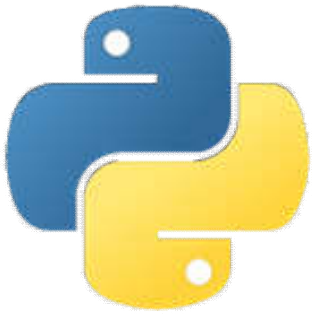


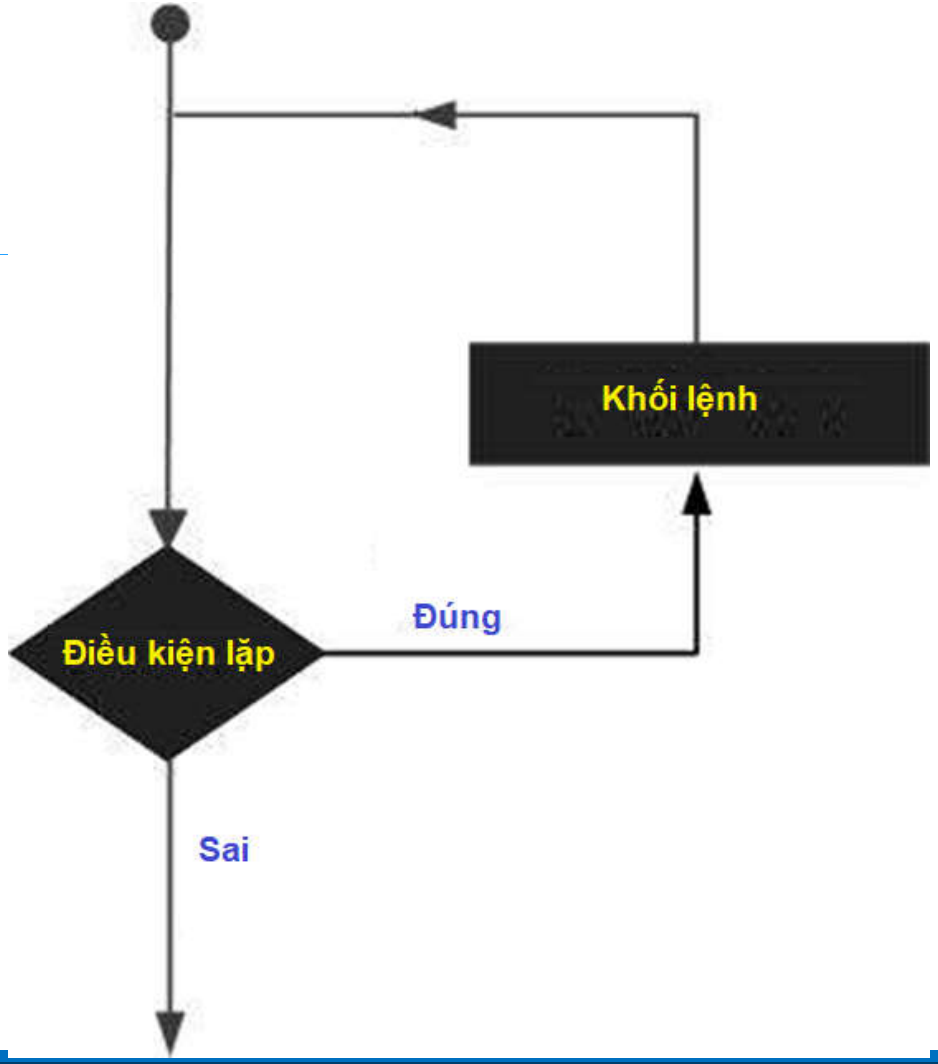
Các cấu lặp & string

Nội dung

1. **Cấu trúc lặp**
2. **Chuỗi & các phương thức xử lý chuỗi**
3. **Q & A**

1.1. Cấu trúc lặp

- Trong Python các câu lệnh (statement) được thực hiện một cách tuần tự từ trên xuống dưới. Tuy nhiên trong một vài trường hợp, lập trình viến muốn thực hiện một khối các câu lệnh nhiều lần, khi đó có thể sử dụng vòng lặp (loop).

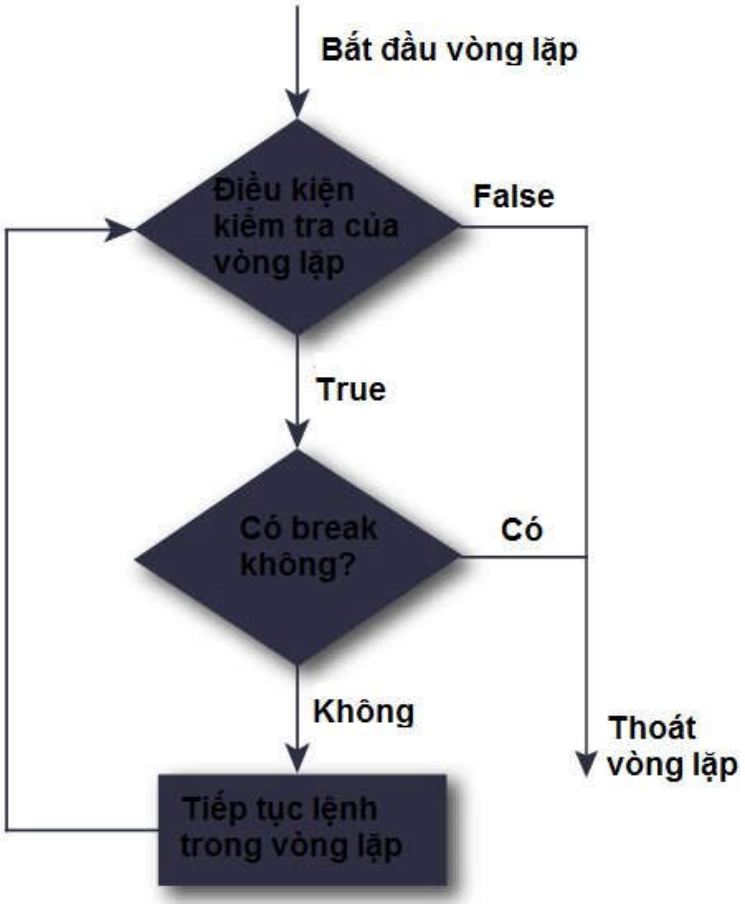


1.2. Các cấu trúc lặp

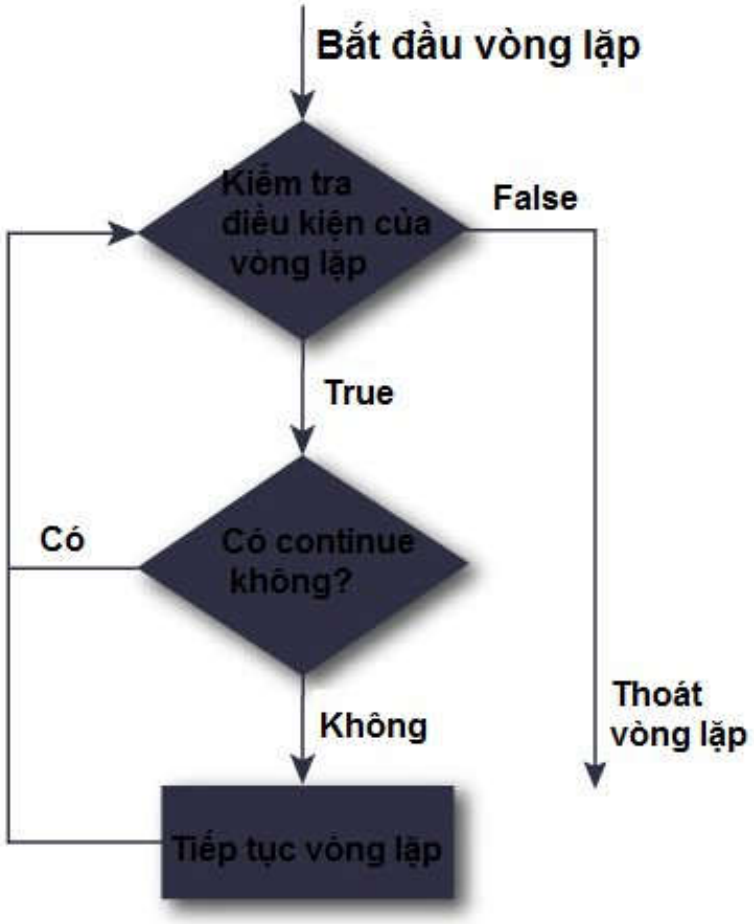
- Python cung cấp cho 2 loại vòng lặp sau:
  - Vòng lặp for
  - Vòng lặp while
- Các câu lệnh (statement) có thể được sử dụng bên trong vòng lặp:
  - continue
  - break
  - pass
- Các câu lệnh liên hợp với vòng lặp:
  - else

Lệnh	Mô tả
break	Đây là câu lệnh ngừng vòng lặp.
continue	Đây là câu lệnh để bỏ qua các câu lệnh còn lại trong khối lệnh (block), và ngay lập tức kiểm tra lại điều kiện trước khi tiếp tục thực thi lại khối lệnh.
pass	Lệnh **pass** trong vòng lặp chỉ đơn giản là một đánh dấu, nhắc lập trình viên nhớ thêm đoạn mã (code) nào đó trong tương lai. Nó là một lệnh ***null*** (Không làm gì cả).

❖Break



❖Continue



1.3. Vòng lặp for với hàm range()

- Đơn giản nhất của vòng lặp for trong Python là sử dụng 'for' với 'range‘.
- Nếu biến ‘_i' có giá trị chạy trong pham vi (1, 6) có nghĩa rằng x có thể nhận các giá trị 1, 2, 3, 4, 5. Cú pháp sẽ được viết như sau:

```
for _i in range (1, 6) :
```

❖**Ví dụ 1**: In ra các số tự nhiên liên tiếp từ 1 đến 9.

❖**Thuật toán**: Sử dụng vòng lặp với số tự nhiên _i chạy từ 1 đến 10. Trong mỗi vòng lặp thì in ra giá trị của _i.

❖**Code mẫu**:

```
for _i in range (1,10):
    print (_i)
```

❖**Ví dụ 2**: In ra các bình phương của số tự nhiên liên tiếp từ 1 đến 5.

❖**Thuật toán**: Sử dụng vòng lặp với số tự nhiên _i chạy từ 1 đến 5. Trong mỗi vòng lặp thì in ra giá trị của _i*_i (tức là _i bình phương).

❖**Code mẫu**:

```
for _i in range (1,6):
    print (_i*_i)
```

```
for _i in range(1, 6):
    _a = _i * _i
    print(_a)
```

- Cú pháp tổng quan của range(): range([start], stop, [step])
- Trong đó:
  - start: Giá trị bắt đầu. Giá trị mặc định của start là 0. Giá trị start là tuỳ ý và nếu như không được sử dụng trong hàm thì giá trị mặc định của nó sẽ là 0.
  - stop: Giá trị dừng. Phần tử cuối cùng trong vòng lặp luôn nhỏ hơn stop.
  - step: Khoảng cách giữa hai phần tử liền kề trong danh sách lặp. Giá trị step là tuỳ ý và nếu không được sử dụng trong hàm thì giá trị mặc định của nó sẽ là 1.

```
#In ra so tu 1 -> 9
for _i in range(10):
    print(_i)
#In ra cac so tu 1 -> 9, moi so cach nhau 2 don vi.
for _i in range(1,10,2):
    print(_i)
```

1.4. Lồng các lệnh lặp for

❖Có thể lồng trong các lệnh lặp là các lệnh lặp khác.

```
for _i in range(1, 6):
    for _j in range (3,5):
        _a = _i * _j
        print(_a)
```

❖Có thể lồng lệnh lặp trong các lệnh điều khiển (if/elif/else) hoặc ngược lại

```
--- Code 1: If trong for ---
for _i in range(1, 6):
    if _i > 3:
        _a = _i * _i
        print('Binh phuong cua %r:
%r' % (_i, _a))
    else:
        _a = _i
        print(_a)
```

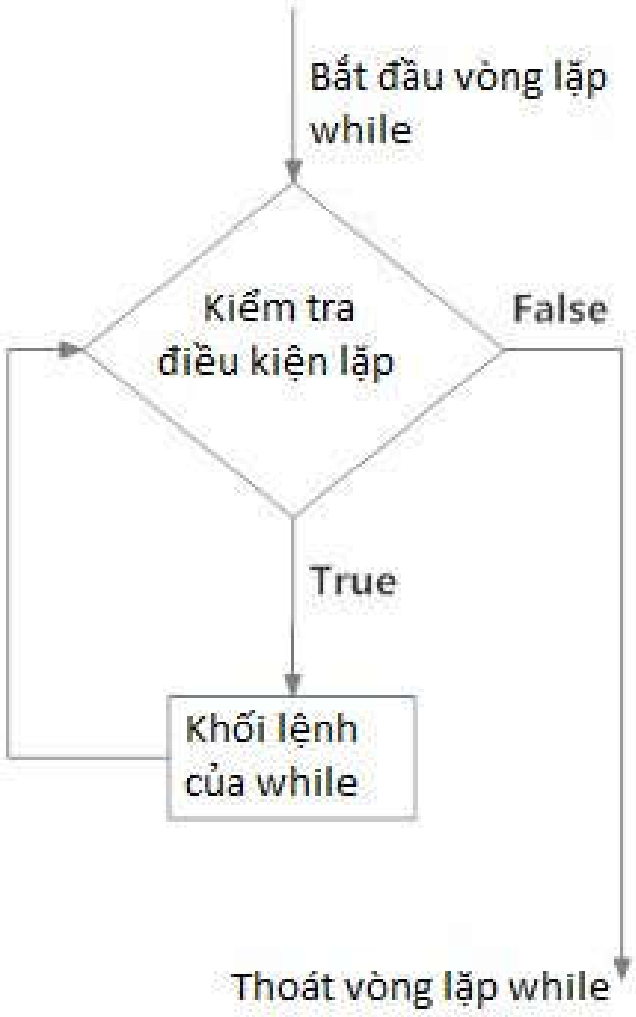
```
--- Code 2: For trong if ---
_n = int(input('Nhap vao so tu nhien: '))
if _n <= 5:
    for _i in range (1,_n):
        print(_i)
else:
    print('So _n lon hon 5.')
```

Luyện tập for

- **Bài 1**: Nhập vào số nguyên n từ bàn phím. In ra tích của 2 nhân với các giá trị nhỏ hơn n. Ví dụ: Nhập n = 4 thì in ra 2 = 2*1, 4 =2*2, 6 = 2*3.
- **Bài 2**: Nhập vào 1 số nguyên n từ bàn phím.
  - Nếu số đó lớn hơn 10 thì in ra dòng text: *Số nhập vào phải bé hơn 10*.
  - Nếu số đó nhỏ hơn hoặc bằng 10: In ra những số chẵn trong khoảng từ 1 đến n.
- **Bài 3**: In ra các số trong khoảng từ 80 - 100 thỏa mãn điều kiện vừa chia hết cho 2, vừa chia hết cho 3.
- **Bài 4**: Nhập vào 1 số nguyên n < 20 từ bàn phím. In ra các số thoản mãn điều kiện chia hết cho 5 hoặc chia hết cho 7.

1.5. Vòng lặp với While

```
while điều_kiện_kiểm_tra:
        Khối lệnh của while
```



```
# Vi du: In ra cac so nguyen duong nho hon 9.
count = 0
n = 0
while (count < 9):
      print ('Số thứ', n,' là:', count)
      n = n + 1
      count = count + 1
print ("Finish!")
```

```
# Vi du: Nhap vao tu ban phim so nguyen n. Tinh tong cac so tu 1 den n.
n = int(input("Nhap so 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("SUM: ", tong)
```

1.5.1. Lặp while vô hạn

❖Lấy lại ví dụ trên, chỉ cần bỏ đi dòng i=i+1 vòng lặp sẽ trở thành vô hạn

```
n = int(input("Nhap so 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

print("SUM: ", tong)
```

1.5.2. While kết hợp else

❖Kết hợp while và else:

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

Luyện tập while

- **Bài 1**: Tính và in ra tích của 10 số tự nhiên đầu tiên.
- **Bài 2**: Nhập vào số nguyên dương n từ bàn phím. Tính và in ra n! (giai thừa của n).
- **Bài 3**: Nhập vào số nguyên dương n từ bàn phím. Kiểm tra xem số n có phải là số nguyên tố hay không. Nếu là số nguyên tố thì in ra dòng text: Đây là số nguyên tố. Nếu không thì in ra: Không phải số nguyên tố.
- **Bài 4**: Nhập vào từ bàn phím số nguyên n. In ra tổng của các số thỏa mãn hai điều kiện: nhỏ hơn n và là số chẵn.

1. Cấu trúc lặp
2. Chuỗi & các phương thức xử lý chuỗi
3. Q & A

2.1. String

- Chuỗi (string) là một kiểu (type) thông dụng nhất trong Python.
- Chú ý rằng trong Python không có kiểu ký tự (character), ký tự đơn giản được coi là một string có độ dài 1.
- Cú pháp khai báo:

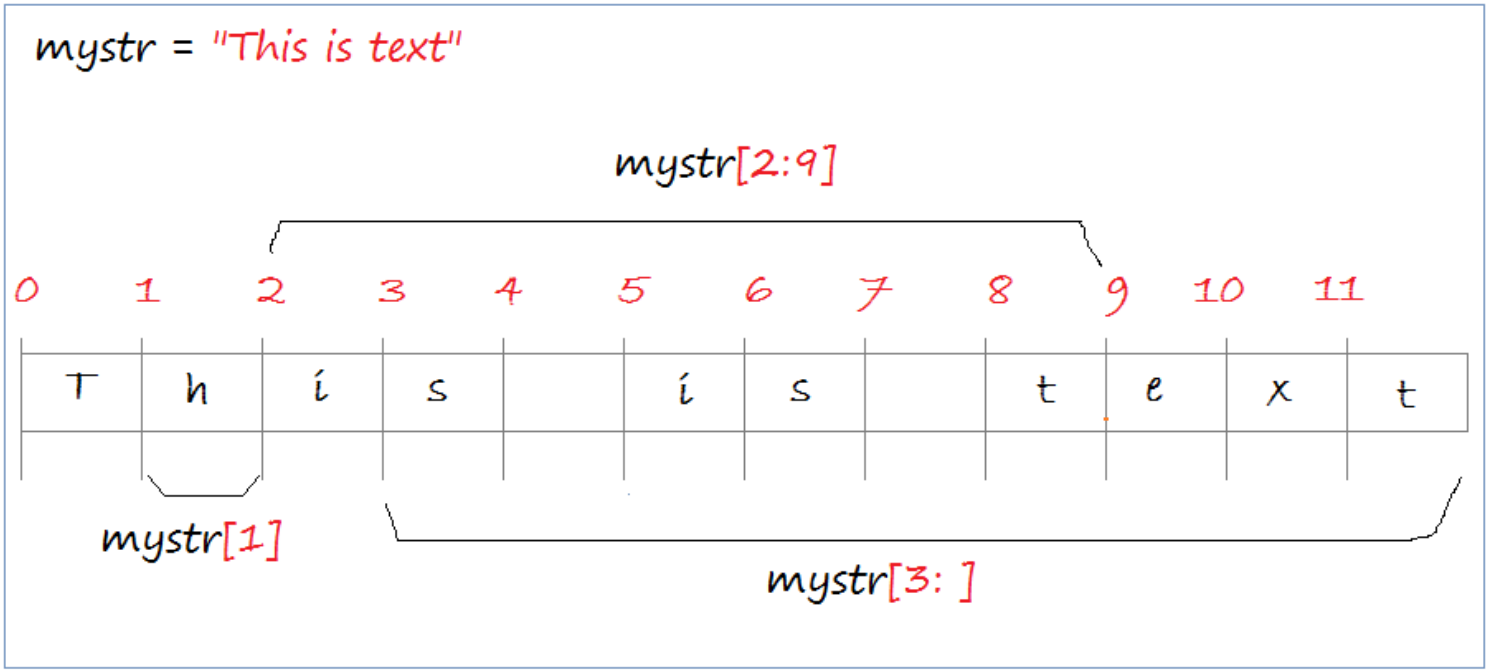
```
str1 = "Hello Python"
str2 = 'Hello Python'

str3 = "I'm from Vietnam"

str4 = 'This is a "Cat"! '
```

2.2. Substring

- Do không hỗ trợ kiểu ký tự (Character type), ký tự được coi là một string với độ dài 1.
- Các ký tự trong string được đánh chỉ số bắt đầu từ 0. Vì vậy, có thể truy cập vào các chuỗi con (substring) thông qua chỉ số này.



[]	Trả về ký tự tại vị trí cho bởi chỉ số.	a = "Hello" a[1] ==> "e"
[:]	Trả về một chuỗi con chứa các ký tự cho bởi phạm vi (range)	a = "Hello" a[1:4] ==> "ell" a[1:] ==> "ello"

```
# khai báo 1 string
mystr = "This is text"


# --> h
print ("mystr[1] = ", mystr[1])

# --> is is t
print ("mystr[2,9] = ", mystr[2:9])


# --> s is text
print ("mystr[3:] = ", mystr[3:])
```

2.3. Toán tử thao tác với string

- Toán tử cơ bản: + (toán tử ghép chuỗi) và * (toán tử lặp chuỗi)
- Toán tử membership:
  - in: trả về true nếu một ký tự là có mặt trong chuỗi đã cho, nếu không nó trả về false.
  - not in: trả về true nếu một ký tự là không tồn tại trong chuỗi đã cho, nếu không nó trả về false.
- Toán tử quan hệ: <; >; <=; >=; ==; !=

```
# Membership trong string
str1 = "javapoint"
str2 = 'sssit'
str4 = 'java'
str5 = "it"

_a = str4 in str1
print('str4 in str1', _a)

_b = str5 in str2
print('str5 in str2', _b)

_c = str4 not in str1
print('str4 not in str1',_c)
```

2.4. Hàm xử lý chuỗi

- Python cung cấp các phương thức đa dạng đã được xây dựng sẵn để thao tác với các chuỗi:

1. Cấu trúc lặp
2. Chuỗi & các phương thức xử lý chuỗi
3. Q & A


Q
&
A