TS. TRẦN QUANG QUÝ (tqquy@ictu.edu.vn)

Fundamental Python Programming

Trường Đại học Công nghệ Thông tin & Truyền thông Thái Nguyên.

1

Ngôn ngữ Python

1.1 Giới thiệu về ngôn ngữ Python

Python là ngôn ngữ lập trình cấp cao, đa mục đích được sử dụng rộng rãi. Ban đầu được thiết kế bởi Guido van Rossum vào năm 1991 và được duy trì, phát triển bởi tổ chức Python Software Foundation.

Guido Van Rossum bắt đầu nghiên cứu các sản phẩm dựa trên ứng dụng của mình vào tháng 12 năm 1989 ở Centrum Wiskunde & Informatica (CWI). Tại đây Guido van Rossum có cơ hội được tiếp xúc với ngôn ngữ lập trình ABC, trong quá trình làm việc ông đã nhận thấy ABC có rất nhiều vấn đề , tuy nhiên ông vẫn ưa chuộng những tính năng mà ABC mang lại.

Bằng sự sáng tạo và sự trợ giúp từ các cộng sự, ông đã kết hợp cú pháp của ngôn ngữ ABC và một số tính năng của nó để tạo ra ngôn ngữ lập trình mới. Cái tên được lấy cảm hứng từ một chương trình truyền hình BBC – Monty Python, Flying Circus Circus mà ông rất hâm mộ. Thêm vào đó, với ý muốn đặt một cái tên ngắn, độc đáo và hơi bí ẩn cho phát minh của mình, Guido đặt tên cho ngôn ngữ này là Python!

Ngôn ngữ Python

Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động; do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk, và Tcl. Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý.

Ban đầu, Python được phát triển để chạy trên nền Unix. Nhưng rồi theo thời gian, Python dần mở rộng sang mọi hệ điều hành từ MS-DOS đến Mac OS, OS/2, Windows, Linux và các hệ điều hành khác thuộc họ Unix. Mặc dù sự phát triển của Python có sự đóng góp của rất nhiều cá nhân, nhưng Guido van Rossum hiện nay vẫn là tác giả chủ yếu của Python. Ông giữ vai trò chủ chốt trong việc quyết định hướng phát triển của Python.

Guido van Rossum

1.2 Các phiên bản của Python

Sự phát triển Python đến nay có thể chia làm các giai đoạn:

- *Python 1:* bao gồm các bản phát hành 1.x. Giai đoạn này, kéo dài từ đầu đến cuối thập niên 1990. Từ năm 1990 đến 1995, Guido làm việc tại CWI (Centrum voor Wiskunde en Informatica - Trung tâm Toán-Tin học tại Amsterdam, Hà Lan). Vì vậy, các phiên bản Python đầu tiên đều do CWI phát hành. Phiên bản cuối cùng phát hành tại CWI là 1.2.
- *Python 2:* vào năm 2000, Guido và nhóm phát triển Python dời đến BeOpen.com và thành lập BeOpen PythonLabs team. Phiên bản Python 2.0 được phát hành tại đây. Sau khi phát hành Python 2.0, Guido và các thành viên PythonLabs gia nhập Digital Creations.
- *Python 3* còn gọi là Python 3000 hoặc Py3K: Dòng 3.x sẽ không hoàn toàn tương thích với dòng 2.x, tuy vậy có công cụ hỗ trợ chuyển đổi từ các phiên bản 2.x sang 3.x. Nguyên tắc chủ đạo để phát triển Python 3.x là "bỏ cách làm việc cũ nhằm hạn chế trùng lặp về mặt chức năng của Python".

1.3 Các lĩnh vực sử dụng Python

Các đặc trưng của Python giúp cho Python được sử dụng rộng rãi, phổ biến trong cộng đồng lập trình. Có đến hàng hàng ngàn thư viện hỗ trợ ngôn ngữ lập trình Python với mục tiêu nhất định

1.3.1 Web and internet development

Python cung cấp cho người dùng chúng ta rất nhiều công cụ phục vụ cho việc phát triển web, điển hình có thể kể đến như:

- Frameworks nổi bật như Django và Pyramid.
- Các thư viện như Flask và Bottle.
- Các hệ thống quản trị nội dung như Plone và django CMS.

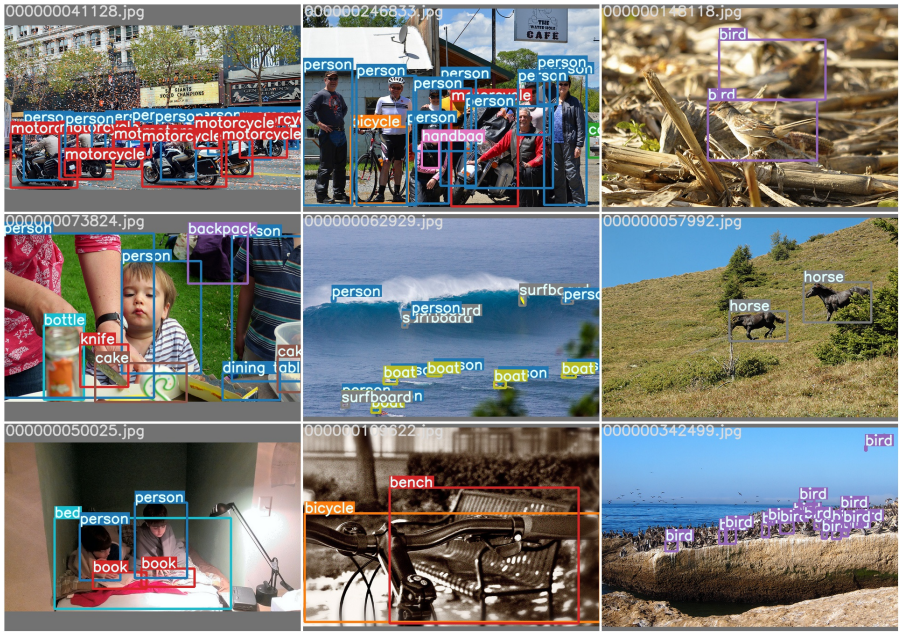
1.3.2 Scientific and numeric

Việc triển khai các thuật toán học máy (Machine Learning) yêu cầu tính toán toán học vô cùng phức tạp. Rất may Python cung cấp cho chúng ta rất nhiều thư viện cho khoa học và số liệu như Numpy, Pandas, Scipy, Scikit-learning, v.v. Dưới đây là một vài thư viện phổ biến trong lĩnh vực này.

- SciPy
- Scikit-learn
- NumPy
- Pandas
- Matplotlib

1.3.3 Artificial Intelligence

Triển khai các dự án về AI liên quan đến hàng tấn thuật toán. Để tiết kiệm thời gian và công sức,chúng ta không cần phải code lại các thuật toán cũng như để cho mọi thứ dễ dàng hơn, Python cung cấp cho chúng ta hơn 100 thư viện được xây dựng sẵn để thực hiện các thuật toán khác nhau. Vì vậy, mỗi khi bạn muốn chạy một thuật toán trên một tập dữ liệu, tất cả những gì bạn phải làm là cài đặt và tải các thư viện cần thiết bằng một lệnh duy nhất. Ví dụ về các thư viện nổi bật trong lĩnh vực này như NumPy, Keras, Tensorflow, Pytorch, v.v.



Object Detection sử dụng YOLOv3

1.4 Cài đặt Python

Điều đầu tiên và quan trọng nhất, để có thể chạy được một chương trình Python trên máy tính của mình thì ta cần phải cài đặt môi trường có thể đọc và chạy được Python. Chúng ta vào trang chủ của Python thông qua GOOGLE và download bản cài đặt mới nhất của Python.

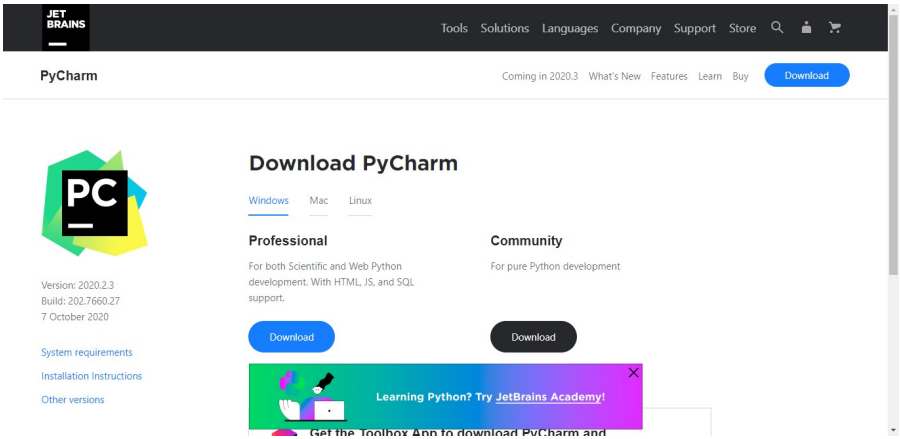
Cài đặt Python

Sau khi đã cài đặt xong môi trường Python cho máy tính của mình, bây giờ chúng ta đã có thể viết code Python ở bất cứ các Text Editor (trình soạn thảo văn bản) nào như Text Document hay Notepad, xịn hơn nữa các có thể dùng Sublime Text, VS Code,...

Nhưng để có thể phát triển cũng như viết chương trình Python một cách hiệu quả và nhanh chóng hơn, người ta thường sẽ lựa chọn các IDEs

(môi trường phát triển tích hợp), Trong số các Python IDE đang có nhiều người dùng hiện nay có thể kể đến như PyCharm, Spyder, PyDev, Atom,... thì PyCharm có lẽ là IDE được ưa chuộng cũng như phổ biến nhất vì những tính năng hỗ trợ viết code mạnh mẽ, giao diện người dùng thân thiện,...

Truy cập vào trang chủ của JetBrains để tải về PyCharm phiên bản mới nhất. Sau khi tải về, chạy file .exe vừa tải được và bắt đầu quá trình cài đặt PyCharm



Pycharm

2

Nhập/xuất dữ liệu trong Python

2.1 Nhập/xuất dữ liệu

2.1.1 Lệnh print

Chương trình đầu tiên trong Python chính là in ra dòng chữ Hello World, những ai khi bắt đầu học lập trình đều quen thuộc với nội dung này. Trong các ngôn ngữ lập trình thì nó luôn luôn tồn tại một hoặc nhiều hàm được cùng để hiển thị dữ liệu ra khi chương trình chạy. VD như trong C thì có `printf`, trong C++ thì có `cout`,... Và đối với python thì nó là `print`.

Để có thể in một nội dung ra màn hình trong Python chúng ta sử dụng lệnh print(). Nếu trong đó có nhiều hơn một nội dung thì chúng ta sử dụng dấu phẩy giữa các nội dung. Ngoài ra chúng ta có thể thêm tham số **sep** trong hàm print() để có thể thay đổi giá trị khoảng cách.

```
print ('Hello world!')
```

```
## Hello world!
```

```
print('Welcome', 'to', 'Python!')
```

```
## Welcome to Python!
```

```
print('Welcome', 'to', 'Python!', sep ='--')
```

```
## Welcome--to--Python!
```

2.1.2 Nhập dữ liệu từ bàn phím

Để nhập dữ liệu từ bàn phím trong Python, chúng ta sử dụng phương thức input(). Ví dụ chương trình sau, nhập một dãy ký tự và in ra màn hình

```
#print("Mời bạn nhập vào nội dung gì đó: ")
#x = input()
#print("Nội dung bạn nhập vào là: ",x)
```

Chú ý: Câu lệnh input() trả về một chuỗi ký tự, do đó nếu chúng ta muốn kết quả trả về là số nguyên thì ta cần ép kiểu như trên. Ví dụ: int("123") sẽ chuyển ký tự "123" thành số 123

Trong Python, để chú thích một câu lệnh, chúng ta sử dụng dấu # để chú thích một dòng code hoặc 3 dấu nháy """ để cho khối lệnh code.

Bài tập áp dụng: Nhập vào hai số nguyên a, b từ bàn phím. Viết chương trình in ra tích, tổng, hiệu và thương của hai số này.

2.2 Khai báo biến trong Python

2.2.1 Khái niệm về biến

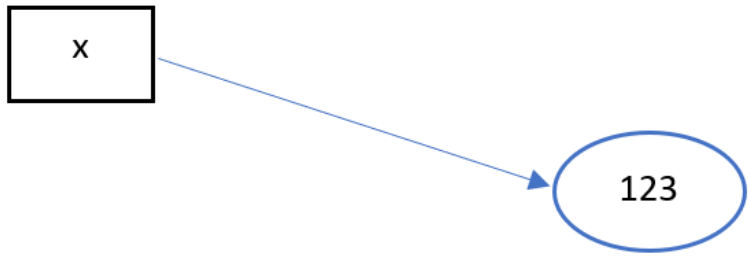
Biến(variable) là đối tượng mà giá trị của nó có thể thay đổi được khi thực hiện chương trình. Mỗi biến cần được đặt tên và cần một lượng ô nhớ máy tính tương ứng để lưu giá trị cho biến. Trong Python mỗi biến chính là một con trỏ chỉ đến ô nhớ chứa giá trị đã được gán cho biến đó. Khác với các ngôn ngữ lập trình khác, Python không có lệnh khai báo biến mà chỉ có

lệnh gán trực tiếp giá trị cho biến theo cách: tên biến = giá trị. Các ô nhớ trong máy tính được đánh địa chỉ bắt đầu từ số 0, để lấy được địa chỉ của ô nhớ, sử dụng phương thức **id(tên biến)**.

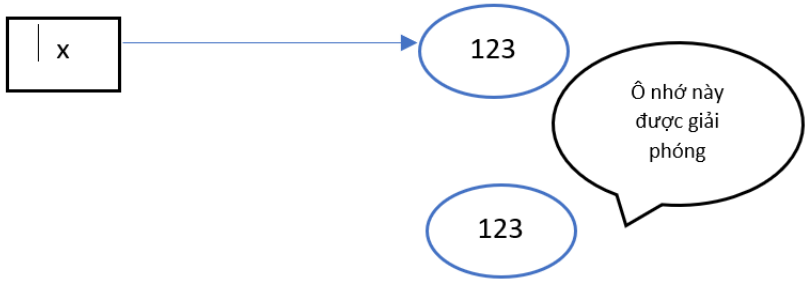
```
# Gán giá trị cho biến x
x = "Learning Python is interesting"
print(x)
#print("Địa chỉ ô nhớ của x là:", id(x))
```

```
## Learning Python is interesting
```

Chú ý: Cùng một chương trình, mỗi lần thực hiện, địa chỉ ô nhớ của biến có thể khác nhau, lý do là do cơ chế cấp phát bộ nhớ cho biến trong Python. Ví dụ: Khi gán x=123, máy tính sẽ sử dụng một ô nhớ nào đó trong bộ nhớ RAM (Random Access Memory) để lưu trữ giá trị 123 và biến x thành con trỏ chỉ tới ô nhớ chứa giá trị 123. Khi ta gán x=321, máy tính sẽ dùng một ô nhớ khác để lưu giá trị 321 và dùng x là con trỏ chỉ tới ô nhớ mới, ô nhớ chứa giá trị 123 khi này sẽ được giải phóng nếu không có biến nào chỉ tới nó.



Cơ chế gán giá trị cho biến



Cơ chế gán giá trị cho biến

Trong trường hợp nhiều biến có cùng giá trị thì các biến này cùng chỉ tới một ô chứa giá trị đó.

```
a = 123
b = 123
print(id(a))
```

```
## 627333296
```

```
print(id(b))
```

```
## 627333296
```

2.2.2 Đặt tên biến

Việc đặt tên biến được quy định bởi từng ngôn ngữ lập trình, trong Python được quy định như sau:

1. Biến có thể chứa các ký tự chữ và ký tự số
2. Không được chứa các ký tự trống và các ký tự đặc biệt như +,-,*,/...
3. Tên biến không được bắt đầu bằng ký tự số
4. Không sử dụng các từ khóa, tên hàm đã được xây dựng trong Python

5. Nên đặt tên ngắn cho dễ nhớ

Ví dụ:

- Tên đúng: x,y,x1,x2
- Tên sai: 1x, 2x, a b, de f.

2.2.3 Gán giá trị cho biến rút gọn

Python cung cấp cho người lập trình cách viết ngắn gọn các biểu thức gán giá trị, rất ngắn gọn và dễ sử dụng, xem bảng sau:

Cách gán giá trị rút gọn cho biến

Cách gán thông thường	Cách gán rút gọn
a=2; b=2; c=2	a=b=c=2
a=1; b=2; c=3	a,b,c=1,2,3
Tráo đổi giá trị của a và b x=a; a=b; b=x	a,b=b,a

Trong python, để kiểm tra kiểu dữ liệu của một biến thì chúng ta có thể sử dụng hàm type với cú pháp như sau: **type(biến cần kiểm tra)**

2.3 Kiểu dữ liệu số

2.3.1 Các dạng dữ liệu số

Ngôn ngữ lập trình Python hỗ trợ 3 kiểu dữ liệu về số, đó là:

- Số nguyên - int
- Số thực - float
- Số phức - complex

2.3.1.1 Kiểu số nguyên - int

Python cho phép thực hiện các phép toán cơ bản với kiểu số nguyên và ngoài ra còn cho phép tính toán với các số nguyên có giá trị rất lớn

```
a=2020
n=50
s=a**n
print(s)
```

```
##
185169081500444162204657163114373271945042602530684332459093029305655
```

Phép toán	Ký hiệu	Ví dụ
Cộng	+	5+2=7
Trừ	-	5-2+3
Nhân	*	5*2=10
Lũy thừa	**	5**2=25
Chia	/	5/2=2.5
Chia lấy thương	//	5//2=2
Chia lấy dư	%	5%2=1

2.3.1.2 Kiểu số thực - float

Xét chương trình tính diện tích hình tròn

```
pi = 3.14
r = 2
s = pi*r*r
print("Dien tich hinh tron la:", s)
```

```
## Dien tich hinh tron la: 12.56
```

Các phép toán thực hiện trên số thực cũng giống như các phép toán thực hiện trên kiểu số nguyên. Trong Python cho phép viết các số thực dưới dạng kí pháp khoa học của nó, chẳng hạn như $3.0 * 10^3$ có thể viết là $3.0e - 3$

2.3.1.3 Kiểu số phức - complex

Một số phức là một số có dạng $a + bj$. Trong đó a, b là các số thực; a được gọi là phần thực và b được gọi là phần ảo

```
x = 2 + 1j
y = 3 - 2j
print("Tong bang: ", x+y)
```

```
## Tong bang:  (5-1j)
```

```
print("Hieu bang: ", x-y)
```

```
## Hieu bang:  (-1+3j)
```

```
print("Tich bang: ", x*y)
```

```
## Tich bang:  (8-1j)
```

```
z = complex(1,2)
t = complex(2)
print(z)
```

```
## (1+2j)
```

```
print(t)
```

```
## (2+0j)
```

2.3.1.4 Hàm tạo số ngẫu nhiên

Python cung cấp mô-đun **random** cho phép sinh các số ngẫu nhiên cả về số nguyên và số thực.

- Hàm **randint(a,b)** tạo ngẫu nhiên một số nguyên thuộc [a,b]
- Hàm **uniform(a,b)** tạo ngẫu nhiên một số thực [a,b].

Để sử dụng hai câu lệnh này, chúng ta cần import chúng từ mô-đun **random** bằng lệnh **from random import randint, uniform**

```
from random import randint, uniform
x = randint(-10, 10)
y = uniform(-100, 100)
print("So x duoc tao ngau nhien la: ", x)
```

```
## So x duoc tao ngau nhien la:  0
```

```
print("So y duoc tao ngau nhien la: ", y)
```

```
## So y duoc tao ngau nhien la:  -0.18020991037333545
```

Một số hàm toán học thông dụng khác có thể xem tại bảng sau:

Tên hàm	Thực hiện
$sin(x)$	Tính sin của một góc có số đo x(rad)
$cos(x)$	Tính cô sin của một góc có số đo x(rad)
$tan(x)$	Tính tang của một góc có số đo x(rad)
$sqrt(x)$	Tính căn bậc hai của x
$trunc(x)$	Tính phần nguyên của x
pi	Số pi
$round(x, n)$	Làm tròn x đến n chữ số thập phân
$abs(x)$	Tính giá trị tuyệt đối của x
$max(a_1, a_2, \ldots, a_n)$	Trả về giá trị lớn nhất
$min(a_1, a_2, \ldots, a_n)$	Trả về giá trị nhỏ nhất

Để import tất cả các hàm trong mô-đun math, ta thực hiện: from math import *

Các bài tập áp dụng:

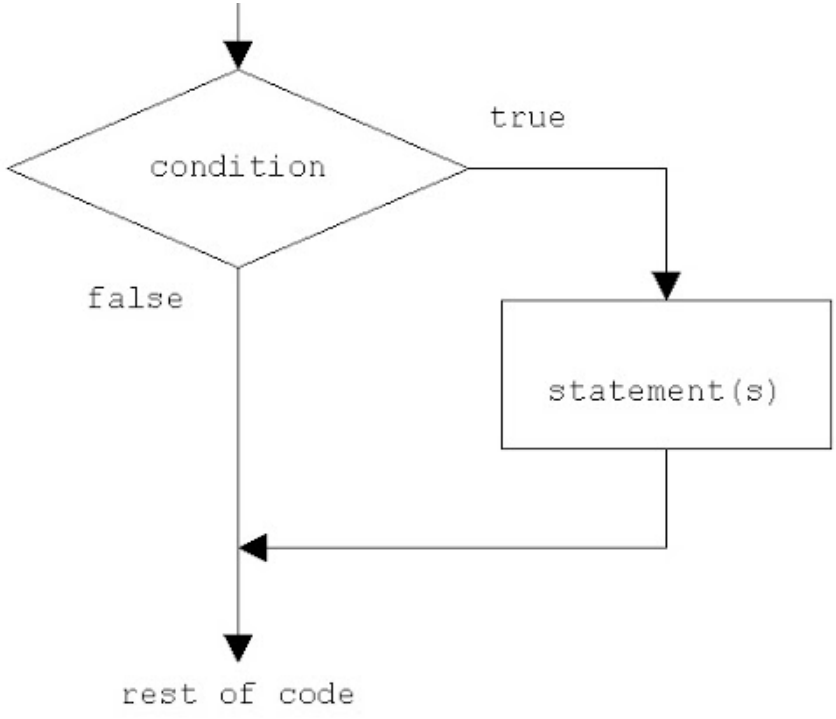
1. Bài 1: Nhập số nguyên dương N có 3 chữ số. Tính tổng các chữ số của N
2. Bài 2: Nhập vào 3 số nguyên dương a,b,c. Hãy tìm ước chung lớn nhất của 3 số này (GCD - Greatest Common Divisor)

3

Cấu trúc rẽ nhánh

3.1 Câu lệnh If

Cú pháp của lệnh if như sau: Nếu điều kiện đúng thì sẽ thực hiện khối lệnh.



Câu lệnh if

```
a = 10
b = 20
if a < b:
  print("a nho hon b")
  print("Tong a va b la: ", a + b)
```

```
## a nho hon b
## Tong a va b la:  30
```

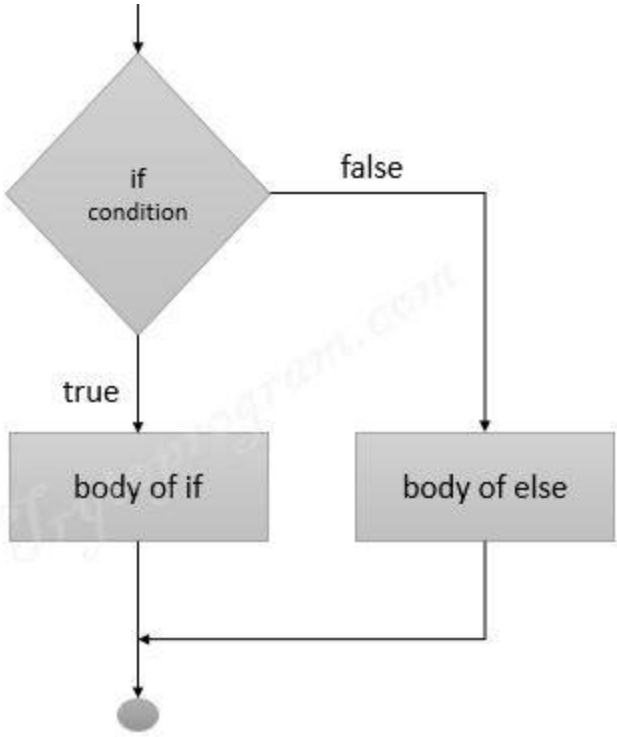
```
a = 30
b = 20
if a < b:
  print("a nho hon b")
  print("Tong a va b la: ", a + b)

print("Vi tri ngoai khoi lenh tren!")
```

```
## Vi tri ngoai khoi lenh tren!
```

Khái niệm về khối lệnh - codeblock: Khối lệnh trong Python được định nghĩa là một nhóm các lệnh liên tiếp nhau và được viết thẳng cột với nhau. Các câu lệnh này được gọi là đồng mức (same level).

3.2 Câu lệnh if-else

Trong câu lệnh này, nếu điều kiện là đúng thì thực hiện khối lệnh 1, còn ngược lại (điều kiện sai) thì thực hiện khối lệnh 2. Sơ đồ thực hiện như sau:



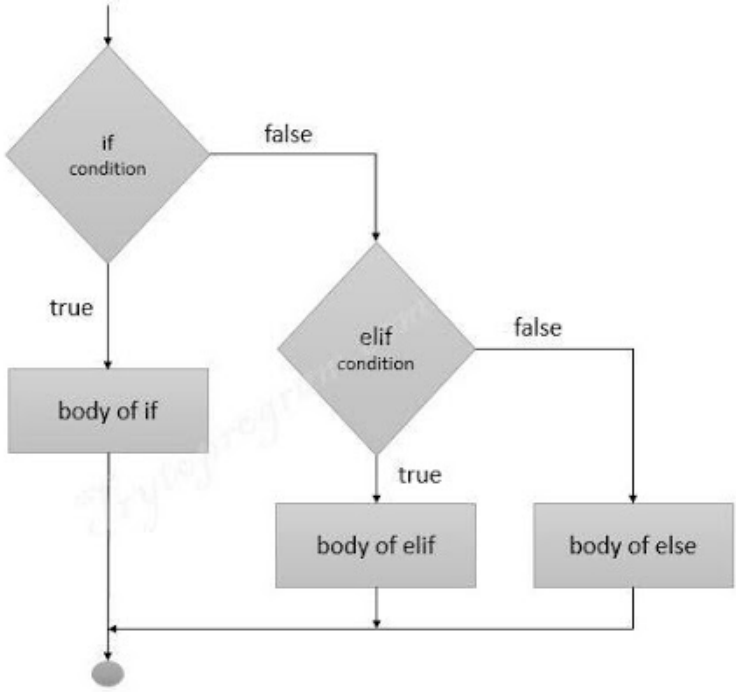
Câu lệnh if-else

```
a = 20
b = 20
if a == b:
  print("a bang b")
else:
  print("a khong bang b")
```

```
## a bang b
```

3.3 Câu lệnh if-elif

Nếu điều kiện 1 đúng thì thực hiện khối lệnh 1, ngược lại nếu điều kiện 2 đúng thì thực hiện khối lệnh 2.



Câu lệnh elif

```
a = 200
b = 50
if a==b:
  print("a bang b")
elif a<b:
  print("a nho hon b")
else:
  print("a lon hon b")
```

```
## a lon hon b
```

Các phép toán logic.

a	b	**not** a	a **and** b	a **or** b
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE

Ví dụ: Kiểm tra điều kiện ba số dương có là ba cạnh của một tam giác hay không.

```
a = 10
b = 20
c = 15
if (a+b>c) and (b+c>a) and (c+a>b):
  print("a,b,c la ba canh cua mot tam giac")
else:
  print("a,b,c khong phai la ba canh cua mot tam giac")
```

```
## a,b,c la ba canh cua mot tam giac
```

Chú ý: Các phép toán lô gic có thứ tự ưu tiên thực hiện thấp hơn so với các phép toán số học và phép toán so sánh. Do đó trong biểu thức so sánh trong biểu thức điều kiện trên, ta có thể viết:

if(a+b>c) and (b+c>a) and (a+c>b) có thể viết được là if a+b>c and b+c>a and a+c>b

Các bài tập áp dụng:

Bài 1: Nhập vào 3 số a,b,c (a#0). Giải phương trình bậc hai $ax^2 + bx + c = 0$.

Bài 2: Nhập vào ba số a,b,c nguyên dương. Kiểm tra xem a,b,c có phải ba cạnh của một tam giác hay không? Nếu là ba cạnh của một tam giác hãy phân loại xem tam giác có 3 cạnh a,b,c đó thuộc loại nào dưới đây: 1. Tam giác đều; 2. Tam giác vuông; 3. Tam giác thường.

Bài 3: Giao đoạn - Nhập vào bốn số a,b,c,d (a<b,c<d). Tính xem có bao nhiêu số nguyên thuộc đoạn giao [a;b] và [c;d].

Bài 4: Giao hai hình chữ nhật - Cho hai hình chữ nhật có cạnh song song hoặc trùng với các trục tọa độ. Hình chữ nhật thứ nhất được xác định bởi điểm trái dưới $A(x_A, y_A)$ và điểm bên phải trên $B(x_B, y_B)$. Hình chữ nhật thứ hai được xác định bởi điểm trái dưới $C(x_C, y_C)$ và điểm phải trên $D(x_D, y_D)$. Xét xem hai hình chữ nhật có giao nhau hay không? Nếu giao nhau (có điểm trong chung) thì tính diện tích phần giao nhau.

Bài 5: Xếp loại học sinh Cho điểm thi Toán, Lý, Hóa , Tin của học sinh. Hãy xếp loại kết quả của học sinh theo tiêu chí

Điểm trung bình	Xếp loại
ĐTB>=9	Xuất sắc
8<=ĐTB<9	Giỏi
7<=ĐTB<8	Khá
5<=ĐTB<7	Trung bình
ĐTB<5	Yếu

4

Cấu trúc lặp

4.1 Định nghĩa công việc lặp

Trong quá trình thực hiện một công việc nào đó, chúng ta sẽ bắt gặp những quá trình mà các công việc được thực hiện giống nhau lặp đi lắp lại. Các công việc đó có thể là giống nhau hoặc chỉ khác nhau về giá trị tính toán. Thường có hai loại lặp đó là:

- Lặp với số lần lặp xác định trước
- Lặp với số lần lặp chưa được xác định trước

4.2 Vòng lặp for

Cấu trúc của vòng lặp for trong Python như sau:

```
# for bien_lap in chuoi_lap:
# <Khoi lenh for>
```

Trong cú pháp trên, chuoi_lap là chuỗi cần lặp, bien_lap là biến nhận giá trị của từng mục bên trong chuoi_lap trên mỗi lần lặp. Vòng lặp sẽ tiếp tục cho đến khi nó lặp tới mục cuối cùng trong chuỗi.

Khối lệnh của for được thụt lề để phân biệt với phần còn lại của code.

```
#Lap ky tu trong chuoi ICTU
for chu in 'ICTU':
    print('Chu cai hien tai la:', chu)

#Lap tu trong mot chuoi
```

```
## Chu cai hien tai la: I
## Chu cai hien tai la: C
## Chu cai hien tai la: T
## Chu cai hien tai la: U
```

```
chuoi = ['ICTU','vi','yeu', 'ma', 'den']
for tu in chuoi:
    print('Toi yeu', tu)
```

```
## Toi yeu ICTU
## Toi yeu vi
## Toi yeu yeu
## Toi yeu ma
## Toi yeu den
```

4.2.1 Hàm range

Chúng ta có thể sử dụng hàm range() để tạo ra một dãy số. Ví dụ, range(100) sẽ tạo một dãy số từ 0 đến 99 (100 số). Hàm range(số bắt đầu, số kết thúc, khoảng cách giữa hai số) được sử dụng để tạo dãy số tùy chỉnh. Nếu không đặt khoảng cách giữa hai số thì Python sẽ hiểu mặc định nó bằng 1.

Ví dụ: Viết 10 lần "Anh xin lỗi", ta sẽ cho biến i lặp từ 0 đến 10 như dưới đây:

```
for i in range(10):
    print ("Anh xin loi em!")
```

```
## Anh xin loi em!
## Anh xin loi em!
## Anh xin loi em!
## Anh xin loi em!
## Anh xin loi em!
## Anh xin loi em!
## Anh xin loi em!
## Anh xin loi em!
## Anh xin loi em!
## Anh xin loi em!
```

```
print("Em oi, anh chep xong 10 lan roi!")
```

```
## Em oi, anh chep xong 10 lan roi!
```

Hàm range() không lưu tất cả các giá trị trong bộ nhớ mà nó lưu giá trị bắt đầu, giá trị kết thúc và khoảng cách giữa hai số từ đó tạo ra số tiếp theo trong dãy. Hàm range() trả về một biến chứa một dãy các giá trị nằm trong một khoảng xác định theo quy luật. Hàm này có 3 dạng:

- range(stop): nhận 1 tham số là giá trị cuối của dãy và trả về dãy số [0, stop – 1]. Ví dụ range(5) sẽ trả lại dãy số 0, 1, 2, 3, 4. Bước nhảy giữa các giá trị kế tiếp là 1.
- range(start, stop): nhận 2 tham số là giá trị đầu và cuối của dãy, trả về dãy [start, stop-1]. Ví dụ, range(1, 5) sẽ trả lại dãy số 1, 2, 3, 4 (bước nhảy giữa các giá trị kế tiếp là 1).
- range(start, stop, step): tương tự như trường hợp 2 nhưng bước nhảy được xác định bởi biến step. Ví dụ, range(1, 10, 2) sẽ trả lại dãy 1, 3, 5, 7, 9 (bước nhảy giữa các giá trị kế tiếp là 2).

Trong hàm range, start, stop và step có thể nhận cả giá trị âm và dương:

- Nếu step nhận giá trị âm (và start > stop) sẽ tạo ra dãy giảm dần. Ví dụ range(0, -5, -1) sẽ tạo ra dãy 0, -1, -2, -3, -4.
- Nếu stop < start và step dương, hoặc stop > start và step âm, sẽ tạo ra chuỗi trống (không có giá trị nào). Ví dụ, range(2, -3, 1) và range(2, 3, -1) đều không tạo ra phần tử nào.

4.2.2 Sử dụng vòng lặp for kết hợp

```
companies = ['Apple', 'Google', 'Microsoft', 'Huawei',
        'Yandex', 'Facebook']
for c in companies:
    print(c)
```

```
## Apple
## Google
## Microsoft
## Huawei
## Yandex
## Facebook
```

Trong ví dụ trên, companies là một biến kiểu danh sách (list). Kiểu dữ liệu này chúng ta chưa học nhưng bạn có thể hình dung nó tương tự như kiểu mảng của C. Biến kiểu list cũng có thể trở thành danh sách dữ liệu cho vòng lặp for. Khi này biến chạy c sẽ lần lượt nhận các giá trị trong danh sách theo thứ tự.

```
total = 0
for num in (-22.0, 3.5, 8.1, -10, 0.5):
    if num > 0:
        total = total + num
```

Trong ví dụ này chúng tính tổng các số dương từ một tuple. Tuple cũng là một kiểu dữ liệu tập hợp mà chúng ta sẽ học chi tiết sau.

4.3 Cấu trúc lặp while

Xét ví dụ sau:

```
sum = 0
i = 1
while i < 100:
    sum += i
    i += 1
print(sum)
```

```
## 4950
```

Trong ví dụ này chúng ta tính tổng các số từ 1 đến 99. Để thực hiện việc này chúng ta sử dụng một biến i với vai trò biến đếm. Biến i ban đầu nhận giá trị 1. Qua mỗi vòng lặp i sẽ tăng thêm 1 đơn vị, và biến sum sẽ cộng thêm giá trị bằng i. Quá trình này sẽ lặp đi lặp lại chừng nào i vẫn nhỏ hơn 100. Đến khi i đạt giá trị 100 thì vòng lặp kết thúc.

Cấu trúc while của Python là một lệnh phức hợp với một mệnh đề. Trong đó, header là while :. Biểu thức logic là loại biểu thức mà kết quả trả về thuộc kiểu bool (giá trị True hoặc False). Trong các ví dụ trên, biểu thức logic là i < 100 và char.lower() != 'close'.

Chừng nào biểu thức logic còn nhận giá trị True thì sẽ thực hiện danh sách lệnh ở phần tiếp. Để tránh tạo thành vòng lặp vô tận, bên trong danh sách lệnh của vòng lặp while phải có lệnh có tác dụng làm thay đổi giá trị của biểu thức logic.

Trong ví dụ thứ nhất, chúng ta liên tục yêu cầu người dùng nhập vào giá trị mới cho biến char. Điều này đảm bảo khả năng thoát khỏi vòng lặp.

Trong ví dụ thứ hai, chúng ta làm biến đổi giá trị của i qua mỗi vòng lặp (tăng thêm 1 đơn vị). Điều này đảm bảo đến một lúc nào đó i sẽ có giá trị lớn hơn 100 và kết thúc vòng lặp.

4.4 Điều khiển vòng lặp, break và continue

Khi làm việc với vòng lặp có thể xuất hiện hai tình huống:

1. Bạn muốn kết thúc vòng lặp sớm hơn dự định hoặc theo điều kiện. Lấy ví dụ, bạn cầm một danh sách khách hàng và muốn tìm tên một khách trong đó. Bạn xem lần lượt từng tên từ đầu danh sách. Nếu gặp tên khách hàng cần tìm, bạn sẽ dừng lại chứ không tiếp tục xem đến hết danh sách nữa. Ở đây bạn phá vỡ hoàn toàn vòng lặp.

2. Bạn muốn bỏ qua một chu kỳ trong vòng lặp để bắt đầu chu kỳ tiếp theo. Lấy ví dụ, bạn dự định chơi 10 ván bài. Ở ván thứ 3 bài xấu quá, bạn quyết định bỏ cuộc để chờ chơi ván tiếp theo. Ở đây bạn không dừng hoàn toàn vòng lặp mà chỉ bỏ không làm những việc nhẽ ra phải làm trong chu kỳ đó và khởi động một chu kỳ mới.

Hai tình huống này được Python giải quyết bằng hai lệnh tương ứng: break (phá vỡ vòng lặp, kết thúc sớm) và continue (bỏ qua chu kỳ hiện tại, bắt đầu một chu kỳ lặp mới). Cả hai loại vòng lặp (for, while) đều có thể sử dụng break và continue.

Ví dụ với break:

```
# breaking out of a loop early
for item in [1, 2, 3, 4, 5]:
    if item == 3:
        print(item, " ...break!")
        break
    print(item, " ...next iteration")
```

```
## 1  ...next iteration
## 2  ...next iteration
## 3  ...break!
```

Câu lệnh **continue** được viết trong khối lệnh của các vòng lặp. Khi thực hiện câu lệnh **continue**, các câu lệnh phía sau trong khối lệnh sẽ không được thực hiện mà sẽ quay lại lần lặp tiếp theo

```
for x in 1,2,3,4,5:
  if(x%2 == 0):
    continue
  print("x= ", x, "la so le")
```

```
## x=  1 la so le
## x=  3 la so le
## x=  5 la so le
```

```
for x in 1,2,3,4,5:
  if(x == 4):
    break
  print("x= ", x)
```

```
## x=  1
## x=  2
## x=  3
```

```
for x in range(3):
  for y in range(5):
    if y == 2:
```

```
        break       print("x, y:",x,y,sep=' ')
```

```
## x, y: 0 0
## x, y: 0 1
## x, y: 1 0
## x, y: 1 1
## x, y: 2 0
## x, y: 2 1
```

4.5 Bài tập áp dụng for, while

Hãy sử dụng cấu trúc lặp for và while thực hiện các bài toán sau:

Bài 1. Cho một số tự nhiên $n(n \leq 10^{1000})$. Tính tổng các chữ số của n. Ví dụ: $n = 123$, tổng các chữ số bằng $1 + 2 + 3 = 6$

Bài 2. Chữ số lớn nhất và chữ số nhỏ nhất. Cho một số tự nhiên $n(n \leq 10^{1000})$. Tìm chữ số lớn nhất và chữ số nhỏ nhất của n. Ví dụ $n = 123$, chữ số lớn nhất bằng 3, chữ số nhỏ nhất bằng 1.

Bài 3. Cho hai số nguyên dương a và b. Sử dụng thuật toán Euclid tìm ước chung lớn nhất và bội số chung nhỏ nhất của a và b.

Bài 4. Kiểm tra số nguyên tố - Cho một số tự nhiên N nhập vào từ bàn phím, hãy kiểm tra xem N có phải là một số nguyên tố hay không?

Bài 5. Số hoàn hảo - Một số tự nhiên N được gọi là một số hoàn hảo nếu tổng các ước dương của nó bằng 2N. Ví dụ: N=6 có các ước dương là 1,2,3,6. Tổng bằng 2*6=12. Như vậy 6 là một số hoàn hảo. Hãy nhập vào một đoạn [a;b]. Kiểm tra xem có bao nhiêu số hoàn hảo nằm trong khoảng này.

Bài 6. Số lũy thừa dạng 2-3: Một số tự nhiên N được gọi là có dạng lũy thừa 2-3 nếu N có dạng: $N = 2^x 3^y$. Ví dụ: 2,3,6,12... là các số lũy thừa 2-3. Nhập vào một số tự nhiên N từ bàn phím, kiểm tra xem N có phải là lũy thừa 2-3 không?

5

Kiểu dữ liệu chuỗi

5.1 Định nghĩa

Chuỗi là một dãy các ký tự liên tiếp. Mọi ký tự nằm trong dấu "" (ngoặc kép) và dấu ” (ngoặc đơn) đều được xem là chuỗi trong Python. Chuỗi rỗng được xác định là chuỗi không có ký tự nào.

Ví dụ về chuỗi như sau: “Lập trình Python!”, ‘Đây cũng là chuỗi!’

Các ký tự trong chuỗi là các ký tự thuộc bảng mã Unicode hoặc bảng mã ASCII. Chính vì vậy, chuỗi trong Python có hỗ trợ tiếng Việt. Ký tự trong chuỗi được đánh chỉ số thứ tự bằng 0,1,...theo chiều từ trái sang phải và -1,-2,...theo chiều từ phải sang trái. Số ký tự trong chuỗi được gọi là độ dài của chuỗi

String in Python

5.2 Thao tác với chuỗi

5.2.1 Lấy độ dài chuỗi

Chúng ta sử dụng phương thức *len(biến chuỗi)* để lấy được độ dài của chuỗi. Ví dụ:

```
str1 = "Learning Python is quite interesting!"
print("Chuoi str1: ",str1)
```

```
## Chuoi str1:  Learning Python is quite interesting!
```

```
print("Co do dai la: ", len(str1))
```

```
## Co do dai la:  37
```

5.2.2 Truy cập phần tử của chuỗi

Như vậy có thể thấy rằng chuỗi bản chất chính là một mảng 1 chiều các ký tự. Tuy nhiên, Python không có kiểu dữ liệu ký tự, một ký tự đơn giản chỉ là một chuỗi có độ dài bằng 1. Dấu ngoặc vuông [] có thể được sử dụng để truy cập các phần tử của chuỗi. Ký tự đầu tiên có chỉ số là 0.

```
str1 = "Hello World!"
print(str1[0])
```

```
## H
```

```
str2 = "HELLO"
print(str2[:])
```

```
## HELLO
```

```
print(str2[0:])
```

```
## HELLO
```

```
print(str2[:5])
```

```
## HELLO
```

```
print(str2[:3])
```

```
## HEL
```

```
print(str2[0:2])
```

```
## HE
```

```
print(str2[1:4])
```

```
## ELL
```

Truy cập chuỗi bằng chỉ mục âm: Sử dụng các chỉ mục âm để lấy ra chuỗi con bắt đầu từ cuối chuỗi:

```
str3 = "Hello World!"
print(str3[-5:-2])
```

```
## orl
```

5.2.3 Ghép chuỗi và lặp chuỗi

Python cho phép tạo một chuỗi mới bằng cách ghép các chuỗi lại với nhau:

```
str1 = "Tran Quang Quy"
str2 = " ICTU"
str = str1 + str2
print(str)
```

```
## Tran Quang Quy ICTU
```

Để ghép nhiều lần một chuỗi lại với nhau, ngoài cách ghép trực tiếp chuỗi như trên, Python còn cho phép thực hiện một cách ngắn gọn hơn như sau:

```
str1 = "Python Programming"
str1 = str*7
print(str1)
```

```
## Tran Quang Quy ICTUTran Quang Quy ICTUTran Quang Quy
ICTUTran Quang Quy ICTUTran Quang Quy ICTUTran Quang Quy
ICTUTran Quang Quy ICTU
```

5.2.4 So sánh chuỗi

So sánh chuỗi được thực hiện bằng cách lần lượt so sánh các ký tự trong chuỗi từ trái sang phải. Nếu gặp một ký tự không bằng nhau thì chuỗi nào chứa ký tự lớn hơn sẽ là chuỗi lớn hơn.

```
str1 = "Lap trinh Python"
str2 = "That la thu vi"
if str1 < str2:
```

```
    print("Xau nho hon la: ", str1)      else:
    print("Xau lon hon la: ", str2)
```

```
## Xau nho hon la:  Lap trinh Python
```

5.2.5 Đảo ngược thứ tự chuỗi

Để đảo ngược thứ tự của một chuỗi, ta thực hiện:

```
str = "Python is perfect!"
str1 = str[::-1]
print(str)
```

```
## Python is perfect!
```

```
print(str1)
```

```
## !tcefrep si nohtyP
```

5.2.6 Bài tập luyện tập chuỗi

Thực hiện luyện tập một số dạng bài tập về chuỗi cơ bản sau đây:

Bài 1: Chuỗi đối xứng: Một chuỗi được gọi là đối xứng nếu đọc chúng từ trái sang phải cũng giống như đọc từ phải sang trái. Ví dụ: “madam”, “2332” được coi là các chuỗi đối xứng. Hãy viết chương trình kiểm tra tính đối xứng của một chuỗi.

Bài 2: Nén chuỗi: Cho một chuỗi nào đó gồm các ký tự chữ cái La-tinh thường. Chúng ta thực hiện nén chuỗi theo quy tắc sau: Với một dãy các ký tự liên tiếp, thay thế bằng: . Ví dụ: str = “mmmbddddddccca”. Chuỗi này được nén lại là : m3bd6c3a. Yêu cầu: Hãy nhập vào một chuỗi và thực hiện nén chuỗi đó và in ra màn hình.

Bài 3: Tách số và ký tự chữ cái: Nhập vào một chuỗi gồm các ký tự số và ký tự chữ cái La-tinh nhỏ 'a', 'b', ..., 'z'. Hãy tách chuỗi được nhập thành hai chuỗi với điều kiện một chuỗi chỉ chứa các ký tự chữ cái và một chuỗi chỉ chưa các ký tự số. Ví dụ: Cho chuỗi sau: "tranquangquy1988" được tách thành hai chuỗi là "tranquangquy" và "1988".

5.3 Thao tác nâng cao với chuỗi

5.3.1 Tách chuỗi

Chúng ta sử dụng phương thức: .split() để tách chuỗi thành các chuỗi con với một khoảng cách dấu trắng.

```
str1 = "Lap trinh Python"
str2 = str1.split()
print("Chuoi str1 duoc tach thanh la: ", str2)
```

```
## Chuoi str1 duoc tach thanh la:  ['Lap', 'trinh',
'Python']
```

5.3.1.1 Tách chuỗi có sử dụng tham số

Ký tự phân cách mặc định trong Python là ký tự trăng, tuy nhiên chúng ta có thể thay đổi ký tự trắng này thành một dạng khác. Ví dụ:

```
str1 = "Hoc lap trinh Python, that thu vi"
str2 = str1.split(",")
print("Chuoi str1 duoc tach ra la: ", str2)
```

```
## Chuoi str1 duoc tach ra la:  ['Hoc lap trinh Python', '
that thu vi']
```

5.3.1.2 Tham số về số chuỗi tối đa được tách

Chúng ta có thể tách được một chuỗi tối đa theo ký tự và số chuỗi con tối đa được tách như sau:

```
str1 = "Tu#hoc#lap#trinh#Py#thon"
str2 =  str1.split("#", 2)
print("Chuoi duoc tach toi da ra la: ", str2)
```

```
## Chuoi duoc tach toi da ra la:  ['Tu', 'hoc',
'lap#trinh#Py#thon']
```

Chú ý: Tham số mặc định là -1, nghĩa là chuỗi được tách tất cả

```
str1 = "Tu#hoc#lap#trinh#Python"
str2 = str1.split("#", -1)
print(str2)
```

```
## ['Tu', 'hoc', 'lap', 'trinh', 'Python']
```

5.3.1.3 Một số thao tác khác với chuỗi

Chuyển đổi ký tự chữ hoa, chữ thường: Để chuyển đổi chuỗi từ chữ thường thành chữ hoa, chúng ta sử dụng phương thức upper(), chuyển đổi thành chữ thường sử dụng phương thức lower():

```
str11 = 'tran quang quy'
str22 = str11.upper()
print(str22)
```

```
## TRAN QUANG QUY
```

```
str3 = 'TRAN QUANG QUY'
str4 = str3.lower()
print(str4)
```

```
## tran quang quy
```

Phương thức capitalize() sẽ chuyển ký tự đầu tiên của chuỗi thành chữ hoa, còn phương thức title() sẽ chuyển đổi các ký tự đầu tiên của các phần tử trong chuỗi thành chữ hoa

```
st = 'tu hoc lap trinh python'
st1 = st.capitalize()
print(st1)
```

```
## Tu hoc lap trinh python
```

```
st2 = st.title()
print(st2)
```

```
## Tu Hoc Lap Trinh Python
```

Phương thức swapcase() trả về chuỗi sau khi chuyển các ký tự hoa thành ký tự thường, ký tự thường thành ký tự chữ hoa

```
st11 = 'Tu Hoc Lap Trinh Python'
st22 = st11.swapcase()
print(st22)
```

```
## tU hOC lAP tRINH pYTHON
```

Chú ý: Sau khi thực hiện các phương thức upper(); lower(); capitalize(); title(); swapcase() thì chuỗi ban đầu không thay đổi.

5.3.1.4 Phép toán in

Phép toán : *str1* **in** *st* trả về True nếu str1 là chuỗi con của chuỗi st; ngược lại phép toán sẽ trả về giá trị là False

```
str11 = "quy"
str22 = "tranquangquy"
if str11 in str22 : print("La mot chuoi con")
else: print("Khong phai la chuoi con")
```

```
## La mot chuoi con
```

5.3.2 Một số phương thức nâng cao

5.3.2.1 Phương thức count()

Phương thức count() trả về số lần xuất hiện của chuỗi str1 trong (biến chuỗi). Ngoài ra trong phương thức này có thể truyền thêm tham số: (biến chuỗi).count(str1,i) trả về số lần xuất hiện của chuỗi str1 trong (biến chuỗi) tính từ chỉ số thứ i. Nếu thêm tham số j như sau: (biến chuỗi).count(str1,i,j) sẽ trả về số lần xuất hiện của chuỗi str1 trong (biến chuỗi) tính từ chỉ số thứ i đến chỉ số j-1.

```
str1 = 'ab'
st = 'ab12abab'
n = st.count(str1)
m = st.count(str1, 1)
k = st.count(str1, 1, 6)
print(n)
```

```
## 3
```

```
print(m)
```

```
## 2
```

```
print(k)
```

```
## 1
```

5.3.2.2 Phương thức tìm kiếm

Phương thức (biến chuỗi).find(str1) - trả về vị trí chỉ số đầu tiên mà chuỗi str1 xuất hiện trong (biến chuỗi). Nếu str1 không xuất hiện trong (biến chuỗi) thì kết quả trả về là -1.

```
str1 = 'ab'
str2 = 'abc'
st = 'acabab'
print('Vi tri dau tien str1 trong st la: ', st.find(str1))
```

```
## Vi tri dau tien str1 trong st la:  2
```

```
print('Vi tri dau tien str2 trong st la: ', st.find(str2))
```

```
## Vi tri dau tien str2 trong st la:  -1
```

Phương thức (biến chuỗi).find(str1, i) trả về vị trí chỉ số đầu tiên mà chuỗi str1 xuất hiện trong (biến chuỗi) tính từ chỉ số thứ i.

Phương thức (biến chuỗi).find(str1, i, j) trả về vị trí chỉ số đầu tiên mà chuỗi str1 xuất hiện trong (biến chuỗi) tính từ chỉ số thứ i đến chỉ số j-1.

```
str1 = 'ab'
st = 'abababb'
print('Vi tri dau tien: ', st.find(str1, 1))
```

```
## Vi tri dau tien:  2
```

```
print('Vi tri dau tien: ', st.find(str1, 1, 3))
```

```
## Vi tri dau tien:  -1
```

Chú ý: Phương thức **st.index(str1)** cũng trả về vị trí chỉ số đầu tiên của chuỗi **str1** xuất hiện trong **st**, nhưng khi **str1** không xuất hiện trong **st** thì chương trình sẽ báo lỗi.

5.3.2.3 Phương thức thay thế

Phương thức **(biến chuỗi).replace**(str_old, str_new) sẽ trả về chuỗi được tạo thành bằng cách thay thế tất cả các chuỗi con str_old bằng chuỗi str_new trong (biến chuỗi). Ngoài ra nếu có tham số như sau: .replace(str_old, str_new, num_max) thì số lần thay thế nhiều nhất là num_max

```
st = 'aaabaa'
str1 = st.replace('aa', "BB")
print(st)
```

```
## aaabaa
```

```
print(str1)
```

```
## BBabBB
```

5.3.2.4 Xóa ký tự trắng bên phải, trái của chuỗi

Sử dụng phương thức **(biến chuỗi).lstrip()** hoặc **(biến chuỗi).rstrip()** sẽ xóa tất cả các ký tự trắng ở bên trái hoặc bên phải của chuỗi

```
st11 = ' tran quang quy'
st22 = st11.lstrip()
print(st22)
```

```
## tran quang quy
```

Nếu chỉ là **(biến chuỗi).strip()** thì sẽ xóa tất cả các khoảng trắng bên phải và bên trái của chuỗi. Nếu truyền thêm tham số: **(biến chuỗi).lstrip(Các ký tự)** hoặc **(biến chuỗi).rstrip(Các ký tự)** hoặc **(biến chuỗi).strip(Các ký tự)** thì chuỗi được trả về sẽ xóa tất cả các ký tự trong tập hợp *Các ký tự*

```
st33 = 'aaACCbab'
str331 = st33.lstrip('aAb')
str332 = st33.rstrip('aAb')
str333 = st33.strip('aAb')
print(str331)
```

```
## CCbab
```

```
print(str332)
```

```
## aaACC
```

```
print(str333)
```

```
## CC
```