Unity
Thống nhất
Excellence
Vượt trội
Leadership
Tiên phong

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

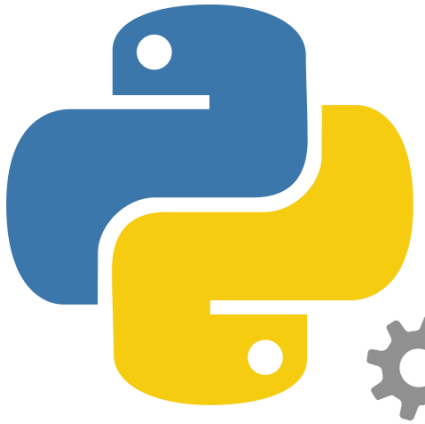
TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT

Môn Học

KỸ THUẬT LẬP TRÌNH VỚI PYTHON

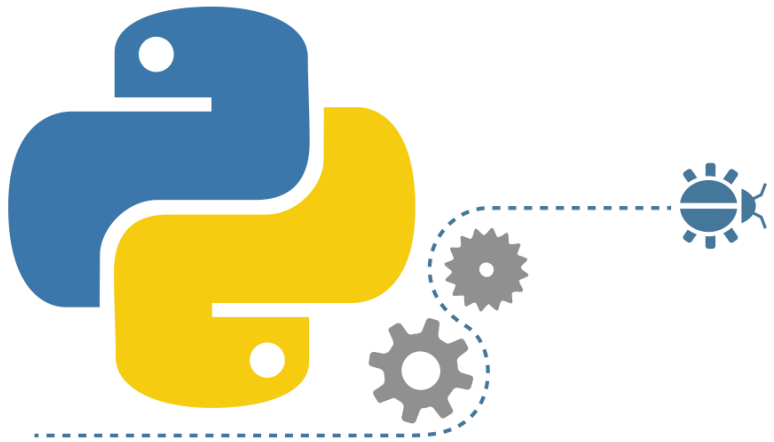
GV: Ths. Trần Duy Thanh

thanhtd@uel.edu.vn



www.uel.edu.vn

CÁC BIỂU THỨC ĐIỀU KIỆN VÀ VÒNG LẶP



Mục tiêu bài học

- Hiểu và vận hành được các cấu trúc điều kiện: boolean, if, else, elif
- Nắm được biểu thức pass
- So sánh được số thực trong Python
- Hiểu và vận hành được cấu trúc vòng lặp: *while, for, while/else, for/else*
- *Hiểu và vận hành được biểu thức break, continue*

Nội dung bài học

- ***3.1.Các cấu trúc điều kiện***
- *3.1.1. Biểu thức Boolean*
- *3.1.2. Biểu thức If*
- *3.1.3. Biểu thức if ... else*
- *3.1.4. Biểu thức If ... elif lồng nhau*
- *3.1.5. Biểu thức pass*
- *3.1.6. So sánh số thực trong Python*
- *3.1.7. Sử dụng if/else như phép gán*
- ***3.2.Các cấu trúc lặp***
- *3.2.1. Vòng while*
- *3.2.2. Vòng for*
- *3.2.3. câu lệnh break*
- *3.2.4. câu lệnh continue*
- *3.2.5. Lệnh while/else*
- *3.2.6. Lệnh for/else*
- *3.2.7. Vòng lặp lồng nhau*

Nội dung bài học

3.1. Các cấu trúc điều kiện

3.1.1. Biểu thức Boolean

3.1.2. Biểu thức If

3.1.3. Biểu thức if ... else

3.1.4. Biểu thức If ... elif lồng nhau

3.1.5. Biểu thức pass

3.1.6. So sánh số thực trong Python

3.1.7. Sử dụng if/else như phép gán

3.1.1. Biểu thức Boolean

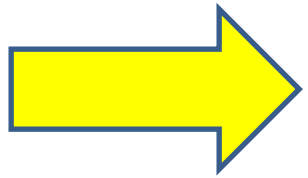
Biểu thức Boolean (Boolean Expression) còn được gọi là Predicate. Là một biểu thức rất quan trọng và phổ biến trong các lệnh của Python cũng như ngôn ngữ lập trình khác.

Các giá trị là **True** hoặc **False**, dựa vào các giá trị này mà ta điều hướng các công việc trong phần mềm.

3.1.1. Biểu thức Boolean

Ví dụ:

```
a = True
b = False
print('a =', a, ' b =', b)
# gán lại kết qua cho a
a = False
print('a =', a, ' b =', b)
```



```
a = True b = False
a = False b = False
```

3.1.1. Biểu thức Boolean

Bảng tổng quát:

Biểu thức	Ý nghĩa
x == y	True nếu x=y, False nếu x khác y
x < y	True nếu x <y, False nếu x>=y
x <= y	True nếu x<=y, False nếu x>y
x > y	True nếu x>y, False nếu x<=y
x >= y	True nếu x>=y, False nếu x<y
x != y	True nếu x khác y, False nếu x=y

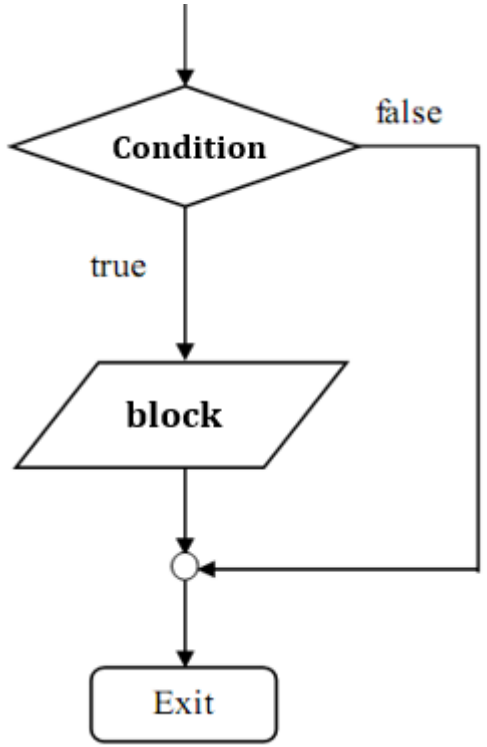
3.1.1. Biểu thức Boolean

Ví dụ:

Biểu thức	Ý nghĩa
10 < 20	True
10 >= 20	False
x < 100	True if x nhỏ hơn100; ngược lại False

3.1.2. Biểu thức If

Biểu thức if là một biểu thức điều kiện rất quan trọng và phổ biến trong Python. Biểu thức if đứng một mình chỉ quan tâm tới điều kiện đúng (True). Khi điều kiện đúng thì khối lệnh bên trong if sẽ được thực thi.



Cú pháp:

```
if condition :
    block
```

3.1.2. Biểu thức If

Ví dụ:

```
dtb=float(input("Nhập điểm trung bình:"))
if dtb>=5:
    print("Bạn đã đậu!")
    print("Hú hồn!")
```

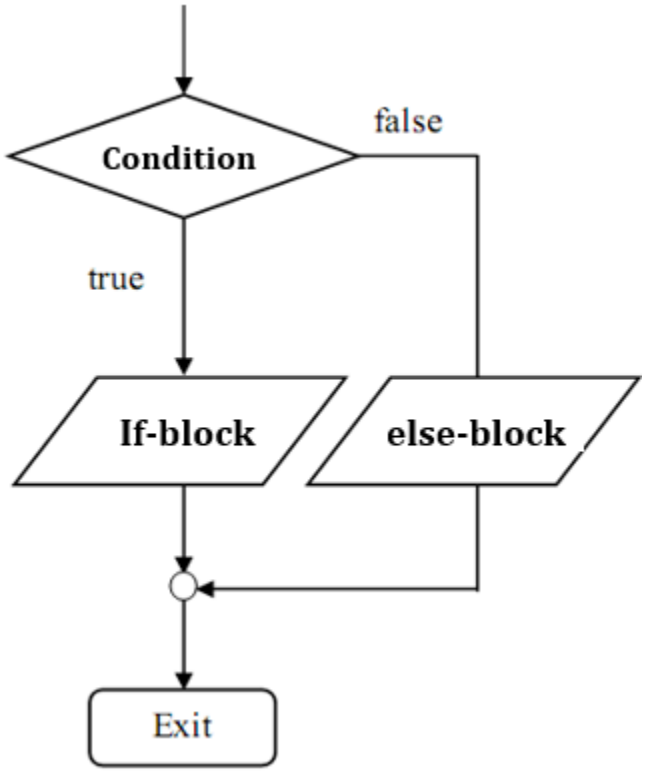
Lưu ý rằng Python không dùng ngoặc nhọn để bao bọc các dòng lệnh, mà ta dùng phím Tab hoặc khoảng trắng thụt đầu dòng. Tui nghĩ ta nên dùng Tab cho nó lẹ.

3.1.3. Biểu thức if ... else

Biểu thức if...else là một biểu thức điều kiện rất quan trọng và phổ biến trong Python. Biểu thức này quan tâm điều kiện đúng(True) và sai(False). Nó phổ biến hơn biểu thức if.

Cú pháp:

```
if condition :
    if-block
else:
    else-block
```



3.1.3. Biểu thức if ... else

Ví dụ:

```
dtb = float(input("Nhập điểm trung bình:"))
if dtb >= 5:
    print("Bạn đã đậu!")
    print("Hú hồn!")
else:
    print("Ở nhà lấy Vợ")
    print("Đi phụ hồ")
```

3.1.4. Biểu thức If ... elif lồng nhau

Với các điều kiện thức tạp, Python cũng hỗ trợ kiểm tra điều kiện if elif lồng nhau:

```
dtb = float(input("Nhập điểm trung bình:"))
if dtb >= 9:
    print("Bạn xếp loại giỏi")
elif dtb>=7:
    print("Bạn xếp loại khá")
elif dtb>=5:
    print("Bạn xếp loại Trung bình")
else:
    print("Chia tay hoàng hôn")
```

3.1.5. Biểu thức pass

Biểu thức pass khá lợi hại, nó dùng để dành chỗ lập trình. Ví dụ bạn biết chỗ đó phải viết rất nhiều coding, nhưng tại thời điểm này chưa kịp làm. Ta sẽ dùng pass để đánh dấu vị trí đó.

```
a=float(input("Nhập hệ số a:"))
b=float(input("Nhập hệ số b:"))
if a==0:

else:
    x=-b/a;
    print("{0}x+{1}=0".format(a,b))
    print("có nghiệm x=",x)
```

Lỗi, Python không cho để trống như thế này

```
a=float(input("Nhập hệ số a:"))
b=float(input("Nhập hệ số b:"))
if a==0:
    pass
else:
    x=-b/a;
    print("{0}x+{1}=0".format(a,b))
    print("có nghiệm x=",x)
```

3.1.6. So sánh số thực trong Python

Khi ta thao tác với số thực thì có một chút rắc rối ở chỗ Sai Số, nên ta cần có một ngưỡng Sai Số cho phép (tùy thuộc vào quyết định của người dùng)

```
1   d1 = 1.11 - 1.10
2   d2 = 2.11 - 2.10
3   print('d1 =', d1, ' d2 =', d2)
4   if d1 == d2:
5       print('d1 và d2 bằng nhau')
6   else:
7       print('d1 và d2 khác nhau')
```

Nếu mắt thường quan sát ta tưởng rằng d1 bằng d2 vì lý do sau:
1.11-1.10➔0.01
2.11-2.10➔0.01
Nhưng thực ra số thực nó có sai số, nó không phải 100% là 0.01

d1 = 0.010000000000000009 d2 = 0.009999999999999787
d1 và d2 khác nhau

3.1.6. So sánh số thực trong Python

Do đó ta cho 1 cái ngưỡng so sánh theo sai số cho phép, ví dụ:

```
1   d1 = 1.11 - 1.10
2   d2 = 2.11 - 2.10
3   print('d1 =', d1, ' d2 =', d2)
4   diff = d1 - d2 # Compute difference
5   if diff < 0: # Compute absolute value
6       diff = -diff
7   if diff < 0.0000001: # Are the values close enough?
8       print('Same')
9   else:
10      print('Different')
```

3.1.7. Sử dụng if/else như phép gán

Đôi khi việc thực hiện if else trong một biểu thức quá đơn giản sẽ làm cho if else phức tạp không cần thiết:

```
a=5
b=7
if a != b:
    c = 113
else:
    c = 115
print(c)
```

```
c = 113 if a != b else 115
```

expression-1 **if** *condition* **else** *expression-2*

Nội dung bài học

3.2. Các cấu trúc lặp

3.2.1. Vòng while

3.2.2. Vòng for

3.2.3. câu lệnh break

3.2.4. câu lệnh continue

3.2.5. Lệnh while/else

3.2.6. Lệnh for/else

3.2.7. Vòng lặp lồng nhau

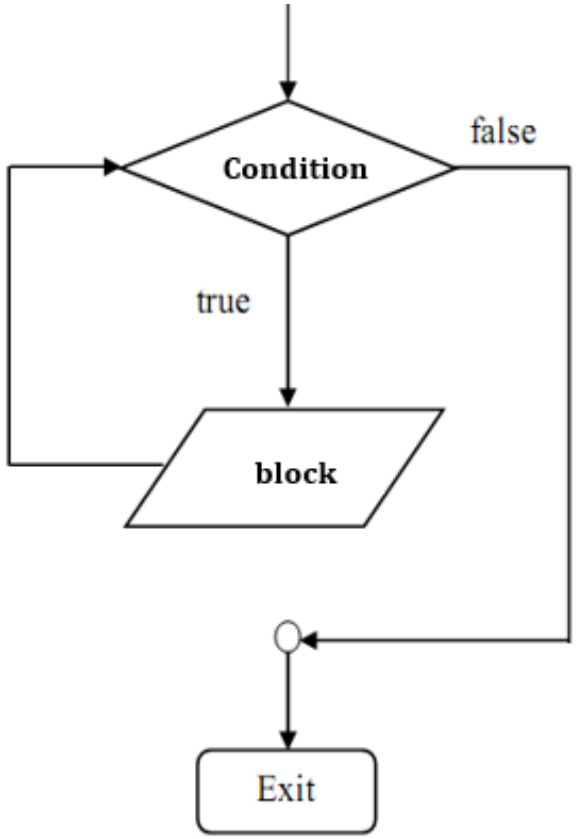
3.2.1. Vòng while

while dùng để yêu cầu 1 công việc được lặp đi lặp lại

Nếu **condition** là True thì **block** sẽ được lặp lại

Cú pháp:

```
while condition :
    block
```



-Có thể block sẽ không được thực hiện lần nào nếu condition là False ngày từ đầu

-Ta có thể kết thúc vòng while bằng cách đưa condition về False hoặc dùng từ khóa **break** để thoát.

3.2.1. Vòng while

Ví dụ:

Viết chương trình yêu cầu nhập vào một số nguyên dương [1..10], nếu nhập sai yêu cầu nhập lại. Khi nhập đúng thì xuất ra bình phương của giá trị mới nhập vào.

```
1    value=-1;
2    while value < 1 or value > 10:
3        value=int(input("Nhập giá trị [1..10]:"))
4    print("value=",pow(value,2));
```

3.2.1. Vòng while

```
1  #s=1+2+3+...+N
2  print("Nhập N:")
3  n=int(input())
4  s=0
5  i=1
6  while i<=n:
7      s=s+i
8      i=i+1
9  print("Tổng =",s)
```

Khởi tạo: s=0, i=1, n=5

1) i<=n ⇔1 <=5➔True
   s=s+i=0+1=1; i=i+1=1+1=2

2) i<=n ⇔ 2 <=5 ➔True
   s=s+i=1+2=3; i=i+1=2+1=3

3) i<=n ⇔3 <=5➔True
   s=s+i=3+3=6; i=i+1=3+1=4

4) i <=n ⇔4 <=5➔True
   s=s+i=6 +4=10; i=i+1=4+1=5

5)i<=n ⇔5<=5➔True
   s=s+i=10+5=15; i=i+1=5+1=6

6) i<=n ⇔6 <=5➔False➔Dừng while

➔xuất tổng = 15

3.2.2. Vòng for

for dùng để lặp tuần tự các công việc, for sử dụng range để định nghĩa vùng dữ liệu lặp và bước lặp

Cú pháp hàm range:

range(*begin* , *end* , *step*)

begin: Giá trị bắt đầu

end: Giá trị cuối

step: Bước nhảy

Ví dụ cách hoạt động của range:

- range(10) ➔ 0; 1; 2; 3; 4; 5; 6; 7; 8; 9
- range(1, 10) ➔ 1; 2; 3; 4; 5; 6; 7; 8; 9
- range(1, 10, 2) ➔ 1; 3; 5; 7; 9
- range(10, 0, -1) ➔ 10; 9; 8; 7; 6; 5; 4; 3; 2; 1
- range(10, 0, -2) ➔ 10; 8; 6; 4; 2
- range(2, 11, 2) ➔ 2; 4; 6; 8; 10

3.2.2. Vòng for

Các Ví dụ về for:

```
for n in range(10):
    print(n,end=' ')
```
→ 0 1 2 3 4 5 6 7 8 9

```
for n in range(1,10):
    print(n,end=' ')
```
→ 1 2 3 4 5 6 7 8 9

```
for n in range(1,10,2):
    print(n,end=' ')
```
→ 1 3 5 7 9

```
for n in range(10,0,-1):
    print(n,end=' ')
```
→ 10 9 8 7 6 5 4 3 2 1

```
for n in range(10,0,-2):
    print(n,end=' ')
```
→ 10 8 6 4 2

```
for n in range(2,11,2):
    print(n,end=' ')
```
→ 2 4 6 8 10

3.2.2. Vòng for

```
1  n=int(input("Mời nhập sô:"))
2  s=0
3  if n % 2==0:
4      for x in range(2,n+1,2):
5          s=s+x
6  elif n%2!=0:
7      for x in range(1,n+1,2):
8          s=s+x
9  print("Tổng s=",s)
```

N=8, s=0

N=8 là số chẵn n%2==0=➔8%2=0

1) x=2➔s=s+x=0+2=2
2) x=4➔s=s+x=2+4=6
3) X=6➔s=s+x=6+6=12
4) X=8➔s=s+x=12+8=20
5) X=10➔thấy 10>n+1=9

3.2.3. câu lệnh break

break là từ khóa đặc biệt dùng để thoát khỏi vòng lặp chứa nó trực tiếp khi đạt được mức yêu cầu nào đó.

Gặp lệnh break, chương trình sẽ không thực hiện bất cứ lệnh nào bên dưới nó, mà thoát luôn khỏi vòng lặp.

3.2.3. câu lệnh break

Ví dụ: Viết chương trình vòng lặp vĩnh cửu cho phép phần mềm chạy liên tục, khi nào hỏi thoát mới thoát phần mềm:

```
while True:
    a=int(input("Nhập giá trị:"))
    print("Giá trị bạn nhập ",a)
    s=input("Tiếp tục phần mềm không?(c/k):")
    if s=="c":
        break
print("BYE!")
```

3.2.4. câu lệnh continue

continue là từ khóa đặc biệt dùng để nhảy sớm tới lần lặp kế tiếp, các lệnh bên dưới continue sẽ không được thực thi. Lưu ý khác với break, gặp break thì ngừng luôn vòng lặp, gặp continue chỉ dừng lần lặp hiện tại đang dở dang để chuyển qua lần lặp tiếp theo.

3.2.4. câu lệnh continue

Ví dụ:

Tính tổng các chữ số lẻ từ 1->15, ngoại trừ số 3 và số 11

```
sum=0
for n in range(1,16,2):
    if n is 3 or n is 11:
        continue
    sum+=n
print(sum)
```

3.2.5. Lệnh while/else

Python hỗ trợ else block trong trường hợp while kết thúc một cách bình thường (tức là không phải dùng break để kết thúc)

```
while condition:
    while-block
else:
    else-block
```

Nếu while kết thúc một cách bình thường thì else-block sẽ tự động được thực hiện ngay sau đó.

3.2.5. Lệnh while/else

Ví dụ:

```
count = sum = 0
print('Nhập danh sách các số dương để tính trung')
while count < 5:
    val = float(input('Nhập số: '))
    if val < 0:
        print('Số 0 sai quy tắc, thoát phần mềm')
        break
    count += 1
    sum += val
else:
    print('Trung Bình =', sum/count)
```

Nếu nhập số <0 thì sẽ thoát while và bên trong else cũng không được thực thi (do kết thúc bằng lệnh break). Nếu nhập đúng toàn bộ giá trị, sau khi while chạy xong thì else sẽ tự động được gọi(kết thúc while bình thường)

3.2.6. Lệnh for/else

Python hỗ trợ else block trong trường hợp for kết thúc một cách bình thường (tức là không phải dùng break để kết thúc)

```
for expression:
    for-block
else:
    else-block
```

Nếu for kết thúc một cách bình thường thì else-block sẽ tự động được thực hiện ngay sau đó.

3.2.6. Lệnh for/else

Ví dụ:

```
a=int(input("Nhập a:"))
s=0
for n in range(5,10):
    if 4%a is 1:
        print("Ngừng for")
        break
    s=s+n
else:
    print("Sum=",s)
```

Nếu nhập a là số chẵn thì tính ra sum, còn số lẻ không ra sum(do ngừng for bằng break)

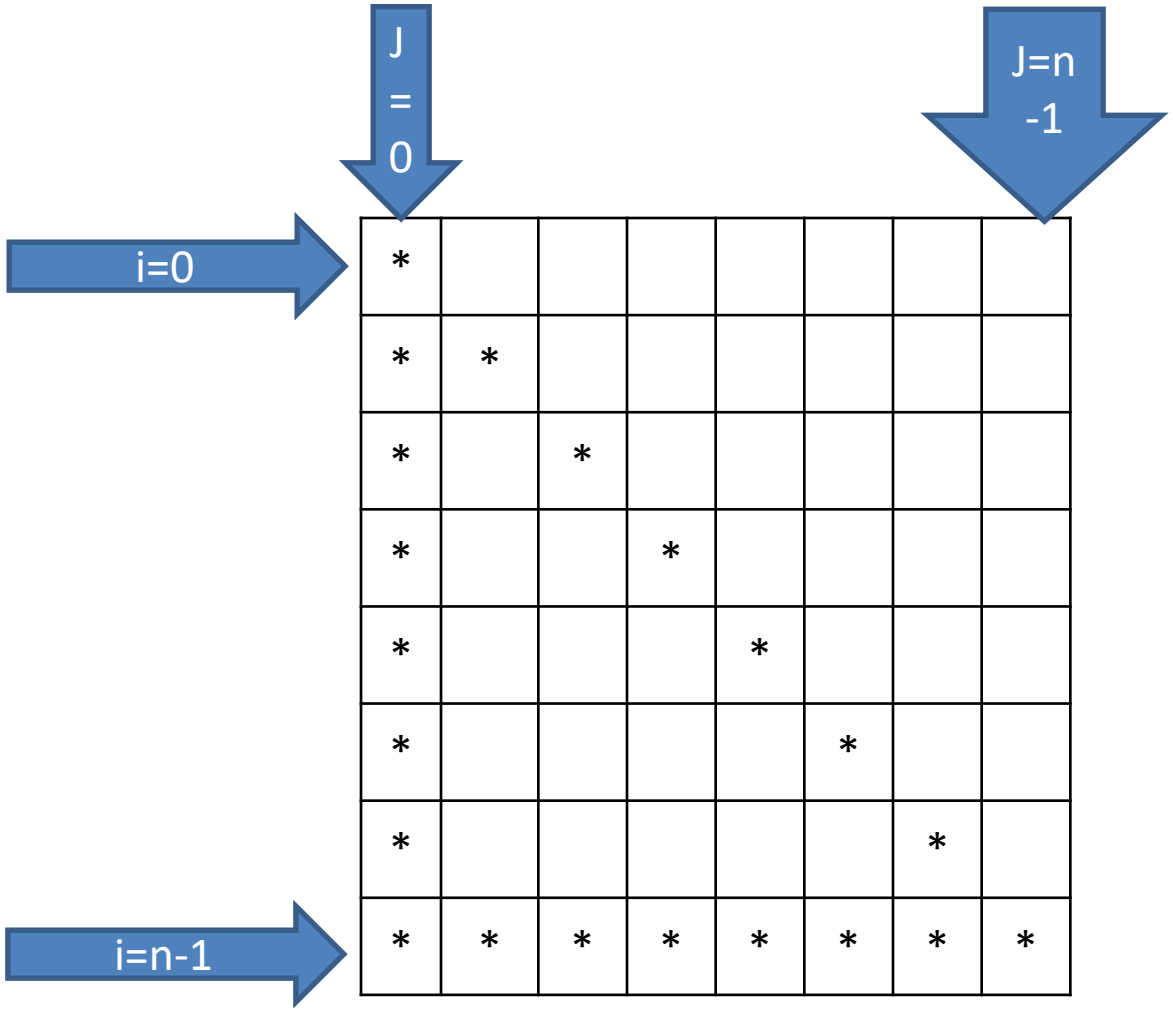
3.2.7. Vòng lặp lồng nhau

Python cũng như các ngôn ngữ khác, ta có thể viết các vòng lặp lồng nhau.

```
n=int(input("Nhập chiều cao:"))
for i in range(n):
    for j in range(n):
        if j==0 or i==j or j==n-1:
            print("*",end='')
        else:
            print(" ",end='')
    print()
```

```
Nhập chiều cao:7
*     *
**    *
* *   *
*  *  *
*   * *
*    **
*     *
```

3.2.7. Vòng lặp lồng nhau



THANK YOU

028 37244555 www.uel.edu.vn

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT

Số 669, đường Quốc lộ 1, khu phố 3, phường Linh Xuân,
quận Thủ Đức, Thành phố Hồ Chí Minh.

UNITY - EXCELLENCE - LEADERSHIP