

# Ngôn Ngữ Lập Trình Python

## Cấu Trúc Rẽ Nhánh, Vòng Lặp

# Nội Dung

- Cấu trúc rẽ nhánh
- Vòng lặp while
- Vòng lặp for
- Lệnh break, continue
- Function
- Bài tập

# Cấu trúc rẽ nhánh if-else

```
if expression:
```

```
    # If-block
```

```
if expression:
```

```
    # If-block
```

```
elif 2-expression:
```

```
    # 2-if-block
```

```
elif 3-expression:
```

```
    # 3-if-block
```

```
...
```

```
elif n-expression:
```

```
    # n-if-block
```

```
if expression:
```

```
    # If-block
```

```
else:
```

```
    # else-block
```

```
if expression:
```

```
    # If-block
```

```
elif 2-expression:
```

```
    # 2-if-block
```

```
...
```

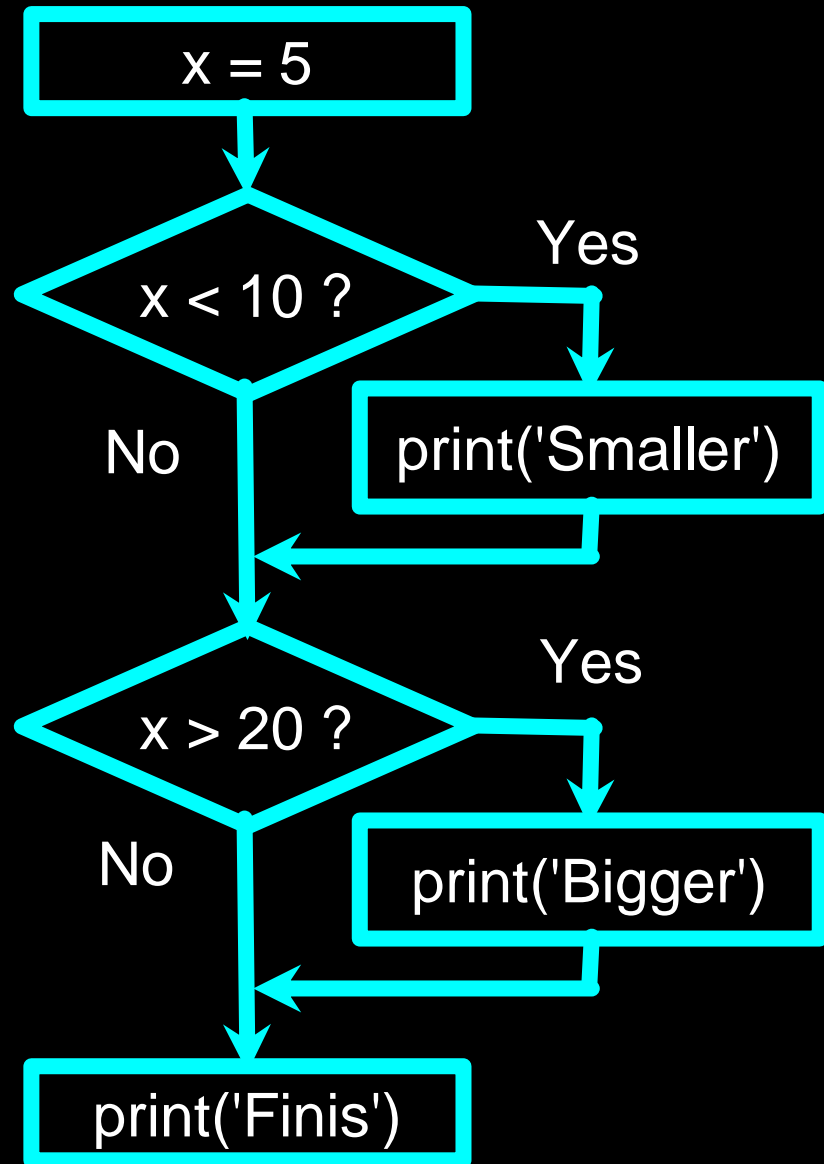
```
elif n-expression:
```

```
    # n-if-block
```

```
else:
```

```
    # else-block
```

# Conditional Steps



Program:

```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')

print('Finis')
```

Output:

**Smaller**  
**Finis**

# Comparison Operators

- **Boolean expressions** ask a question and produce a Yes or No result which we use to control program flow
- **Boolean expressions** using **comparison operators** evaluate to True / False or Yes / No
- Comparison operators look at variables but do not change the variables

Python	Meaning
<	Less than
<=	Less than or Equal to
==	Equal to
>=	Greater than or Equal to
>	Greater than
!=	Not equal

Remember: “=” is used for assignment.

[http://en.wikipedia.org/wiki/George\\_Boole](http://en.wikipedia.org/wiki/George_Boole)

# Cấu trúc rẽ nhánh if-else

- Chú ý: python nhạy cảm với việc viết khối mã

```
num = 3
if num > 0:
    print(num, "là số dương.")
print("Thông điệp này luôn được in.")

num = -1
if num > 0:
    print(num, "là số dương.")
print("Thông điệp này cũng luôn được in.")
```

# Cấu trúc rẽ nhánh if-else

```
# Kiểm tra xem số âm hay dương  
# Và hiển thị thông báo phù hợp
```

```
num = 3
```

```
if num >= 0:  
    print("Số dương hoặc bằng 0")  
else:  
    print("Số âm")
```

```
num1=-1
```

```
if num1 >= 0:  
    print("Số dương hoặc bằng 0")  
else:  
    print("Số âm")
```

# Cấu trúc rẽ nhánh if-else

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



# Cấu trúc rẽ nhánh if-else

- Chú ý: python nhạy cảm với việc viết khối mã

```
num = float(input("Nhập một số: "))
if num >= 0:
    if num == 0:
        print("Số 'Không'")
    else:
        print("Số 'dương'")
else:
    print("Số 'âm'")
```

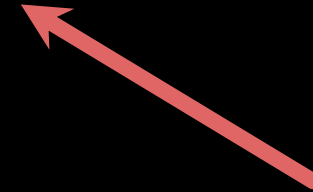
# The try / except Structure

- You surround a dangerous section of code with `try` and `except`
- If the code in the `try` works - the `except` is skipped
- If the code in the `try` fails - it jumps to the `except` section

```
$ cat notry.py
astr = 'Hello Bob'
istr = int(astr)
print('First', istr)
astr = '123'
istr = int(astr)
print('Second', istr)
```

```
$ python3 notry.py
```

```
Traceback (most recent call last):
File "notry.py", line 2, in <module>
istr = int(astr)ValueError: invalid literal
for int() with base 10: 'Hello Bob'
```



All  
Done

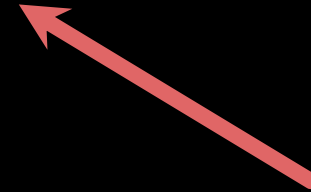
The  
program  
stops  
here



```
$ cat notry.py
astr = 'Hello Bob'
istr = int(astr)
print(100 + istr)
```

```
$ python3 notry.py
```

```
Traceback (most recent call last):
File "notry.py", line 2, in <module>
istr = int(astr)ValueError: invalid literal
for int() with base 10: 'Hello Bob'
```



All  
Done

# Vòng lặp while

```
while biểu_thức:  
    câu_lệnh
```

```
x=0  
#define a while loop  
while(x <4):  
    print(x)  
    x = x+1
```

# Vòng lặp while

- Sử dụng while để tính tổng các số từ 1 đến n

```
n = int(input("Nhập n: ")) #Nhập số n tùy ý
tong = 0 #khởi báo và gán giá trị cho tong
i = 1 #khởi báo và gán giá trị cho biến đếm i

while i <= n:
    tong = tong + i
    i = i+1 # cập nhật biến đếm

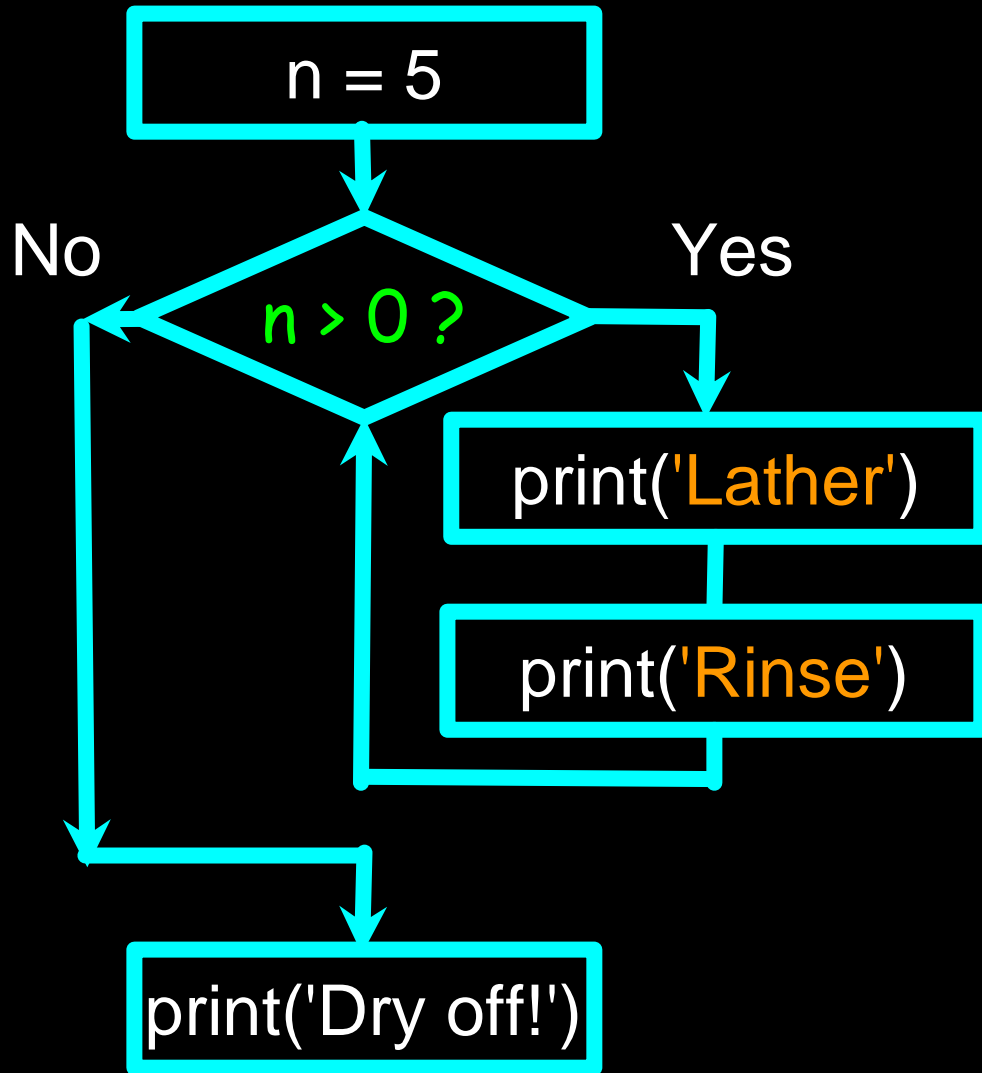
print("Tổng là", tong)
```

# Vòng lặp while

- Kết hợp while với else

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

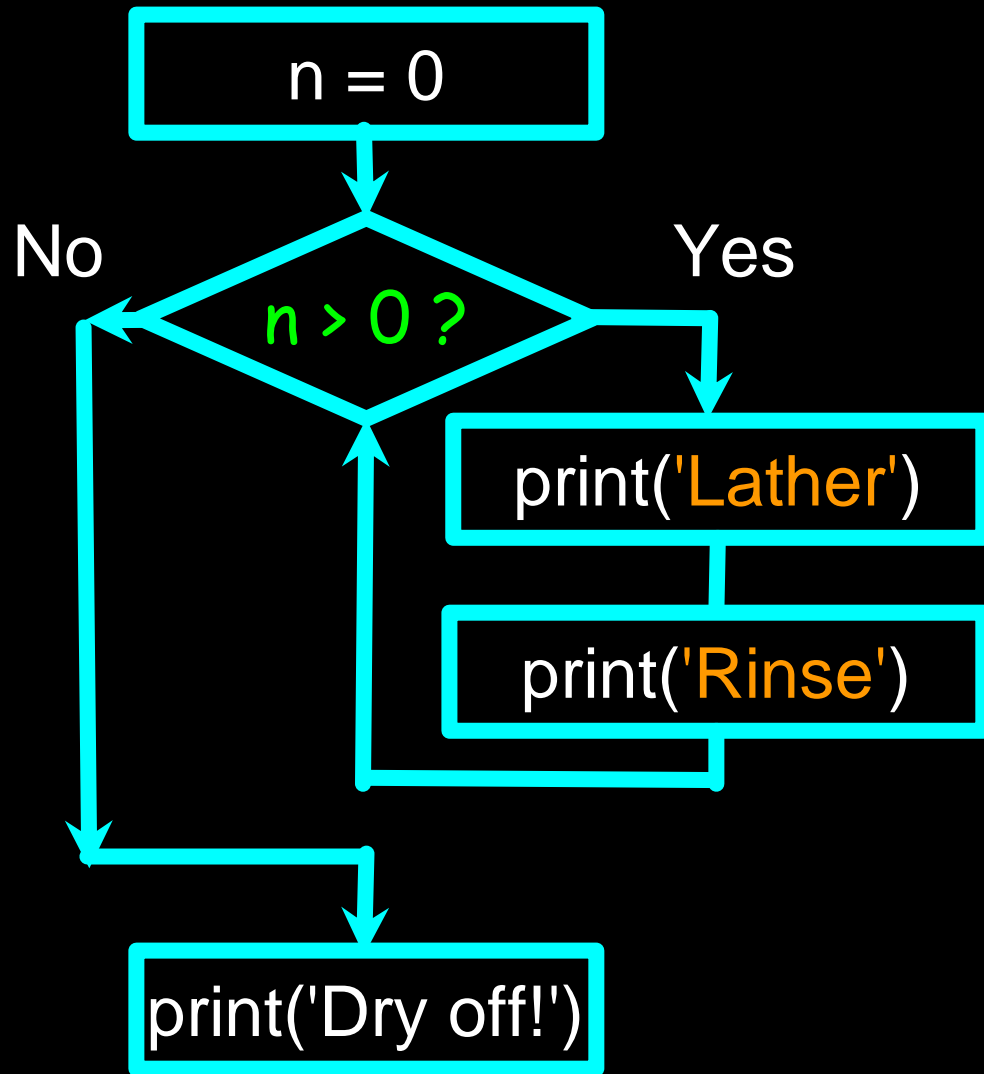
# An Infinite Loop



```
n = 5
while n > 0 :
    print('Lather')
    print('Rinse')
    print('Dry off!')
```

What is wrong with this loop?





# Another Loop

```
n = 0
while n > 0 :
    print('Lather')
    print('Rinse')
print('Dry off!')
```

What is this loop doing?

# Breaking Out of a Loop

- The **break** statement ends the current loop and jumps to the statement immediately following the loop
- It is like a loop test that can happen anywhere in the body of the loop

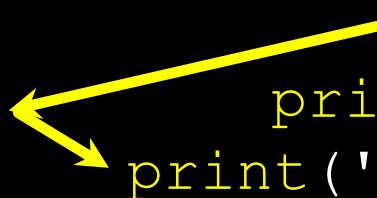
```
while True:
    line = input('> ')
    if line == 'done' :
        break
    print(line)
print('Done!')
```

```
> hello there
hello there
> finished
finished
> done
Done!
```

# Breaking Out of a Loop

- The **break** statement ends the current loop and jumps to the statement immediately following the loop
- It is like a loop test that can happen anywhere in the body of the loop

```
while True:
    line = input('> ')
    if line == 'done' :
        break
    print(line)
print('Done!')
```



```
> hello there
hello there
> finished
finished
> done
Done!
```

# Finishing an Iteration with **continue**

The **continue** statement ends the current iteration and jumps to the top of the loop and starts the next iteration

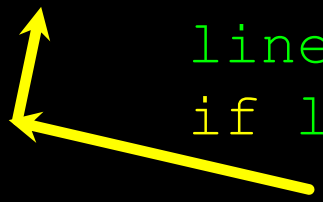
```
while True:
    line = input('> ')
    if line[0] == '#' :
        continue
    if line == 'done' :
        break
    print(line)
print('Done!')
```

```
> hello there
hello there
> # don't print this
> print this!
print this!
> done
Done!
```

# Finishing an Iteration with **continue**

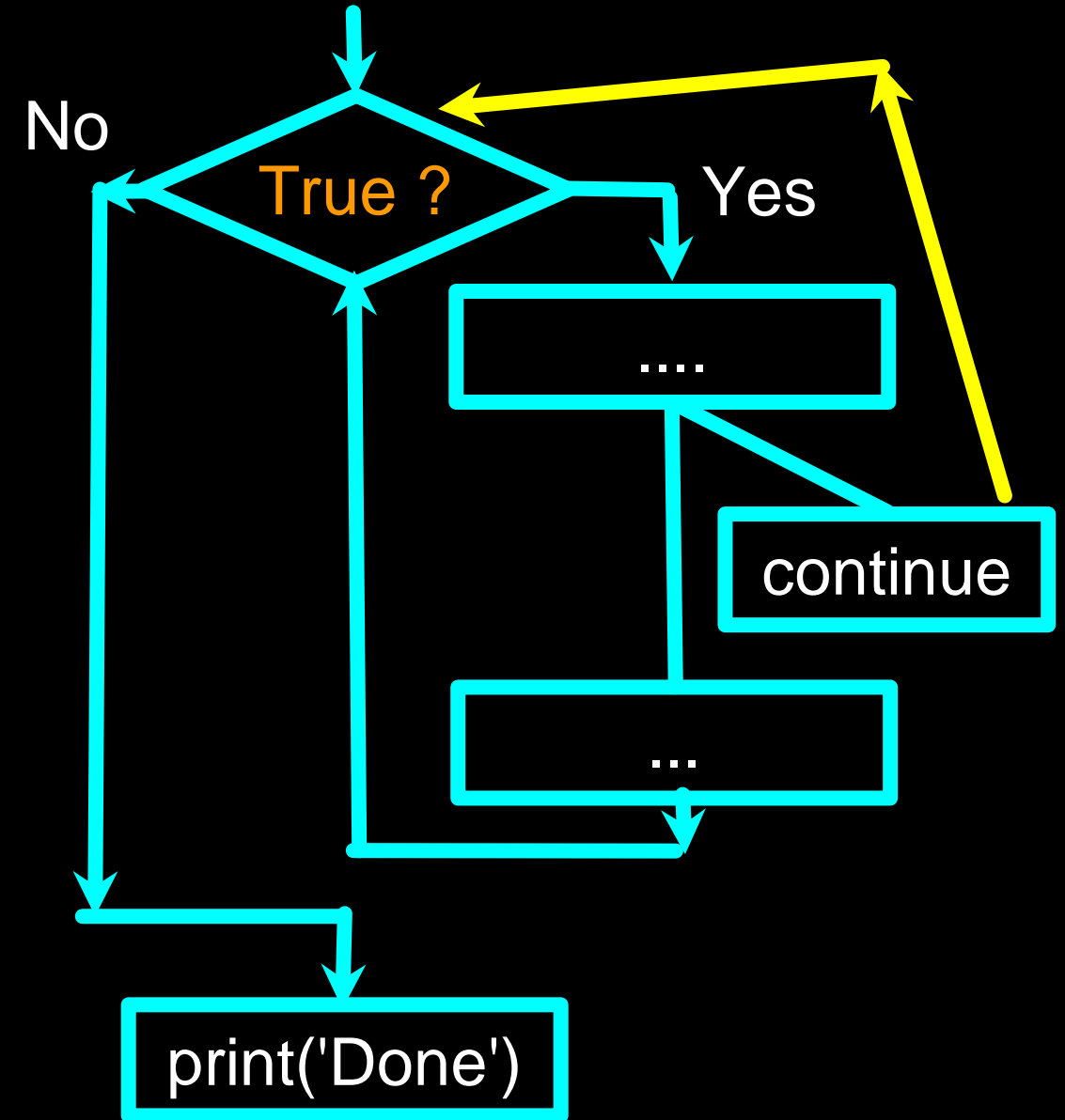
The **continue** statement ends the **current iteration** and jumps to the **top of the loop** and starts the next iteration

```
while True:
    line = input('> ')
    if line[0] == '#' :
        continue
    if line == 'done' :
        break
    print(line)
print('Done!')
```



```
> hello there
hello there
> # don't print this
> print this!
print this!
> done
Done!
```

```
while True:
    line = raw_input('> ')
    if line[0] == '#' :
        continue
    if line == 'done' :
        break
    print(line)
print('Done!')
```



# Vòng lặp for

```
for variable_1, variable_2, .. variable_n in sequence:  
    # for-block
```

- Dùng hàm **range(a, b)** để tạo danh sách gồm các số từ **a** đến **b-1**, hoặc tổng quát hơn là **range(a, b, c)** trong đó **c** là bước nhảy

```
for d in range(10,20):           # in các số từ 10 đến 19  
    print(d)  
for d in range(20,10,-1):       # in các số từ 20 đến 11  
    print(d)
```

# Vòng lặp for

- Ví dụ :

```
x=0  
#Define a for loop  
for x in range(2,7):  
    print(x)
```



# Vòng lặp for

- Ví dụ : Cách sử dụng vòng lặp for cho chuỗi

```
Months = ["Jan", "Feb", "Mar", "April", "May", "June"]  
for m in Months:  
    print(m)
```

# Lệnh break

- Lệnh break: cho phép bạn ngắt hoặc chấm dứt việc thực hiện vòng lặp.

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

Khi break được thực thi thì "#code khác bên trong vòng lặp for" sẽ bị bỏ qua và chuyển đến "#code bên ngoài vòng lặp for".

# Lệnh break

```
while dieu_kien_kiem_tra:

    #code bên trong vòng lặp while

    if dieu_kien:

        break

    #code khác bên trong vòng lặp while

#code bên ngoài vòng lặp while
```

# Lệnh break

```
# use the break and continue statements
for x in range (10,20):
    if (x == 15): break
    print(x)
```

```
for val in "string":
    if val == "i":
        break
    print(val)

print("Kêf thúc!")
```

# Lệnh continue

- Lệnh continue sẽ khiến việc lặp không thực thi ở vị trí lặp hiện tại, NHƯNG sẽ tiếp tục thực hiện các vòng lặp sau đó.

```
for var in sequence:

    #khôí code bên trong vòng lặp for

    if dieu_kien:

        continue

    #code khác bên trong vòng lặp for

#code bên ngoài vòng lặp for
```

Khi continue được thực thi thì "#code khác bên trong vòng lặp for" bị bỏ qua và quay trở lại "#Khôí code bên trong vòng lặp for"

# Lệnh continue

```
while dieu_kien_kiem_tra:

    #code bên trong vòng lặp while

    if dieu_kien:

        continue

    #code khác bên trong vòng lặp while

#code bên ngoài vòng lặp while
```

# Lệnh continue

```
for x in range (10,20):  
    if (x % 5 == 0) : continue  
    print(x)
```

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

# Hàm enumerate

- Cách sử dụng hàm "enumerate" cho "vòng lặp For"
- Hàm enumerate trong “vòng lặp for” thực hiện hai điều:
  - Nó trả về giá trị chỉ số cho các phần tử
  - Và phần tử trong tập đang được xét.

```
#use a for loop over a collection  
Months = ["Jan", "Feb", "Mar", "April", "May", "June"]  
for i, m in enumerate (Months):  
    print(i,m)
```

Khi chương trình được thực thi, đầu ra của hàm enumerate trả về tên tháng với chỉ số của tháng trong danh sách như (0-Jan), (1- Feb), (2 – Mar), v.v.



# Function

- Cú pháp khai báo hàm

```
def <tên-hàm>(danh-sách-tham-số):  
    <lệnh 1>  
    ...  
    <lệnh n>
```

```
def tinh(a, b):  
    return a*b
```

- Ví dụ: hàm tính tích 2 số

```
def tinh(a, b):  
    return a*b
```

Hàm trả về kết quả bằng lệnh **return**, nếu không trả về thì coi như trả về **None**

# Function

- Hàm có thể chỉ ra giá trị mặc định của tham số
  - Chú ý: các tham số có giá trị mặc định phải đứng cuối danh sách tham số

```
def tich(a, b = 1):  
    return a*b
```

```
print(tich(10, 20))    # 200  
print(tich(10))        # 10  
print(tich(a=5))       # 5  
print(tich(b=6, a=5))  # 30
```

# Thực Hành

1. Viết chương trình nhập số  $A$  và kiểm tra xem  $A$  có phải là số nguyên tố hay không?
2. Viết chương trình in ra tất cả số chẵn trong khoảng  $(M, N)$  .  $N, M$  nhập từ bàn phím.