TRƯỜNG THPT NGUYỄN THỊ MINH KHAI
TỔ TIN HỌC

CHỦ ĐỀ F. LẬP TRÌNH CƠ BẢN
LẬP TRÌNH PYTHON (PHẦN 1)

1. Biến

- Biến là tên một vùng nhớ, trong quá trình thực hiện chương trình, giá trị của biến có thể thay đổi
- Quy tắc đặt tên biến:
  - Không trùng với từ khóa (được sử dụng với ý nghĩa xác định không thay đổi
  - Chỉ chứa chữ cái, chữ số và dấu "_" (gạch dưới)
  - Bắt đầu bằng chữ cái hoặc dấu "_" (gạch dưới)
  - Phân biệt chữ hoa và thường

  VD: những tên biến đặt đúng: lop, x1, _10a1

2. Hằng

- Hằng là những biến có giá trị chỉ định trước và không thể thay đổi trong quá trình thực hiện chương trình
- Python không cung cấp công cụ khai báo hằng
- Khi lập trình bằng Python, người ta thường sử dụng hằng số như một loại biến với cách đặt tên đặc biệt

  Ví dụ: bắt đầu bằng dấu gạch dưới và sau đó là các kí tự La tinh in hoa, gán giá trị cần thiết cho nó và tự quy ước không gán lại giá trị cho các biến đó

3. Chú thích trong python

- Khi soạn thảo chương trình, ngoài các câu lệnh, người lập trình có thể viết thêm các dòng chú thích.
- Các dòng chú thích không ảnh hưởng đến nội dung chương trình mà chỉ giúp cho người đọc nhanh chóng biết được mục đích của các câu lệnh và ý nghĩa của chương trình.
- Trong Python, thông tin chú thích viết trên một dòng, bắt đầu bằng kí tự #.
- Nhờ kí tự đánh dấu đó mà máy tính nhận biết được dòng chú thích

VD:

```
x1 = (-b - math.sqrt(b*b - 4*a*c))/(2*a)
x2 = -b/a - x1    # Định lí Viet
```

4. Lệnh gán

- Cú pháp:

Biến = <Biểu thức>

Trong đó: <Biểu thức>: thường gặp là biểu thức số học. Biểu thức số học có thể là một số, một tên biến hoặc các số và biến liên kết với nhau bởi các phép toán số học

VD: x=5 # biến x có giá trị là 5
A=b+1 # biến A có giá trị là kết quả phép toán của b+1
_PI = 3.1416 # Sử dụng như hằng π = 3.1416

5. Các phép toán số học

Phép toán	Kí hiệu trong Python	Ví dụ
Cộng	+	3 + 12 = 15
Trừ	-	15 – 3 = 12
Nhân	*	12 * 5 = 60
Chia	/	16 / 5 = 3.2
Chia lấy phần nguyên	//	16 // 5 = 3
Chia lấy phần dư	%	16 % 5 = 1
Lũy thừa	**	2 ** 3 = 8

❖ Lưu ý:

- Các phép toán được thực hiện theo thứ tự như trong toán học
- Trong biểu thức chỉ sử dụng các cặp ngoặc tròn để xác định thứ tự thực hiện các phép tính
- Trước và sau mỗi tên biến, mỗi số hoặc dấu phép tính có thể có số lượng tùy ý các dấu cách (dấu trắng)

6. Một số hàm toán học thường dụng

- Cách sử dụng hàm toán học:
  - Hàm **abs( )** sử dụng trực tiếp.
  - Các hàm còn lại ta cần đưa vào chương trình câu lệnh **import math** trước khi gọi hàm lần đầu tiên
  - Lời gọi hàm có dạng: **math.<tên_hàm>**
- Các hàm toán học thông dụng:

Hàm	Ý nghĩa toán học
abs(x)	Tính $\lvert x \rvert$
ceil(x)	Trả về số nguyên nhỏ nhất, lớn hơn hoặc bằng giá trị x
gcd(x, y)	Tính ước chung lớn nhất của số nguyên x và y

sqrt(x)	Tính căn bậc hai của x
log(x)	Tính lnx
exp(x)	Tính e^x

VD:

```
import math
x = math.sqrt(5)
```

7. Kiểu dữ liệu

- Một số kiểu dữ liệu cơ bản của Python bao gồm: **int** (số nguyên), **float** (số thực), **str** (xâu kí tự), **bool** (lôgic).
- Lệnh **type( )** dùng để nhận biết kiểu dữ liệu của biến trong Python.

VD:

```
>>> a = 5
>>> print(type(a))
<class 'int'>
>>> a = 5.0
>>> print(type(a))
<class 'float'>
>>> print(type(5/3))
<class 'float'>
```

8. Câu lệnh vào – ra đơn giản

❖ **Câu lệnh vào:**

- Lệnh nhập giá trị vào từ bàn phím:

Biến = **input**(*dòng thông báo*)

VD: a= input()
a= input(‘hãy nhập vào giá trị:’)
→ biến a sẽ là dữ liệu kiểu xâu kí tự

- Dữ liệu nhập vào có dạng xâu kí tự. Nếu muốn chuyển dữ liệu này sang số nguyên hay thực để tính toán cần có câu lệnh int() hay float() như sau:
  - Biến = **int**(input(*dòng thông báo*))
  - Biến = **float**(input(*dòng thông báo*))

VD:

```
a=int(input("nhập vào 1 số nguyên= "))
b=float(input("nhập vào 1 số thực= "))
c=a+b
print(c)
```

❖ **Câu lệnh ra**

- Cú pháp:

print(*danh sách biểu thức***)**

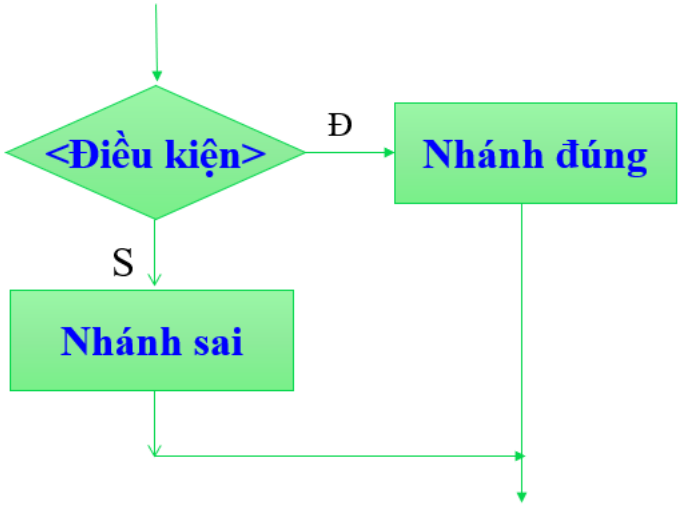
Trong đó: Danh sách biểu thức: là các biểu thức viết cách nhau bởi dấu ",". Câu lênh print() sẽ in ra màn hình giá trị các biểu thức theo đúng thứ tự và cách nhau bởi dấu cách

VD:

```
t = van + li + sinh
print("Tổng ba môn: ",t,"trung bình: ",t/3)
```

9. Câu lệnh rẽ nhánh

❖ **Rẽ nhánh là gì?**



❖ **Điều kiện rẽ nhánh**

- *<điều kiện>:* là biểu thức nhận giá trị logic True hoặc False
  - Biểu thức so sánh:
- Sử dụng các phép so sánh:

So sánh	Kí hiệu trong Python
Lớn hơn	>
Lớn hơn hoặc bằng	>=
Nhỏ hơn	<
Nhỏ hơn hoặc bằng	<=
Bằng	==
Khác	!=

VD:

Điều kiện	Giá trị logic của điều kiện với A = 5, B = 10
A < B	True
A*A + B*B <= 100	False
A + 5 != B	False
2*A == B	True

- Biểu thức logic

- Sử dụng các phép logic:

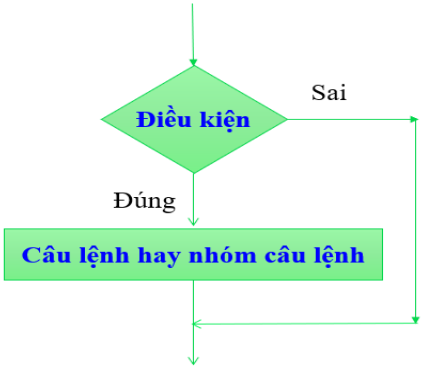
Phép tính	Biểu thức	Ý nghĩa
and	x and y	Cho kết quả True khi và chỉ khi x và y đều nhận giá trị True
or	x or y	Cho kết quả False khi và chỉ khi x và y đều nhận giá trị False
not	not x	Đảo giá trị logic của x

VD:

Điều kiện	Giá trị của biểu thức logic điều kiện A = 5, B = 10
(A < B) and (A + 5 != B)	False
(3*A > B) or (2*A == B)	True
not (A*A + B*B <= 100)	True

❖ Cú pháp rẽ nhánh dạng thiếu

if *<điều kiện>*:
câu lệnh hay nhóm câu



VD:

```
t = 9
if t<10:
    print(t,"không phải là số nguyên dương có hai chữ số")
```

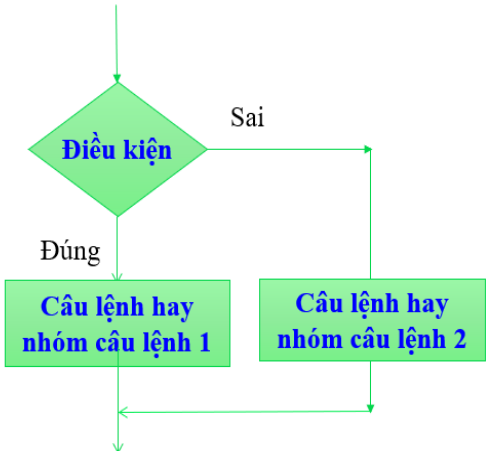
❖ **Cú pháp rẽ nhánh dạng đủ**

if *<điều kiện>*:

câu lệnh hay nhóm câu lệnh 1

else:

câu lệnh hay nhóm câu lệnh 2



VD:

```
A = int(input("Nhập vào một số nguyên: "))
if  A%2 == 0 :
    print(A,"là số chẵn")
else:
    print(A,"là số lẻ")
```

❖ **Cú pháp rẽ nhánh lồng nhau**

if *<điều kiện1>*:

câu lệnh hay nhóm câu lệnh 1

elif *<điều kiện2>*:

câu lệnh hay nhóm câu lệnh 2

else:

câu lệnh hay nhóm câu lệnh 3

VD:

```
if BMI <=18.5:
    print('thiếu cân')
elif BMI <=22.9:
    print('bình thường')
else:
    print('thừa cân')
```

10. Câu lệnh lặp

❖ **Lặp với số lần biết trước FOR**

- Cú pháp:

for *biến_chạy* ***in*** *range(m, n)*:
khối lệnh lặp

Trong đó:

- *biến_chạy* : biến do người lập trình đặt, sẽ được gán lần lượt các giá trị trong vùng giá trị của lệnh range() và thực hiện <khối lệnh lặp>. VD: i, j, a…
- *range(m, n):*
  - Dùng để khởi tạo dãy số nguyên từ m đến n – 1 (với m < n)
  - m hiểu là giá trị bắt đầu (start), n hiểu là giá trị dừng (stop)
  - Trường hợp m = 0, hàm range(m, n) có thể viết gọn là range(n)

VD: chạy xuất ra 10 lần dòng lệnh "hello"

- Cách 1:

```
for i in range(1,11):
    print('hello')
```

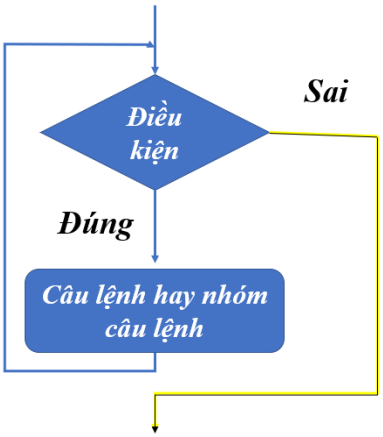
- Cách 2:

```
for i in range(10):
    print('hello')
```

❖ Lặp với số lần không biết trước WHILE

while *<điều kiện>***:**

Câu lệnh hay nhóm câu lệnh



VD:

```
sodem = 1
while sodem <= 6:
    print(sodem)
    sodem = sodem + 1
```