

Ngôn Ngữ Lập Trình Python

Chuỗi Ký Tự

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 đôi...
```

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 bỏ đi ký tự xuống dòng  
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 đôi...
```

```
print("\n")
s = 'First line.\nSecond line.' # \n nghĩa là dòng mới
print(s)
ss = 'First line.\\nSecond line.' # \\n bỏ đi ký tự xuống dòng
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("""\nUsage: thingy [OPTIONS]\n  -h Display this usage message\n  -H hostname Hostname to connect to\n""")
```

```
Usage: thingy [OPTIONS]\n  -h Display this usage message\n  -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
```


Cách Tạo String

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 gia tri \\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
```



```
print(word[6])
```

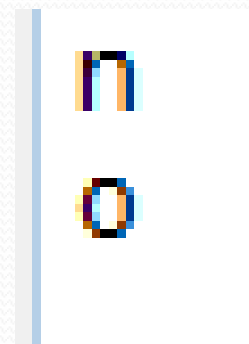
```
File "C:/Users/sony/Desktop/python_co_ban/test/chuoi.py",  
line 44, in <module>  
    print(word[6])
```

```
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]) # ký tự cuối cùng  
print(word[-2]) # ký ke^' cuối
```



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
```

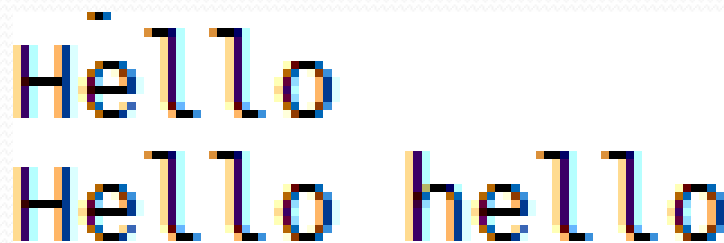
Các Toán Tử Trên Chuỗi

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

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

The word "Python" is displayed in a stylized, multi-colored font with a glitch or pixelated effect. The letters are composed of various shades of blue, purple, and orange, giving it a digital or cyberpunk appearance.

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

The text "Hello" and "Hello hello" are displayed in a stylized, multi-colored font with a glitch or pixelated effect. The letters are composed of various shades of blue, purple, and orange, giving it a digital or cyberpunk appearance.

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)
```

```
PyPyPy
```

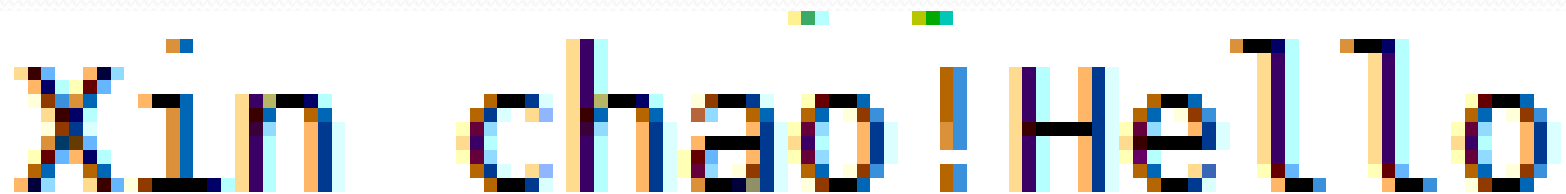
```
thonthonthonthonthon
```

```
PyPyPythonthonthonthon
```

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)
```



Xin chao! Hello

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 = 99.33333333333333333333333
```

```
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 = 99.333333333333333333
```

```
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 99.3333333333333333
```

```
In: 100 99.3333333333333333 hello
```

```
In: hello 100 99.33
```

Các Hàm Thông Dụng

```
name = 'hello'
number = 100
fnum = 99.33333333333333333333
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 99.3333333333333333
In: 100 99.33333333333333333333 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 = "{} , {} va {}".format('Hello','Hi','World')  
print(ss)
```

Hello , Hi va World

Các Hàm Thông Dụng

su dụng chỉ số để thay đổi vị trí

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

Hi, Hello và World

su dụng tu khoa để sắp xếp thứ tự

```
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)
```

```
print(s1)
```

```
s2= "Chuyen {0} sang he bat phan la {0:o}".format(16)
```

```
print(s2)
```

```
s3= "Chuyen {0} sang he thap luc phan la {0:x}".format(11)
```

```
print(s3)
```

Chuyen 6 sang he nhi phan la 110

Chuyen 16 sang he bat phan la 20

Chuyen 11 sang he thap luc phan la b

Các Hàm Thông Dụng

```
t1 = "Số thập phân {0} ở dạng số mũ sẽ là {0:e}".format(2344.53535)
print(t1)
```

Số thập phân 2344.53535 ở dạng số mũ sẽ là 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 = "faodfjadfuioewufdojfkasdjfk      jdskf"
```

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

chieu dài 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'
```

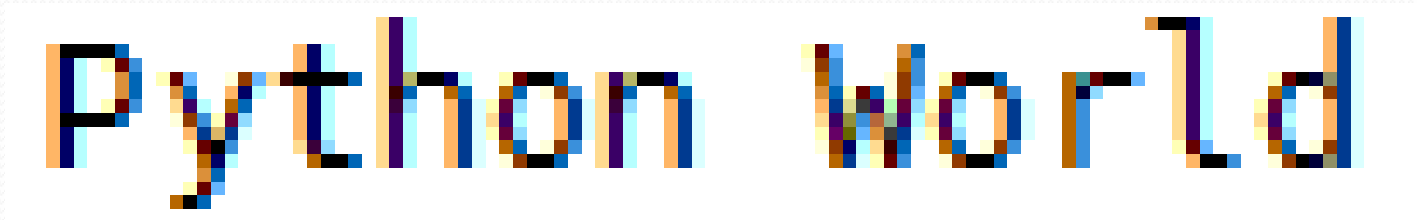
Các Hàm Thông Dụng

- **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)
```



Python World

Các Hàm Thông Dụng

- Thay đổi chữ hoa và chữ thường trong chuỗi

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

```
# Chuyển chuỗi sang chữ hoa và in ra chuỗi
```

```
print(ss.upper())
```

PYTHON CO BAN

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

```
# Chuyển chuỗi sang chữ THƯỜNG và in ra chuỗi
```

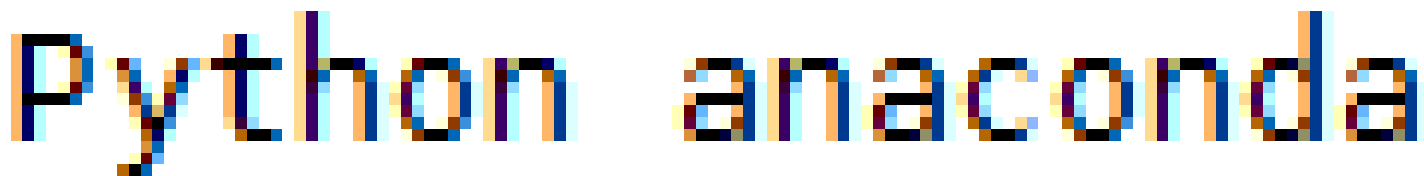
```
print(xx.lower())
```

python nang cao

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())
```

The image shows the output of the Python code: 'Python anaconda'. The word 'Python' is in a large, multi-colored font with a pixelated or glitch effect. The word 'anaconda' is in a smaller, black font. There are small colored squares above the 'P' and the 'a' in 'anaconda'.

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:õ:n

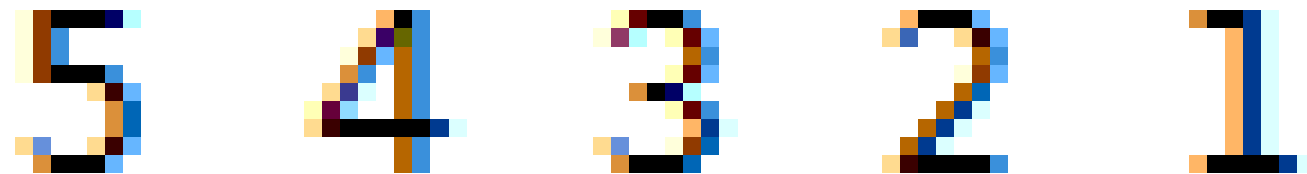
```
ss = '**'.join(['Quan', 'Tri', 'Mang', 'Cham', 'Com'])  
print(ss)
```

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

Các Hàm Thông Dụng

- Đảo ngược chuỗi

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



5 4 3 2 1

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'))
```

```
['', 'ello ', 'i pyt', 'on']
```

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)
```

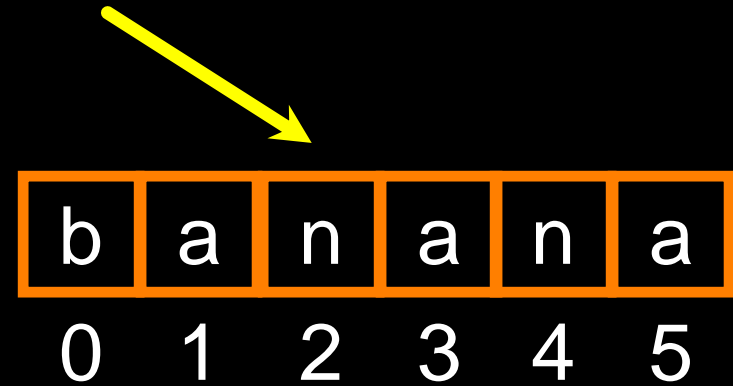
String Library

```
str.capitalize()  
str.center(width[, fillchar])  
str.endswith(suffix[, start[, end]])  
str.find(sub[, start[, end]])  
str.lstrip([chars])
```

```
str.replace(old, new[, count])  
str.lower()  
str.rstrip([chars])  
str.strip([chars])  
str.upper()
```

Searching a String

- We use the `find()` function to search for a substring within another string
- `find()` finds the first occurrence of the substring
- If the substring is not found, `find()` returns `-1`
- Remember that string position starts at zero



```
>>> fruit = 'banana'
>>> pos = fruit.find('na')
>>> print(pos)
2
>>> aa = fruit.find('z')
>>> print(aa)
-1
```


Making everything UPPER CASE

- You can make a copy of a string in **lower case** or **upper case**
- Often when we are searching for a string using **find()** we first convert the string to lower case so we can search a string regardless of case

```
>>> greet = 'Hello Bob'
>>> nnn = greet.upper()
>>> print(nnn)
HELLO BOB
>>> www = greet.lower()
>>> print(www)
hello bob
>>>
```

Search and Replace

- The `replace()` function is like a “search and replace” operation in a word processor
- It replaces **all occurrences** of the **search string** with the **replacement string**

```
>>> greet = 'Hello Bob'
>>> nstr = greet.replace('Bob', 'Jane')
>>> print(nstr)
Hello Jane
>>> nstr = greet.replace('o', 'X')
>>> print(nstr)
HellX Bxb
>>>
```

Stripping Whitespace

- Sometimes we want to take a string and remove whitespace at the beginning and/or end
- `lstrip()` and `rstrip()` remove whitespace at the left or right
- `strip()` removes both beginning and ending whitespace

```
>>> greet = '    Hello Bob    '  
>>> greet.lstrip()  
'Hello Bob '  
>>> greet.rstrip()  
'    Hello Bob'  
>>> greet.strip()  
'Hello Bob'  
>>>
```

Prefixes

```
>>> line = 'Please have a nice day'
>>> line.startswith('Please')
True
>>> line.startswith('p')
False
```

Parsing and Extracting

21
↓

31
↓

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
>>> data = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
>>> atpos = data.find('@')
>>> print(atpos)
21
>>> sppos = data.find(' ', atpos)
>>> print(sppos)
31
>>> host = data[atpos+1 : sppos]
>>> print(host)
uct.ac.za
```



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