đơn
\\"	Dấu nháy kép
\a	ASCII Bell
\b	ASCII Backspace
\f	ASCII Formfeed
\n	ASCII Linefeed
\r	ASCII Carriage Return
\t	ASCII Horizontal Tab
\v	ASCII Vertical Tab
\ooo	Ký tự có giá trị bát phân là ooo
\xHH	Ký tự có giá trị thập lục phân là HH

Cách Tạo String

```
print("C:\\Python32\\Quantrimang.com")
```

C:\\Python32\\Quantrimang.com

```
print("In dòng này\\nthành 2 dòng")
```

In dòng này

thành 2 dòng

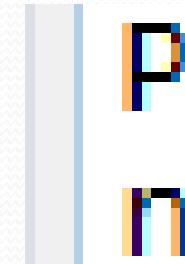
```
print(" In giá trị \x61 \x62 \x63")
```

In giá trị a b c

Truy Cập Phần Tử Của Chuỗi

- Có thể truy cập các ký tự trong chuỗi dựa vào chỉ số. Trong Python chỉ số bắt đầu từ 0.

```
word = 'Python'  
print(word[0]) # ký tự ở vị trí số 0  
print(word[5]) # ký tự ở vị trí số 5
```



```
File "C:/Users/sony/Desktop/python_co_ban/test/chuoi.py",  
line 44, in <module>  
    print(word[6])  
  
IndexError: string index out of range
```

Truy Cập Phần Tử Của Chuỗi

- Chỉ số cũng có thể là số âm, bắt đầu đếm từ bên phải:

```
word ='Python'  
print(word[-1]) # ky tu cuoi cung  
print(word[-2]) # ky ke^' cuoi
```



Lưu ý rằng vì -0 cũng tương tự như 0, nên các chỉ số âm bắt đầu từ -1.

	P	y	t	h	o	n
Index	0	1	2	3	4	5
	-6	-5	-4	-3	-2	-1

Truy Cập Phần Tử Của Chuỗi

- Trong khi index cũng được sử dụng để lấy chuỗi con:

```
print(word[0:2]) # các ký tự từ vị trí 0 (bao gồm) đến 2 (loại trừ)
```

Py

```
print(word[2:5]) # các ký tự từ vị trí 2 (bao gồm) đến 5 (loại trừ)
```

tho

Truy Cập Phần Tử Của Chuỗi

- Các chỉ số trong cắt chuỗi có thiết lập mặc định khá hữu ích, có 2 chỉ số bị bỏ qua theo mặc định, là 0 và kích thước của chuỗi được cắt.

```
print(word[:2]) # các ký tự từ đầu đến vị trí thứ 2 (loại bỏ)
```

Py

```
print(word[4:]) # các ký tự từ vị trí thứ 4(lấy) đến hết
```

on

```
print(word[-3:]) # các ký tự thứ ba tính từ cuối lên (lấy) đến hết  
hon
```

Các Toán Tử Trên Chuỗi

- Không thể thay đổi các ký tự trong chuỗi bằng phép gán '='

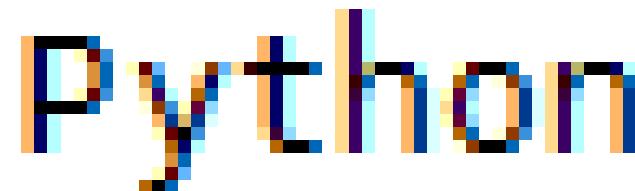
```
word = 'Python'  
word[0] = 'K'
```

```
File "C:/Users/sony/Desktop/python_co_ban/test/chuoi.py",  
line 55, in <module>  
    word[0]= 'K'  
  
TypeError: 'str' object does not support item assignment
```

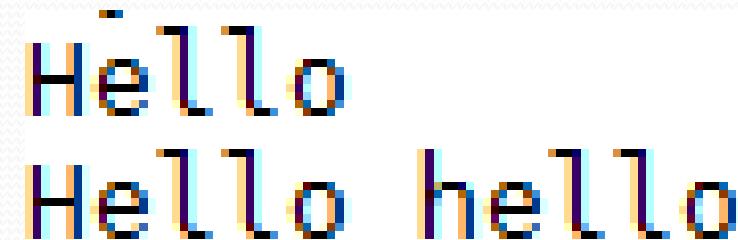
Các Toán Tử Trên Chuỗi

- **Nối hai chuỗi (+):** Các chuỗi có thể được nối với nhau bằng toán tử +

```
a ='Py'  
b ='thon'  
print(a+b)
```



```
x = "Hello World!"  
print(x[:6])  
print(x[0:6] + "hello")
```



Các Toán Tử Trên Chuỗi

- Lặp lại chuỗi (*): toán tử *

```
a ='Py'
```

```
b ='thon'
```

```
print(3*a)
```

```
print(b*5)
```

```
print(3*a+5*b)
```

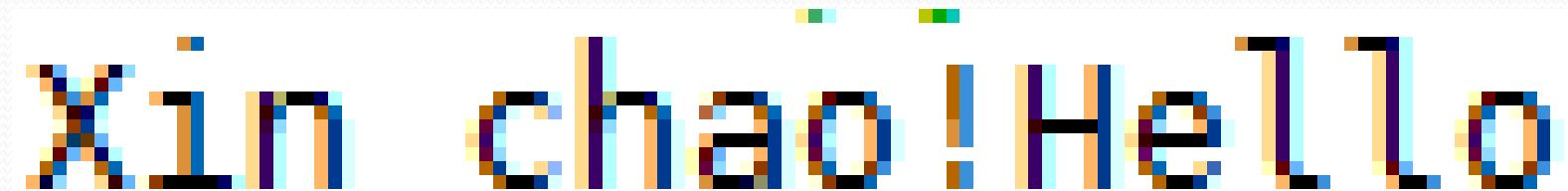
The diagram illustrates the result of string repetition. It shows three rows of text. The top row contains the string 'Py' repeated three times. The middle row contains the string 'thon' repeated five times. The bottom row contains the string 'Python' repeated five times. The letters are colored in a repeating pattern of blue, orange, and yellow.

Py Py Py
thon thon thon thon thon
Python Python Python Python Python

Các Toán Tử Trên Chuỗi

- Nếu muốn nối các chuỗi trong nhiều dòng khác nhau, hãy sử dụng dấu ngoặc đơn ():

```
s = ('Xin '
      'chao!'
      'Hello')
print(s)
```

A decorative graphic showing the words "Xin Chao ! Hello" composed of colored pixels. The letters are formed by a grid of small colored squares, primarily in shades of blue, green, and yellow.

Các Toán Tử Trên Chuỗi

- **Toán tử `in` và `not in`:**

- `in`: Kiểm tra tồn tại-trả về đúng nếu chữ cái tồn tại trong chuỗi cho trước.
- `not in`: Kiểm tra tồn tại-trả về đúng chữ cái không tồn tại trong chuỗi cho trước

```
ss = "hello world"
```

```
print("l" in ss) # in ra True
```

```
print('l' not in ss) # in ra False
```

```
print("k" in ss) # in ra False
```

```
print('k' not in ss) # in ra True
```

```
print("hello" in ss) # in ra True
```

```
print('hello' not in ss) # in ra False
```

Các Toán Tử Trên Chuỗi

- **Toán tử %:** dùng để định dạng chuỗi

```
name = 'hello'
```

```
number = 99999
```

```
fnum = 99.333333333333333333
```

```
print('In: %s %d %.5f' % (name,number,fnum))
```

In: hello 99999 99.33333

Vòng Lặp Và Chuỗi

- Có thể sử dụng vòng lặp for khi cần duyệt qua một chuỗi
- Ví dụ: đếm số ký tự "i" trong chuỗi dưới đây

```
count = 0
```

```
for letter in 'tin hoc, lap trinh python':
```

```
    if(letter == 'i'):
```

```
        count += 1
```

```
print('Co', count,'chu i duoc tim thay')
```

Co 2 chu i duoc tim thay

Vòng Lặp Và Chuỗi

- Ví dụ: nhập vào một số nguyên, in từ chữ số ra màn hình

```
n = 12345698765
```

```
ss = str(n) # chuyen so nguyen sang chuoi
```

```
for i in ss:
```

```
    print("%s" %i)
```

Các Hàm Thông Dụng

- Định dạng **.format()** và **dấu {}**: Python đã hỗ trợ hàm .format để thay cho %d, %s và các ký hiệu tương tự như vậy cho việc định dạng chuỗi. Việc sử dụng hàm format() đơn giản hơn dùng %d, %s ... Vì nó có thể được dùng cho nhiều loại dữ liệu khác nhau.

```
name = 'hello'
```

```
number = 100
```

```
fnum = 99.33333333333333
```

```
print('In: %s %d %.5f' % (name,number,fnum))
```

```
print('In: {} {} {}'.format(name, number,fnum))
```

```
print('In: {1} {2} {0}'.format(name, number,fnum))
```

```
print('In: {0} {1} {2:.2f}'.format(name, number,fnum))
```

```
In: hello 100 99.33333
```

```
In: hello 100 99.3333333333333
```

```
In: 100 99.3333333333333 hello
```

```
In: hello 100 99.33
```

Các Hàm Thông Dụng

```
name = 'hello'  
number = 100  
fnum = 99.33333333333333  
print('In: %s %d %.5f' % (name, number, fnum))  
print('In: {} {} {}'.format(name, number, fnum))  
print('In: {1} {2} {0}'.format(name, number, fnum))  
print('In: {0} {1} {2:.2f}'.format(name, number, fnum))  
  
In: hello 100 99.33333  
In: hello 100 99.33333333333333  
In: 100 99.33333333333333 hello  
In: hello 100 99.33
```

Các Hàm Thông Dụng

- Phương thức **format()** rất linh hoạt và đầy sức mạnh khi dùng để định dạng chuỗi.
- Định dạng chuỗi chứa dấu **{}** làm trình giữ chỗ hoặc trường thay thế để nhận giá trị thay thế.
- Bạn cũng có thể **sử dụng đối số vị trí hoặc từ khóa để chỉ định thứ tự**.

```
ss = "{}{}, {}{}".format('Hello','Hi','World')
print(ss)
```

Hello, Hi va World

Các Hàm Thông Dụng

sử dụng chỉ số để thay đổi vị trí

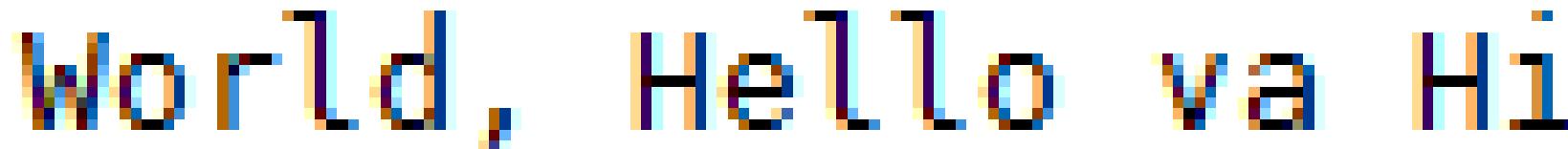
```
ss1 = "{1}, {0} và {2}".format('Hello','Hi','World')
print(ss1)
```



Hi, Hello và World

sử dụng từ khóa để sắp xếp thu tu

```
ss2 = "{c}, {a} và {b}".format(a='Hello',b='Hi',c='World')
print(ss2)
```



World, Hello và Hi

Các Hàm Thông Dụng

- Phương thức format() có thể có những **đặc tả định dạng tùy chọn**. Chúng được tách khỏi tên trường bằng dấu :
 - Có thể căn trái < , căn phải > , căn giữa ^ một chuỗi trong không gian đã cho.
 - Có thể định dạng số nguyên như số nhị phân, thập lục phân.
 - Số thập phân có thể được làm tròn hoặc hiển thị dưới dạng số mũ

```
s1= "Chuyen {0} sang he nhi phan la {0:b}" .format(6)      Chuyen 6 sang he nhi phan la 110  
print(s1)                                                 Chuyen 16 sang he bat phan la 20  
  
s2= "Chuyen {0} sang he bat phan la {0:o}" .format(16)    Chuyen 11 sang he thap luc phan la b  
print(s2)  
  
s3= "Chuyen {0} sang he thap luc phan la {0:x}" .format(11)  
print(s3)
```

Các Hàm Thông Dụng

```
t1 ="So thap phan {0} o dang so mua se la {0:e}" .format(2344.53535)  
print(t1)
```

So thap phan 2344.53535 o dang so mua se la 2.344535e+03

```
# căn chỉnh chuỗi
```

```
t2 = "|{:<10}|{:^10}|{:>20}|".format('Hello','Hi','World')  
print(t2)
```

| Hello | Hi | World |

Các Hàm Thông Dụng

- Hàm **len()**: lấy chiều dài của chuỗi

```
ss = "faodfjadfuioewufdojfkasdjfjk      jdskf"
```

```
print('chieu dai chuoi', len(ss))
```

chieu dai chuoi 39

- Hàm **enumerate()**: trả về đối tượng liệt kê, chứa cặp giá trị và index của phần tử trong string, khá hữu ích trong khi lặp

```
ss = "hello world"
```

```
for i,c in enumerate(ss):
```

```
    print('chi so la: {} va ky tu la \'{}\''.format(i,c))
```

Các Hàm Thông Dụng

```
ss ="hello world"  
for i,c in enumerate(ss):  
    print('chi so la: {} va ky tu la \'{}\''.format(i,c))
```

```
print(list(enumerate(ss)))  
[(0, 'h'), (1, 'e'), (2, 'l'), (3, 'l'), (4, 'o'), (5, ' '),  
(6, 'w'), (7, 'o'), (8, 'r'), (9, 'l'), (10, 'd')]
```

```
chi so la: 0 va ky tu la 'h'  
chi so la: 1 va ky tu la 'e'  
chi so la: 2 va ky tu la 'l'  
chi so la: 3 va ky tu la 'l'  
chi so la: 4 va ky tu la 'o'  
chi so la: 5 va ky tu la ' '  
chi so la: 6 va ky tu la 'w'  
chi so la: 7 va ky tu la 'o'  
chi so la: 8 va ky tu la 'r'  
chi so la: 9 va ky tu la 'l'  
chi so la: 10 va ky tu la 'd'
```

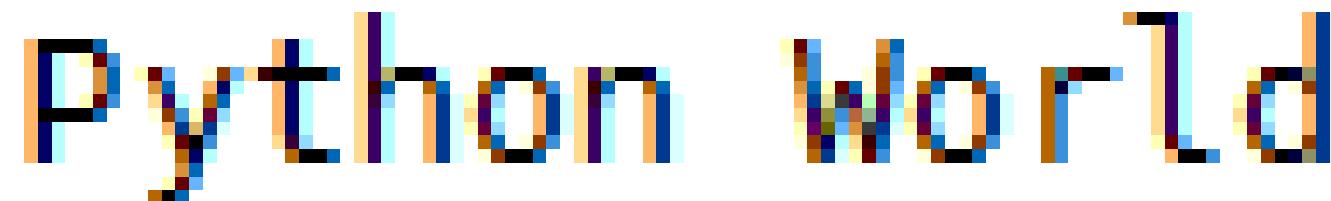
Các Hàm Thông Dụng

- **Thay thế chuỗi :** Phương thức thay thế **replace()** trả về một bản sao của chuỗi trong đó các giá trị của chuỗi cũ đã được thay thế bằng giá trị mới.

```
x = "hello World"
```

```
y = x.replace("hello","Python")
```

```
print(y)
```



Các Hàm Thông Dụng

- Thay đổi chữ hoa và chữ thường trong chuỗi

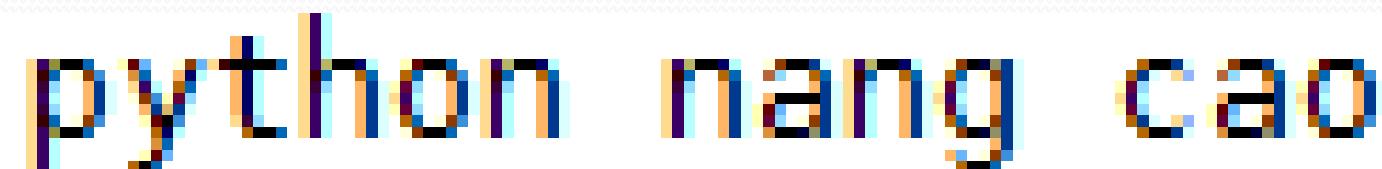
```
ss="python co ban"
```

```
# Chuyen chuoi sang chu hoa va in ra chuoi  
print(ss.upper())
```



```
xx="PYTHON NANG CAO"
```

```
# Chuyen chuoi sang chu THUONG va in ra chuoi  
print(xx.lower())
```



Các Hàm Thông Dụng

- Thay đổi chữ hoa và chữ thường trong chuỗi
 - Viết hoa chữ cái đầu tiên:

```
p="python anaconda"  
print(p.capitalize())
```

A decorative graphic consisting of two words, "Python" and "anaconda", rendered in a colorful, pixelated font. The letters are composed of various small colored squares, creating a mosaiced effect. The word "Python" is on the left, and "anaconda" is on the right, separated by a small gap.

Các Hàm Thông Dụng

- **Kết hợp chuỗi:** Hàm nối **join()** là một cách linh hoạt để kết hợp chuỗi. Với hàm join, bạn có thể thêm bất kỳ ký tự nào vào chuỗi.

```
print(":".join("Python")) P : y : t : h : o : n
```

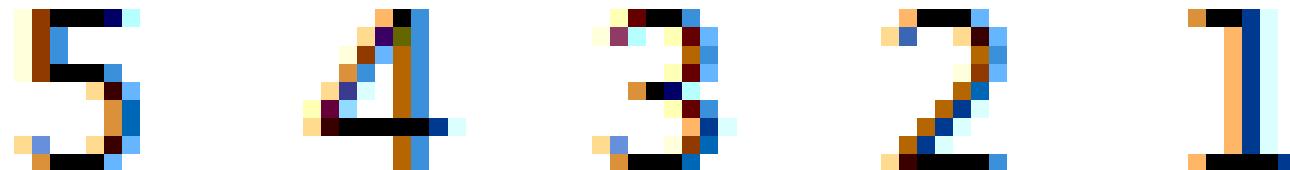
```
ss = '** '.join(['Quan', 'Tri', 'Mang', 'Cham', 'Com'])  
print(ss)
```

```
Quan**Tri**Mang**Cham**Com
```

Các Hàm Thông Dụng

- Đảo ngược chuỗi

```
string="12345"  
print(' '.join(reversed(string)))
```



Các Hàm Thông Dụng

- Phân tách chuỗi

```
word="hello hi python"  
print(word.split(' '))
```

```
[ 'hello', 'hi', 'python']
```

```
word="hello hi python"  
print(word.split('h'))
```

```
[ ' ', 'elllo', 'i pyton]
```

Các Hàm Thông Dụng

Các hàm khác:

- center()
- count()
- encode() và decode()
- find()
- isalpha(), isalnum() và isdigit()
- ...

<https://techblog.vn/bai-17-cac-ham-xu-ly-chuoi-trong-python>

Các Hàm Thông Dụng

Lưu ý:

```
x = "hello"  
x.replace("hello","Python")  
print(x)
```

```
x = "hello"  
x = x.replace("hello ","Python")  
print(x)
```

Thực Hành

1. Viết hàm tìm độ dài của một chuỗi (không dùng hàm len())
2. Viết chương trình đếm số lượng các từ trong một chuỗi (giả sử các từ trong chuỗi cách nhau bởi một khoảng trắng)
3. Viết hàm đếm số lượng nguyên âm(a e i o u), phụ âm có trong chuỗi