LẬP TRÌNH PYTHON TỪ CƠ BẢN ĐẾN NÂNG CAO

Thực hiện: TS.Trần Bình Long

Điều khiển luồng và vòng lặp

1. Lệnh if, if...else, if...elif...else
2. Vòng lặp for
3. Vòng lặp while
4. Lệnh break và continue
5. Lệnh pass
6. Các kỹ thuật vòng lặp

Cấu trúc lệnh if

```
if điều kiện:
    khối lệnh
```

Chương trình kiểm tra *điều kiện* và thực hiện các *lệnh* khi *điều kiện* là True. Nếu *điều kiện* False thì *lệnh* không thực hiện.

Khối lệnh của ***if*** bắt đầu với thụt lề và kết thúc lệnh ***if*** với dòng không thụt lề.

Cấu trúc lệnh if

```
>>> # Nếu là số dương sẽ in thông báo
    num = 3
    if num > 0:
        print(num, "là số dương.")
    print("Thông báo này luôn được in.")
    num = -1
    if num > 0:
        print(num, "là số dương.")
    print("Thông báo này cũng được in.")
```

Kết quả

```
    3 là số dương.
    Thông báo này luôn được in.
    Thông báo này cũng được in.
```

Lệnh if...else

Cấu trúc lệnh if...else

```
if điều kiện:
    Khối lệnh của if
else:
    Khối lệnh của else
```

Lệnh ***if*** kiểm tra điều kiện và thực hiện khối lệnh ***if*** nếu điều kiện đúng. Nếu điều kiện sai, khối lệnh của ***else*** sẽ được thực hiện.

Thụt lề được sử dụng trong các khối lệnh.

Lệnh if...else

```
>>> # Kiem tra so am hay duong
>>> num = 3
    if num >= 0:
        print("So duong hoac bang 0")
    else:
        print("So am")
    # Output: So duong hoac bang 0
    num1=-1
    if num1 >= 0:
        print("So duong hoac bang 0")
    else:
        print("So am")
    # Output: So am
```

Lệnh if...elif...else

Cấu trúc lệnh if...elif...else

```
if điều kiện:
    Khối lệnh của if
elif điều kiện:
    Khối lệnh của elif
else:
    Khối lệnh của else
```

elif là viết gọn của ***else if***, cho phép kiểm tra nhiều điều kiện. Nếu điều kiện sai, sẽ kiểm tra điều kiện của khối elif tiếp theo và tiếp tục cho đến hết. Nếu tất cả điều kiện đều sai sẽ thực hiện khối lệnh của ***else***. Có thể có nhiều ***elif***, phần ***else*** là tùy chọn

Lệnh if...elif...else

Ví dụ:

```
>>> x = int(input("Nhap mot so: "))
    if x < 0:
        print('So am')
    elif x == 0:
        print('So 0')
    elif x == 1:
        print('So 1')
    else:
        print('So duong')
```

Lệnh ***if*** ... ***elif*** ... ***elif***… thay thế cho lệnh ***switch*** hay ***case*** trong ngôn ngữ lập trình khác.

Lệnh if lồng nhau

Lệnh ***if***... ***else*** trong khối lệnh ***if***... ***else*** khác tạo thành lệnh if lồng nhau.

Không giới hạn số lệnh được lồng vào nhau.

Thụt lề là cách để nhận diện mức độ lồng nhau.

```
>>> 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 for

For để duyệt các phần tử trong Python như list, string hoặc các đối tượng.

Cú pháp của for

```
for bien_lap in chuoi_lap:
    Khối lệnh của for
```

bien_lap sẽ nhận giá trị của từng mục trong **chuoi_lap** ở mỗi lần lặp.

Vòng ***for*** sẽ thực hiện đến mục cuối cùng trong chuỗi.

Khối lệnh của for được thụt lề.

Vòng lặp for

```
>>> # In chữ cái trong lachong
    for chu in ‘lachong':
    print('Chữ cái hiện tại:', chu)
```

Kết quả:

```
Chữ cái hiện tại: l
Chữ cái hiện tại: a
Chữ cái hiện tại: c
Chữ cái hiện tại: h
Chữ cái hiện tại: o
Chữ cái hiện tại: n
Chữ cái hiện tại: g
```

Vòng lặp for

```
>>> # Tính tổng tất cả các số trong danh sách A
    # Danh sách A
    A = [1, 3, 5, 9, 11, 2, 6, 8, 10]
    # Biến lưu trữ tổng các số là tong
    tong = 0
    # Vòng lặp for
    for i in A:
        tong = tong+i
    print("Tổng các số là", tong)
    # Output: Tổng các số là 55
```

Hàm range()

Dùng **hàm range()** để tạo dãy số. **Cú pháp**

Hàm range(số bắt đầu, số kết thúc, khoảng cách giữa hai số)

Nếu không đặt khoảng cách giữa hai số thì mặc định bằng 1.

Ví dụ, range(100) tạo dãy số từ 0 đến 99 (100 số).

Ví dụ: Viết 10 lần 'Lập trình Python'

```
>>> for i in range(10):
        print ('Lập trình Python')
```

Hàm range()

Dùng **list()** để hàm **range()** xuất tất cả các giá trị

```
>>> print(range(9))
    # Output: range(0, 9)
    print(list(range(9)))
    # Output: [0, 1, 2, 3, 4, 5, 6, 7, 8]
    print(list(range(2, 5)))
    # Output: [2, 3, 4]
    print(list(range(0, 15, 5)))
    # Output: [0, 5, 10]
```

Hàm range()

Hàm **range()** sử dụng kết hợp với **len()** để lặp qua một dãy sử dụng index:

```
>>> chuoi = [‘học’,’học nữa’,’học mãi’]
    for i in range(len(chuoi)):
        print(‘Chúng ta phải: ‘,chuoi[i])
```

Kết quả:

```
Chúng ta phải:  học
Chúng ta phải:  học nữa
Chúng ta phải:  học mãi
```

Kết hợp for với else

Else không chỉ kết hợp với ***if*** mà còn được dùng trong vòng lặp ***for***. Trong ***for***, khối lệnh của ***else*** được thực thi khi các mục trong chuỗi được lặp hết.

```
>>> B = [0, 2, 4]
    for b in B:
        print(b)
    else:
        print(‘Đã hết số.’)
    # Kết quả:
    0
    2
    4
    Đã hết số.
```

Lệnh break

Lệnh ***break*** được dùng để dừng vòng lặp ***for***. Nếu có ***else***, ***else*** trong ***for*** chỉ thực hiện khi không có ***break***.

```
>>> # Dãy từ 0 đến 10
    for num in range(0,10):
    # Các thừa số của một số trong dãy
       for i in range(2, num):
    #Xác định thừa số (phép chia có số dư bằng 0)
          if num%i == 0:
             j=num/i #Ước lượng thừa số thứ 2
             print ('%d bằng %d * %d' % (num,i,j))
             break #Dừng for trong, chuyển ra for ngoài
       else: # Phần else trong vòng lặp
          print (num, 'là số nguyên tố')
```

Lệnh break

Kết quả:

```
0 là số nguyên tố
1 là số nguyên tố
2 là số nguyên tố
3 là số nguyên tố
4 bằng 2 * 2
5 là số nguyên tố
6 bằng 2 * 3
7 là số nguyên tố
8 bằng 2 * 4
9 bằng 3 * 3
```

Vòng lặp while

While được dùng để lặp lại một khối lệnh, thực hiện khi điều kiện kiểm tra đúng.
while dùng trong trường hợp không biết trước được số lần lặp là bao nhiêu.
Cú pháp của while:

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

Trong Python mọi giá trị khác 0 đều là True, None và 0 được hiểu là False.

While sẽ không thực hiện nếu ngay lần đầu tiên **điều_kiện_kiểm_tra** là False.

Cú pháp của while

Khối lệnh của ***While*** bắt đầu với thụt lề và kết thúc với dòng không thụt lề.

```
>>> count = 0
    n = 0
    while (count < 8):
        print ('Số thứ', n,' là:', count)
        n = n + 1
        count = count + 1
    print ("Hết rồi!")
```

Cú pháp của while

Kết quả:

```
Số thứ 0 là: 0
Số thứ 1 là: 1
Số thứ 2 là: 2
Số thứ 3 là: 3
Số thứ 4 là: 4
Số thứ 5 là: 5
Số thứ 6 là: 6
Số thứ 7 là: 7
Hết rồi!
```

Cú pháp của while

Ví dụ

```
>>> 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 biên đếm i
    while i <= n:
        tong = tong + i
        i = i+1 # cập nhật biến đếm
    print("Tổng là", tong)
```

Kết quả:

```
>>>Nhập n: 11
   Tổng là 66
```

Vòng lặp vô tận

```
>>> 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 bỏ đi dòng i=i+1
    print("Tổng là", tong)
```

Khi này chạy lệnh ta sẽ được:

```
>>> Nhập n: 1
```

Không có lệnh nào được thực hiện. Đây là trường hợp lặp vô tận. Nhấn phím Ctrl + C để thoát khỏi vòng lặp.

While kết hợp với else

Khối lệnh của ***else*** được thực hiện khi điều kiện của ***while*** là False.

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

Kết quả:

```
>>> Đang ở vòng lặp while
    Đang ở vòng lặp while
    Đang ở vòng lặp while
    Đang ở else
```

While kết hợp với else

```
>>> n = 0
    while n < 2:
        print(n,"nhỏ hơn 2")
        n = n + 1
    else:
        print (n,"không nhỏ hơn 2")
```

Kết quả là:

```
>>> 0 là nhỏ hơn 2
    1 là nhỏ hơn 2
    2 không nhỏ hơn 2
```

while trên một dòng

Vòng lặp ***while*** chỉ có một lệnh duy nhất, có thể viết trên cùng một dòng với ***while***.

Vòng lặp vô tận với while một dòng lệnh

```
>>> flag = 1
    while (flag): print ('Flag đã cho là True!')
    Print ('Kết thúc')
```

Đây là vòng lặp vô tận, nhấn tổ hợp phím Ctrl + C để thoát vòng lặp.

Lệnh break và lệnh continue

Lệnh ***break*** và ***continue*** dùng thay đổi các bước thực hiện của vòng lặp.
Vòng lặp thực hiện khối lệnh đến khi điều kiện kiểm tra là False thì dừng, nhưng nếu muốn dừng vòng lặp mà không cần điều kiện kiểm tra, sử dụng lệnh ***break*** và ***continue***.

Lệnh break

Lệnh ***break*** kết thúc vòng lặp chứa nó và chuyển đến lệnh tiếp theo sau vòng lặp.

Nếu lệnh ***break*** ở trong vòng lặp lồng nhau, ***break*** sẽ kết thúc vòng lặp trong chuyển đến vòng lặp ngoài.

Cú pháp của lệnh break:

```
>>> break
```

Với vòng lặp for

```
>>> for var in sequence:
        # khối lệnh bên trong vòng lặp for
        if dieu_kien:
            break
        #lệnh khác bên trong vòng lặp for
    # lệnh bên ngoài vòng lặp for
```

Lệnh break

Với vòng lặp while

```
>>> while dieu_kien_kiem_tra:
        #Lệnh bên trong vòng lặp while
        if dieu_kien:
            break
        #Lệnh khác bên trong vòng lặp while
    #lệnh bên ngoài vòng lặp while
```

Khi ***break*** được thực thi '#lệnh khác bên trong vòng lặp ***for/while***' sẽ bỏ qua và chuyển đến '#lệnh bên ngoài vòng lặp ***for/while***'.

Lệnh break

```
>>> # Lệnh break trong for
    for val in "string":
        if val == "i":
            break
        print(val)
    print("Kết thúc!")
```

Vòng ***for*** thực hiện trong chuỗi "string", và kiểm tra điều kiện, nếu gặp chữ 'i', lệnh ***break*** được thực hiện. Kết quả các chữ khác 'i' trước đó được in ra màn hình.

```
>>> s
    t
    r
    Kết thúc!
```

Lệnh break

```
>>> bien = 10
    while bien > 0:
        print ('Giá trị biến hiện tại là: ', bien)
        bien = bien -1
        if bien == 5:
            break
    print ("OK!")
```

Lệnh trên kiểm tra và in biến theo giá trị giảm dần từ 10, cho đến khi biến bằng 5 thì kết thúc vòng lặp.

```
>>> Giá trị biến hiện tại là: 10
    Giá trị biến hiện tại là: 9
    Giá trị biến hiện tại là: 8
    Giá trị biến hiện tại là: 7
    Giá trị biến hiện tại là: 6
    OK!
```

Lệnh continue

Lệnh ***continue*** bỏ qua phần lệnh còn lại bên trong vòng lặp và thực hiện cho lần lặp hiện tại.

Cấu trúc của continue:

```
>>> continue
```

Với vòng lặp for

```
>>> for var in sequence:
        #khối lệnh bên trong vòng lặp for
        if dieu_kien:
            continue
        #Lệnh khác bên trong vòng lặp for
    #Lệnh bên ngoài vòng lặp for
```

Lệnh continue

Với vòng lặp while

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

Khi ***continue*** được thực thi '#Lệnh khác bên trong vòng lặp ***for/while***' sẽ bị bỏ qua và quay trở lại '#Lệnh bên trong vòng lặp ***for/while***'

Lệnh continnue

```
>>> # Lệnh continue trong for
    for val in "string":
        if val == "i":
            continue
        print(val)
    print("Kết thúc!")
```

Khi ***for*** lặp chuỗi "string" đến chữ "i", sẽ bỏ qua lệnh in print(val) và quay trở lại lệnh ***if*** val=="i":, kết quả:

```
>>> s
    t
    r
    n
    g
    Kết thúc!
```

Lệnh continnue

```
>>> bien = 10
    while bien > 0:
        bien = bien -1
        if bien == 5:
            continue
        print ('Giá trị biên hiện tại là: ', bien)
    print ("OK!")
```

Nếu bien = 5 bỏ qua và thực hiện lần lặp tiếp theo, kết quả:

```
>>> Giá trị biến hiện tại là: 9
    . . . .
    Giá trị biến hiện tại là: 6
    Giá trị biến hiện tại là: 4
    Giá trị biến hiện tại là: 3
    . . ..
    Giá trị biến hiện tại là: 0
    OK!
```

Lệnh pass

Lệnh ***pass*** như một placeholder (trình giữ chỗ) cho việc thực thi các hàm, vòng lặp,... trong tương lai.

Khi lập trình, dự định có hàm, vòng lặp, nhưng chưa thể xây dựng, muốn để lại làm sau. Nhưng hàm, vòng lặp đó không thể để rỗng, trình biên dịch sẽ báo lỗi,

Sử dụng lệnh ***pass*** để xây dựng một khối lệnh rỗng, lúc này trình biên dịch sẽ không báo lỗi.

Trong lệnh ***pass*** các chú thích, bình luận sẽ không bị bỏ qua.

Cấu trúc của lệnh pass

```
>>> pass
```

Lệnh pass

```
>>> # pass giữ chỗ cho for lệnh sẽ thêm vào sau.
    sequence = {'p', 'a', 's', 's'}
    for val in sequence:
        pass
>>> # Pass giữ chỗ cho hàm
    def function(args):
    pass
>>> # pass giữ chỗ cho lớp
>>> class example:
    pass
```

Pass tạo nên một khối lệnh rỗng cho ***for/hàm/lớp***. Khi thực hiện chương trình sẽ không báo lỗi.

Vòng lặp vô tận

Trong Python có 2 vòng lặp, ***for*** và ***while***. Khi sử dụng những vòng lặp này với câu lệnh ***break*** và ***continue*** có thể tạo ra những dạng vòng lặp khác nhau.

Tạo vòng lặp vô tận bằng while.

Khi điều kiện của vòng lặp ***while*** luôn True sẽ được vòng lặp vô tận.
Thoát khỏi vòng lặp vô tận dùng tổ hợp phím ***Ctrl + C***

Vòng lặp vô tận sử dụng while

```
>>> # nhập vào một số và in ra số gấp 3 lần số đó.
    # Vòng lặp là vô tận vì không có điều kiện dừng.
    while True:
        num = int(input('Nhập một số: '))
        print('Gấp ba của ',num,' là ',3 * num)
```

Kết quả:

```
>>> Nhập một số: 3
    Gấp ba của 3 là 9
```

Enter sẽ tiếp tục nhập số mới và chương trình sẽ chạy mãi cho đến khi nhấn Ctrl + C

Vòng lặp while với điều kiện ở đầu

Điều kiện của vòng lặp ***while*** xuất hiện ở đầu và vòng lặp kết thúc khi điều kiện False .

```
>>> n = 15 # gán n = 15
    # n = int(input("Nhập số n: ")), Người dùng tự nhập
    # Khởi tạo tổng và biến đếm i
    tong = 0
    i = 1
    while i <= n:
        tong = tong + i
        i = i+1 # cập nhật biến đếm
    print("Tổng các số từ 1 đến ",n, " là", tong)
```

Kết quả:

```
>>>> Tổng các số từ 1 đến 15 là 120
```

Vòng lặp while với điều kiện ở giữa

Vòng lặp vô tận với lệnh ***break*** trong khối lệnh của vòng lặp

```
>>> # Nhập ký tự cho đến khi được một nguyên âm
    nguyenAm = "aeiouAEIOU"
    # vòng lặp vô hạn
    while True:
        m = input("Nhập một nguyên âm: ")
        # Điều kiện ở giữa khối lệnh
        if m in nguyenAm:
            break
        print("Đây không phải là nguyên âm. Thử lại!")
    print("Đúng rồi!")
```

Vòng lặp while với điều kiện ở cuối

Vòng lặp vô tận với lệnh ***break*** ở cuối, giống vòng lặp ***do***...***while*** trong C, vòng lặp này khối lệnh được thực thi ít nhất một lần.

```
>>> # Tung xúc xắc đến khi người dùng chọn thoát
    import random
    while True:
        input("Nhấn Enter để tung xúc xắc")
        num = random.randint(1,6) # số bất kỳ từ 1 đến 6
        print(" Tung được mặt",num)
        option = input(" Muốn tung tiếp không?(y/n) ")
        # điều kiện
        if option == 'n':
            break
```

Vòng lặp while với điều kiện ở cuối

Kết quả:

```
>>> Nhấn Enter để tung xúc xắc
      Tung được mặt 1
      muốn tung tiếp không?(y/n) y
      Nhấn Enter để tung xúc xắc
      Tung được mặt 4
       Muốn tung tiếp không?(y/n) n
>>>
```

Bài tập

1. Viết chương trình tìm tất cả các số chia hết cho 7 nhưng không phải bội số của 5, nằm trong đoạn 2000 và 3200 (tính cả 2000 và 3200). Các số thu được sẽ được in thành chuỗi trên một dòng, cách nhau bằng dấu phẩy
2. Viết chương trình tính giai thừa của một số cho trước. Ví dụ, nhập số 8 thì kết quả là 40320

Bài tập

3. Nhập số nguyên n, hãy viết chương trình ra một dictionary chứa (i, i*i) phần tử số nguyên từ 1 đến n (bao gồm cả 1 và n). Ví dụ: n là 8 thì kết quả là: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}.

4. Viết chương trình và in giá trị theo công thức: Q = √([(2 * C * D)/H]) (Q bằng căn bậc hai của [(2 nhân C nhân D) chia H]. Với C là 50, H là 30. D là dãy được nhập vào từ bàn phím và phân cách bằng dấu phẩy. Ví dụ: chuỗi D nhập vào là 100,150,180 thì kết quả là 18,22,24

Bài tập

5. Viết chương trình nhập chuỗi là các dòng, chuyển các dòng này thành chữ in hoa và in ra màn hình. Ví dụ nhập vào:
Hello world
Practice makes perfect
Kết quả là:
HELLO WORLD
PRACTICE MAKES PERFECT

Bài tập

6. Viết chương trình nhập vào chuỗi các số nhị phân 4 chữ số, phân tách bởi dấu phẩy, và in các số chia hết cho 5 phân tách bởi dấu phẩy. (**hàm int(i,2)**)
Ví dụ nhập vào: 0100,0011,1010,1001
kết quả là: 1010
7. Viết chương trình tìm tất cả các số trong đoạn 1000 và 3000 (tính cả 2 số này) sao cho tất cả các chữ số trong số đó là số chẵn.
In các số tìm được thành chuỗi cách nhau bởi dấu phẩy, trên một dòng.

Bài tập

8. Viết chương trình nhập vào một câu, đếm số chữ cái và chữ số trong câu đó. Ví dụ, nhập vào câu: hello world! 123 (dict.**isdigit() và** dict**.isalpha()**)
Kết quả là:
Số chữ cái là: 10
Số chữ số là: 3
9. Viết chương trình nhập vào một câu, đếm chữ hoa, chữ thường. (dict**.isupper()** dict**.islower()**)
Ví dụ nhập vào: Lạc Hồng
Kết quả là:
Chữ hoa: 2
Chữ thường: 5

Bài tập

10. Viết chương trình tính số tiền của một tài khoản ngân hàng dựa trên nhật ký giao dịch được nhập vào từ bàn phím. Ví dụ nhập nhật ký được hiển thị như sau:
G 100
R 200
G 300
(G là tiền gửi, R là tiền rút ra).
Kết quả là:
200

Bài tập

11. Viết chương trình sắp xếp tuple (name, age, score) theo thứ tự tăng dần, name là string, age và score là number. Tuple được nhập vào bởi người dùng. Tiêu chí sắp xếp là:
Sắp xếp theo name sau đó sắp xếp theo age, cuối cùng sắp xếp theo score. Thứ tự ưu tiên là tên > tuổi > điểm. (**modul operator import itemgetter**)

Bài tập

12. Một Robot di chuyển trong mặt phẳng bắt đầu từ điểm đầu tiên (0,0). Robot có thể di chuyển theo hướng UP, DOWN, LEFT và RIGHT với số bước nhất định. Ví dụ di chuyển của robot được hiển thị như sau:

UP 5
DOWN 3
LEFT 3
RIGHT 2

Kết quả là: 2
(khoảng cách = sqrt($(x_1-x_0)^2+(y_1-y_0)^2$))

Bài tập

Số sau phía sau hướng di chuyển là số bước đi. Hãy viết chương trình để tính toán khoảng cách từ vị trí hiện tại đến vị trí đầu tiên, sau khi robot đã di chuyển một quãng đường. Nếu khoảng cách là một số thập phân chỉ cần in số nguyên gần nhất.

13. Viết chương trình tính tần suất các từ từ input. Output ra các từ đã sắp xếp theo bảng chữ cái. Giả sử input là: New to Python or choosing between Python 2 and Python 3? Read Python 2 or Python 3. (**sorted(dict.keys())**)

Bài tập

14. Viết chương trình để tạo tuple khác, chứa các giá trị là số chẵn trong tuple (1,2,3,4,5,6,7,8,9,10) cho trước. (**list.append()**)

15. Viết chương trình Python nhận chuỗi nhập vào bởi người dùng, in "Yes" nếu chuỗi là "yes" hoặc "YES" hoặc "Yes", nếu không in "No"

16. Viết chương trình tính 1/2 + 2/3 + 3/4 + ... + n/(n + 1) với n là số được nhập vào (n> 0)

Bài tập

17. Viết chương trình để tạo tất cả các câu có chủ ngữ nằm trong ["Anh",“Tôi"], động từ nằm trong ["Chơi","Yêu"] và tân ngữ là ["Bóng đá","Xếp hình"].