CHƯƠNG 3
HÀM TRONG PYTHON

MỞ BÀI

- Yêu cầu: Xây dựng chương trình thực hiện tính tổ hợp chập k của n

$$C_n^k = \frac{n!}{k!\,(n-k)!}$$

- Thuật toán:
  - Bước 1: Nhập n, k từ bàn phím thỏa mãn điều kiện n>=k>0
  - Bước 2: Tính n!
  - Bước 3: Tính k!
  - Bước 4: Tính (n-k)!
  - Bước 5: Tính C_n^k theo công thức đã cho
  - Bước 6: Kết thúc

MỞ ĐẦU

- **Nhận xét:**

  Bước 2, 3, 4 cần tính giai thừa cho 3 số (n, k, n-k) cần viết code 3 lần.

- Câu hỏi đặt ra:
  - Có thể viết 1 đoạn mã lệnh tính giai thừa của 1 số bất kỳ và sử dụng nhiều lần không?

→ Tức đặt đoạn mã lệnh tính giai thừa của 1 số vào một khối riêng, mỗi khi cần tính giai thừa của 1 số n nào đó, ta chỉ cần gọi khối lệnh và truyền cho nó tham số n.

→ **Sử dụng hàm trong Python**

NỘI DUNG CHƯƠNG

1. Khái niệm về hàm
2. Truyền tham số cho hàm
3. Các thư viện thường dùng

3.1. KHÁI NIỆM VỀ HÀM

3.1.1. Giới thiệu về hàm

3.1.2. Cấu trúc tổng quát của hàm

3.1.3. Nguyên tắc hoạt động của hàm

3.1.4. Sử dụng giá trị trả về cho hàm

3.1.1. GIỚI THIỆU VỀ HÀM

- **Khái niệm**: là một khối lệnh/ chương trình con nhằm thực hiện 1 chức năng cụ thể nào đó.
- **Mục đích & Vai trò:**
  - ✓ Phân rã chương trình lớn thành các chương trình nhỏ hơn để dễ đọc, dễ sửa chữa, dễ nâng cấp.
  - ✓ Chương trình logic, gọn gàng hơn
  - ✓ Linh động sử dụng lại hàm qua lời gọi hàm giúp nhà lập trình tiết kiệm dòng lệnh và thời gian.
  - ✓ Phân chia công việc và ghép kết quả lại → tăng tốc giải bài toán.

3.1.1. GIỚI THIỆU VỀ HÀM

- **Đặc điểm của hàm trong Python**
  - ✓ Cho phép một chương trình có thể có nhiều hàm.
  - ✓ Cho phép trong một hàm định nghĩa một hàm khác (dựa theo ngôn ngữ Script) còn các ngôn ngữ lập trình khác không cho phép.
- **Các yếu tố của hàm**
  - ✓ Định nghĩa hàm (declaration)
  - ✓ Lời gọi hàm (call).

3.1.2. CẤU TRÚC TỔNG QUÁT CỦA HÀM

- **Định nghĩa hàm**

Cú pháp:

```
def  tên_hàm([danh sách tham số]):
    # {Thân hàm}
    [return giá_trị]
```

- **Lưu ý**: tham số là biến được liệt kê trong dấu () khi định nghĩa hàm.

3.1.2. CẤU TRÚC TỔNG QUÁT CỦA HÀM

- **Định nghĩa hàm**
- **Ví dụ:** In lời chào hello

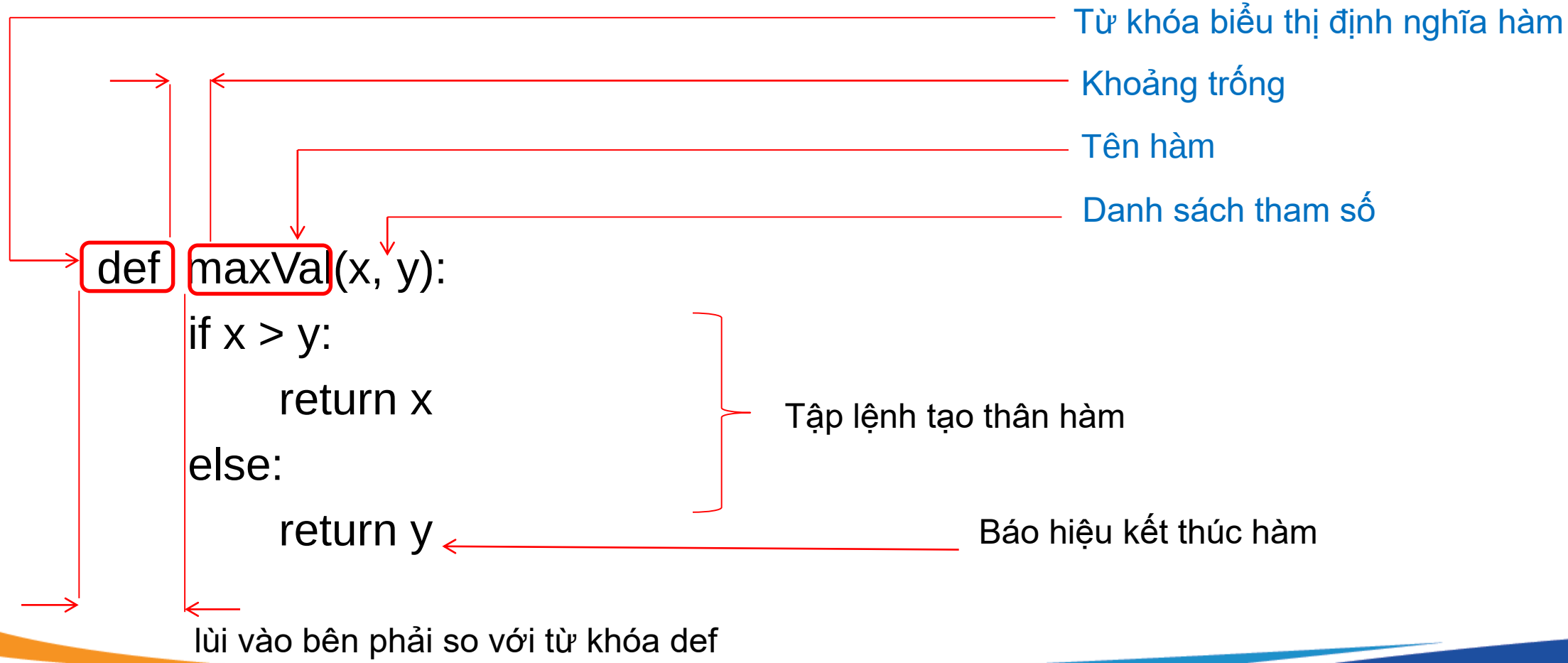
```
#Định nghĩa hàm
def print_hello():
    print("Hello ")
#Lời gọi hàm
print_hello()
```

3.1.2. CẤU TRÚC TỔNG QUÁT CỦA HÀM

- **Giải thích cú pháp định nghĩa hàm**:
  - **<tên_hàm>:** do người dùng đặt theo quy tắc đặt tên, nên gợi nhớ đến nhiệm vụ hàm.
  - **[ds tham số]:** Danh sách các tham số truyền vào làm giá trị đầu vào giúp hàm giải quyết công việc, [dsts] có thể có hoặc không, nếu nhiều thì các tham số cách nhau bởi dấu phẩy.
  - **<khối lệnh>:** gồm hữu hạn các lệnh trong thân hàm thực hiện một công việc cụ thể, được viết lùi sang phải cùng một khoảng trống so với từ khóa def
  - **<giá trị trả về>:** là giá trị trả về của hàm sau khi hoàn thành công việc nếu có lệnh return.

3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

- **Ví dụ hàm**: Cho trước 2 số khác nhau, hãy tìm số lớn nhất.



3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

- Yêu cầu: Xác định các thành phần trong hàm sau

Từ khóa

Tên hàm

Tham số

Mô tả (docstring)

```
def so_chan ( i ):
    """
    Dau vao: i la so nguyen duong
    Tra ve True neu i la so chan, nguoc lai la False
    """
    print("Bên trong hàm")
    return i%2==0
print(so_chan(3))
```

Thân hàm

Lời gọi hàm có truyền đối số

Tham số truyền vào hàm

3.1.2. CẤU TRÚC TỔNG QUÁT CỦA HÀM

- **Bài tập1: hiển thị HCN**

```
***************
*             *
*             *
***************
```

```
def draw_square():
    print('*' * 15)
    print('*', ' '*11, '*')
    print('*', ' '*11, '*')
    print('*' * 15)
draw_square()
```

- **Bài tập 2: Truyền đối số tên cho**

```
Hello Anna
Hello David
Hello Elsa
```

```
def print_hello(name):
    print('Hello ' + name)

print_hello('Anna')
print_hello('David')
print_hello('Elsa')
```

3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

Bài tập: Viết hàm tìm ước chung lớn nhất của 2 số nguyên a, b.

```
a=int(input("Nhap giá trị a= "))
b=int(input("Nhap gia tri b= "))
def uscln(a, b):
    while(a!=b):
        if (a <= b):
            b-=a
        else:
            a-=b
    return a
print("UCLN của",a, "và", b, "= ",uscln(a,b))
```

```
Nhap giá trị a= 27
Nhap gia tri b= 18
UCLN của 27 và 18 =  9
```

3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

- **docstring** (document string): là một chuỗi được đặt ở đầu 1 hàm có nhiệm vụ giải thích ý nghĩa của hàm. Docstring là 1 tùy chọn nhưng NLT nên sd.
- **docstring** là một chuỗi đặt trong ba dấu nháy “““…””””. Chuỗi này được phép nằm trên nhiều dòng liên tiếp.
- Chuỗi docstring có tác dụng:
  - Giải thích 1 cách cô đọng hàm có nhiệm vụ gì
  - Giải thích ảnh hưởng của mỗi tham biến đối với biểu hiện của hàm và mỗi tham biến có kiểu là gì.
- Ví dụ:

```
def polyline(t, length, n, angle):
    """Vẽ n đoạn thẳng với chiều dài cho trước và góc
    (tính bằng độ) giữa chúng.  t là một Turtle.
    """
    for i in range(n):
        fd(t, length)
        lt(t, angle)
```

3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

Các loại tham số của hàm:

- Tham số với giá trị bắt buộc
- Tham số với giá trị mặc định
- Tham số từ khóa
- Tham số tùy ý

3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

- **Tham số với giá trị bắt buộc**: là tham số yêu cầu NLT buộc phải truyền các đối số vào trong lời gọi hàm theo đúng thứ tự, nếu không chương trình dịch sẽ thông báo lỗi.
- **Ví dụ**: Hoán vị hai số

```
def hoanvi(a,b):
    return b,a
so1=int(input("Nhap so thu nhat = "))
so2=float(input("Nhap so thu hai = "))
so1,so2=hoanvi(so1,so2)
print "So thu nhat= ",so1)
print "So thu hai = ",so2)
so1,so2=hoanvi()
```

Gọi hàm: Máy báo lỗi do không truyền đối số bắt buộc

3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

- **Tham số với giá trị mặc định/ngầm định**: các tham số này được NLT thiết lập giá trị ngay khi xây dựng hàm. Điều này nhằm tránh gây lỗi cho người dùng không truyền đối số vào khi gọi hàm.
- **Cú pháp**:

```
def <tên hàm> ([<th.số1>=<g.trị1>, <th.số_2> = <g.trị2>, …]):
        <khối lệnh>
[return <g.trị trả về>]
```

3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

- **Tham số với giá trị mặc định/ngầm định (tt)**:
  - **Ví dụ**: Xây dựng hàm tính diện tích hình tròn với tham số đầu vào là bán kính đường tròn r và pi, r đặt mặc định bằng 0 và pi đặt mặc định bằng 3.14
  - **Thực hiện:**

```
def DT_hinhtron(r=0,pi=3.14):
    dt: float = pi * r *r
    return dt
print("Dien tích hình tròn bán kính mặc định = ", DT_hinhtron())
print("Dien tích hình tròn bán kính r=3 là: ",DT_hinhtron(3))
```

```
Dien tích hình tròn bán kính mặc định =  0.0
Dien tích hình tròn bán kính r=3 là:  28.259999999999998
```

- **Ghi chú**: Python quy định các tham số mặc định phải đứng ở vị trí sau cùng trong danh sách các tham số khi định nghĩa một hàm.

3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

- **Nhận xét:** Với các kiểu tham số ở mục trước khi gọi hàm, chúng ta cần dựa vào vị trí của tham số trong phần định nghĩa hàm để truyền đúng thứ tự.
  → Python cung cấp kiểu tham số từ khóa để khắc phục nhược điểm này.
- **Tham số từ khóa**: Trong **lời gọi hàm** Python cho phép NLT không cần nhớ chính xác vị trí của các tham số mà chỉ cần nhớ **tên tham số**.

Lời gọi hàm:

[<tên biến>=] <tên hàm>([<Tsố1>=<g.trị1>, <Tsố2> = <g.trị2>, …])

- **Ví dụ**:

```
def  dientich(dai,rong):        # Định nghĩa hàm S_HCN
    pass
dientich(rong=10, dai=20)       # Lời gọi hàm
```

3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

- **Tham số từ khóa**:
  - **Chú ý**: Nếu trong hàm có chứa nhiều tham số, mà NLT muốn truyền tham số theo mặc định và truyền tham số theo từ khóa thì các tham số truyền theo mặc định phải đặt ở phía cuối của danh sách tham số khi xây dựng hàm
  - **Ví dụ:**

```
def  tong(a,b,c=5,d=3):   #định nghĩa hàm
tong(b=10,a=7)            #lời gọi hàm
```

```
    def tong(a,c=10,d=5,b):
                           ^
SyntaxError: non-default argument follows default argument
```

3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

- **Tham số tùy ý**: là số lượng đối số truyền vào trong lời gọi hàm không bị giới hạn
- **Cú pháp**: Đặt dấu * trước tên tham số biểu thị kiểu tham số này.
- **Ví dụ**:

```
def tong(*args):
    sum=0
    for i in args:
        sum=sum+i
    return sum
print("tong=", tong(10,20,20))
```

```
"C:\Users\Public\BT Python\venv\Scripts\
tong= 50

Process finished with exit code 0
```

3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

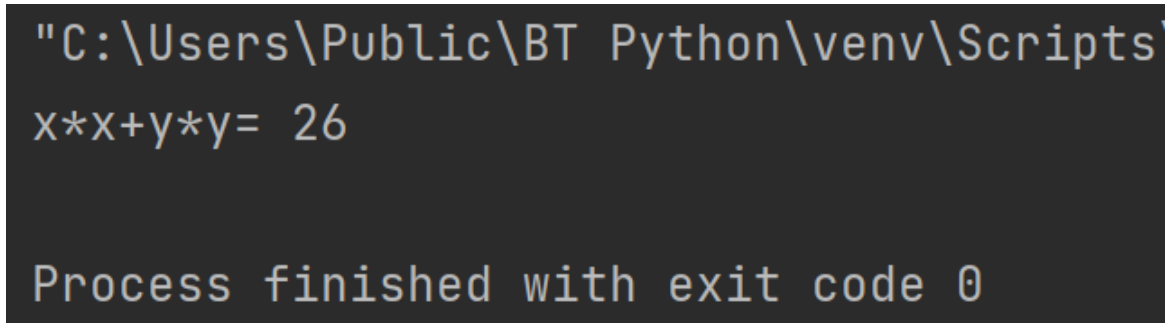
- **Hàm định nghĩa trực tiếp:**
  - **Trong Python** cung cấp cho người dùng 2 cách định nghĩa hàm:
    - Cách 1: Định nghĩa thông qua từ khóa def
    - Cách 2: Định nghĩa hàm vô danh qua lambda
  - **Đặc điểm** hàm lambda:
    - Bắt buộc phải trả về giá trị bằng một biểu thức tính toán.
    - Có thể có một hoặc nhiều tham số
  - **Cú pháp**:

Ten_ham=lambda<tham_số>:<biểu thức tính>

3.1.2. CẤU TRÚC TỔNG QUÁT HÀM

- **Hàm định nghĩa trực tiếp:**
  - **Ví dụ:** Hàm d() được định nghĩa trực tiếp bằng công thức $d(x,y) = x^2+y^2$.
  - **Thực hiện:**

```
d = lambda x,y: x*x+y*y      # định nghĩa trực tiếp hàm d()
print("x*x+y*y=", d(1,5))    # Lời gọi hàm với đối số 1, 3
```



3.1.3 NGUYÊN TẮC HOẠT ĐỘNG CỦA HÀM

- Hàm sẽ không chạy cho đến khi được gọi
- **Lời gọi hàm**: áp dụng hàm với bộ dữ liệu cụ thể (truyền đối số - Arguments)
- **Cú pháp** gọi hàm:

[<tên biến> =] <tên hàm> ([<dsđs>])

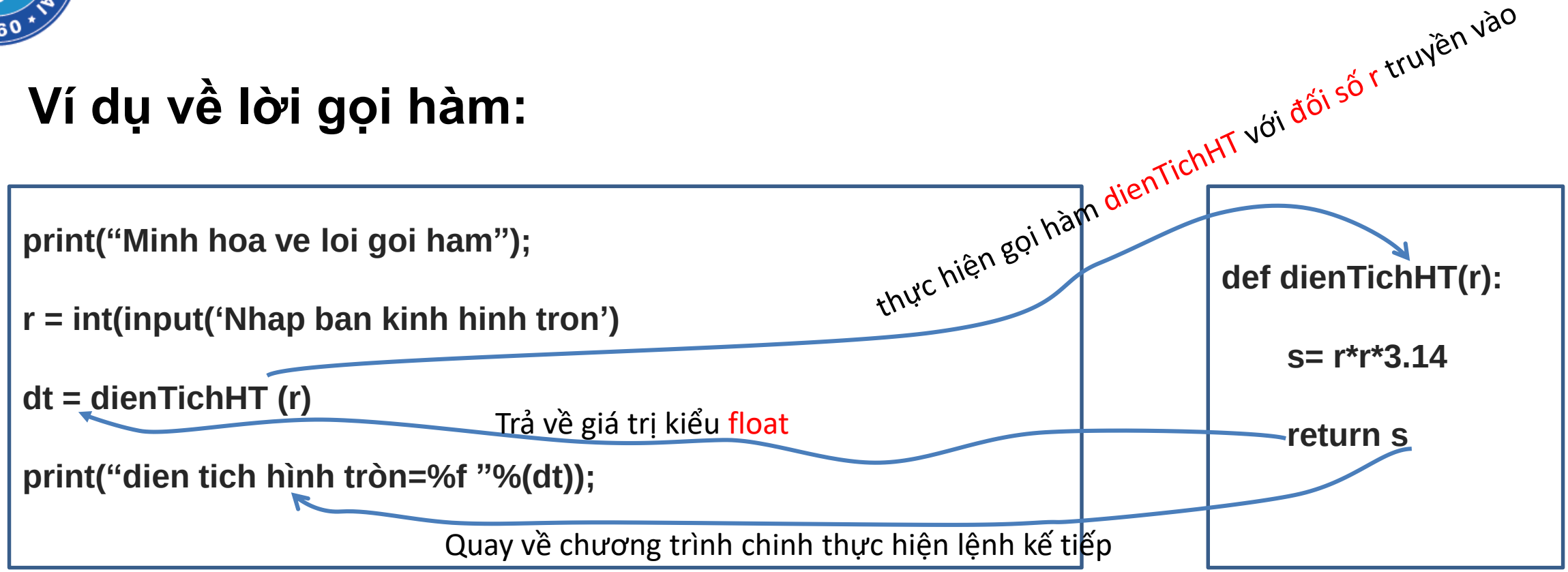
- **Trong đó:**
  - <tên_biến>: Biến được dùng để lưu trữ dữ liệu sau khi kết thúc hàm
  - <dsđs>: ds đối số truyền vào làm giá trị đầu vào cho hàm, <dsđs> có thể có hoặc không, tùy vào hàm xây dựng, nếu có nhiều thì các đối số cách nhau bởi dấu phẩy dưới.
- **Ví dụ**: uscln(9,6)

3.1.3. NGUYÊN TẮC HOẠT ĐỘNG CỦA HÀM

- **Quá trình thực hiện khi gặp lời gọi hàm**
  - Python tạo một namespace riêng để điều khiển các biến nhớ bên trong hàm và namespace này độc lập với namespace tại nơi hàm này được gọi → những gì xảy ra bên trong hàm không ảnh hưởng tới bên ngoài.
  - Gán giá trị của đối số thực cho các tham số tương ứng
  - Thực hiện các câu lệnh trong thân hàm
  - Khi gặp câu lệnh return hoặc câu lệnh cuối cùng của thân hàm thì máy sẽ xóa các đối, các biến cục bộ và rời khỏi hàm

3.1.3. NGUYÊN TẮC HOẠT ĐỘNG CỦA HÀM

- **Ví dụ về lời gọi hàm:**



3.1.3. NGUYÊN TẮC HOẠT ĐỘNG CỦA HÀM

- **Video minh họa lời gọi hàm**

Write code in Python 3.6

Why are there ads?

```
1
2  def dienTichHT(r):
3      s= r*r*3.14
4      return s
5
6
7  print("Minh hoa ve loi goi ham")
8  r = int(input('Nhap ban kinh hinh tron r= '))
9  dt = dienTichHT(r)
10 print("Dien tich hình tròn=%f "%(dt))
```

Visualize Execution NEW: subscribe to our YouTube for weekly videos

3.1.4. SỬ DỤNG GIÁ TRỊ TRẢ VỀ CỦA HÀM

- ***Ý nghĩa***: Giá trị trả về của hàm là một giá trị cụ thể có kiểu phụ thuộc vào biểu thức trả về trong câu lệnh return.
- ***Vai trò***: Có thể sử dụng như một giá trị hằng trong phép gán, các phép toán số học, các câu lệnh nhập xuất.
- ***Ví dụ: Tính tổ hợp chập k của n*** = $C_n^k = \frac{n!}{k!(n-k)!}$

```
def giaithua(a):
    g=1
    for i in range (1,a+1):
        g=g*i
    return g
```

3.1.4. SỬ DỤNG GIÁ TRỊ TRẢ VỀ CỦA HÀM

- *Ví dụ: Tính tổ hợp chập k của n =* $C_n^k = \frac{n!}{k!(n-k)!}$

```
def giaithua(a):
    g=1
    for i in range (1,a+1):
        g=g*i
    return g
```

```
Giai thua cua n=  6
Giai thua cua k=  2
To hop chap k cua n =  3.0
```

- Sử dụng:

```
k=2;n=3
print("Giai thua cua n= ",giaithua(n))
print("Giai thua cua k= ",giaithua(k))
C=giaithua(n)/(giaithua(k)*giaithua(n-k))
print('To hop chap k cua n = ',C)
```

3.1.4. SỬ DỤNG GIÁ TRỊ TRẢ VỀ CỦA HÀM

- ***Hàm trả về nhiều giá trị:***
  - ***Sử dụng dấu "," để phân cách các giá trị***
- ***Ví dụ:***

```
def tim_max_min(a):
    max =a[0]
    min = a[0]
    for i in range (1,len(a)):
        if max< a[i]:
            max = a[i]
        if min > a[i]:
            min = a[i]
    return max, min
```

3.1.4. SỬ DỤNG GIÁ TRỊ TRẢ VỀ CỦA HÀM

- *Hàm trả về nhiều giá trị:*
  - *Sử dụng dấu “,” để phân cách các giá trị*
- *Ví dụ: Các giá trị có kiểu khác nhau*

```
def demo_func():
    return 'hello', 100
# lời gọi hàm
message,time = demo_func()
print(message)
print(time)
```

3.1.4. SỬ DỤNG GIÁ TRỊ TRẢ VỀ CỦA HÀM

- *Hàm trả về nhiều giá trị:*
  - *Sử dụng danh sách (list)*
- *Ví dụ: Các giá trị có kiểu khác nhau*

```
def test_func():
    return ['hello', 100]

# lời gọi hàm
result = test_func()
print (result[0])
print (result[1])
```

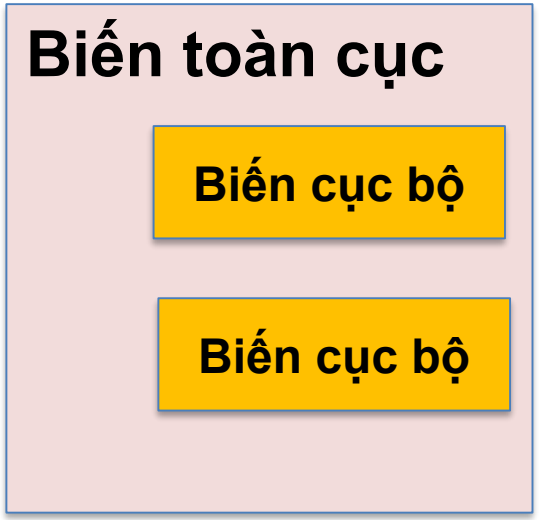
3.2.TRUYỀN THAM SỐ CHO HÀM

3.2.1. Phạm vi và vòng đời của biến

3.2.2. Tham số và hình thức truyền tham số cho hàm

3.2.1. PHẠM VI VÀ VÒNG ĐỜI CỦA BIẾN

- ***Biến có 2 loại:*** *Biến toàn cục và biến cục bộ*
- ***Ý nghĩa***: loại biến sẽ quyết định vòng đời và phạm vi hoạt động của nó.
  - ***Biến toàn cục***: là biến có phạm vi hoạt động toàn chương trình, trong nhiều hàm
    - ✓ ***Vị trí***: được khai báo ngoài mọi hàm.
    - ✓ ***Vòng đời***: từ vị trí của nó được khai báo cho tới hết chương trình.
  - **Biến cục bộ**: và có phạm vi hoạt động trong 1 hàm.
    - ✓ ***Vị trí***: được khai báo trong 1 hàm cụ thể.
    - ✓ ***Vòng đời***: từ vị trí của nó được khai báo cho tới hết hàm.



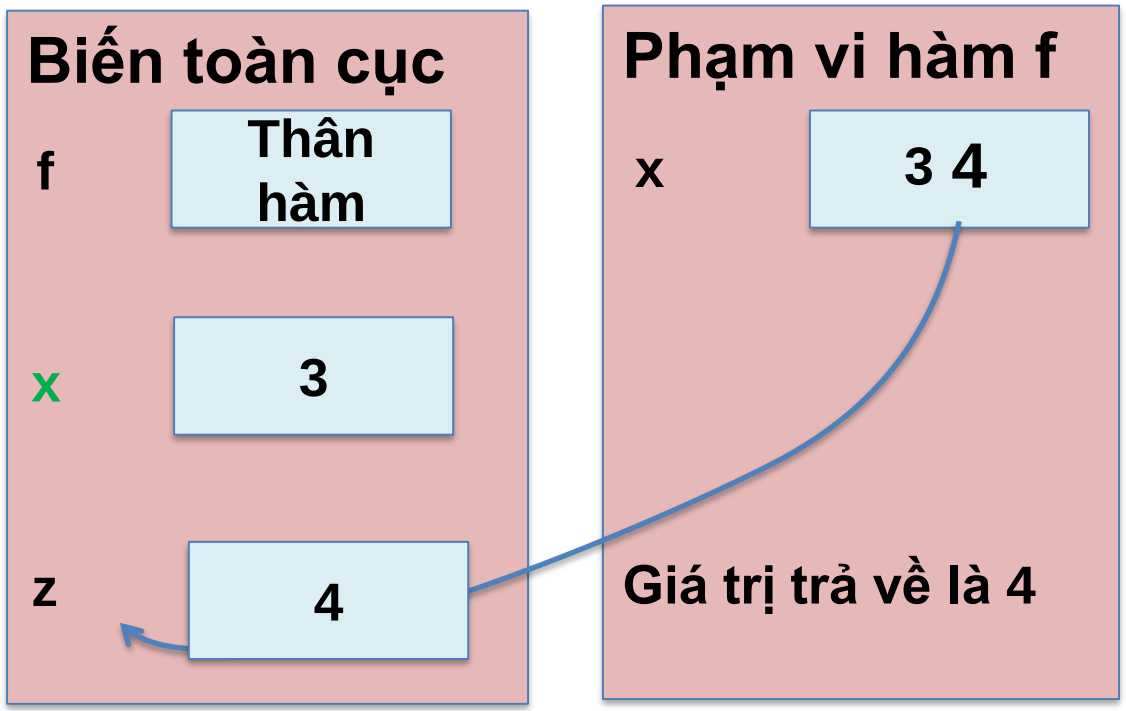
3.2.1. PHẠM VI VÀ VÒNG ĐỜI CỦA BIẾN

- Ví dụ về biến cục bộ và biến toàn cục

```
def f(x):
    x = x+1
    print('Trong ham f(x): x= ',x)
    return x

x = 3
z = f(x)
```

→ x: biến toàn cục
→ x: đối số
→ x: tham số
→ x: biến cục bộ



3.2.1. PHẠM VI VÀ VÒNG ĐỜI CỦA BIẾN

- **Lưu ý:**
  - ✓ Đối số và biến cục bộ của hàm nên có tên khác nhau
  - ✓ Hàm tự xác định kiểu dữ liệu của các đối số để thực hiện các thao tác của hàm phù hợp với kiểu dữ liệu tương ứng
  - ✓ Không thể mang giá trị của đối số ra khỏi hàm
  - ✓ Không thể dùng đối số để thay đổi giá trị của các đại lượng bên ngoài hàm.

```
#Định nghĩa hàm
def print_hello(n):
  print("Hello "*n)
#Lời gọi hàm
print_hello(2)
times=5
print_hello(times)
```

```
def my_function(food):
  for x in food:
    print(x)
fruits =
["apple", "banana", "cherry"]

my_function(fruits)
```

3.2.1. PHẠM VI VÀ VÒNG ĐỜI CỦA BIẾN

- Nếu hàm có biến cục bộ trùng tên với biến toàn cục thì biến cục bộ luôn được ưu tiên.
- Xét ví dụ:

```
n=10                          # biến toàn cục
def func1():
    for i in range(1,n):      # biến toàn cục n=10
        print(i,end=' ')
i=6
def thu(n):
        i=2*n     # Biến toàn cục i trùng với biến cục bộ i
        print()
        print(i) # Hiển thị giá trị biến cục bộ i
x=2
func1()
thu(x)
print(i)          # hiển thị giá trị biến toàn cục i
```

```
1 2 3 4 5 6 7 8 9
4
6
```

3.2.1. PHẠM VI VÀ VÒNG ĐỜI CỦA BIẾN

- Trường hợp hàm muốn làm thay đổi giá trị biến toàn cục thì sử dụng mệnh đề global trước tên biến, còn nếu chỉ sử dụng mà không thay đổi giá trị thì không cần.
- Xét ví dụ:

```
def reset():
    global time_left
    time_left = 0
def print_time():
    print(time_left)
time_left=30
reset()
print_time()
```

```
"C:\Users\Public\BT Python\venv\Scripts\
time_left =  0

Process finished with exit code 0
```

3.2.1. PHẠM VI VÀ VÒNG ĐỜI CỦA BIẾN

- Nếu không sd global thì Python sẽ báo lỗi khi thay đổi giá trị của biến toàn cục
- Ví dụ:

```
# Bên trong hàm thamso() không thể truy cập được biến a, b
a,b = 5,6
def thamso():
    a = a-1
    print("a=5 b=6 : \n \t\t a*b= ",a*b)
thamso()
print("\n a=5 b=6 : \n \t\t a*b= ",a*b)
```

→ Kết quả: Báo lỗi

```
Traceback (most recent call last):
  File "C:\Users\", line 14, in <module>
    thamso()
  File "C:\Users\", line 12, in thamso
    a = a-1
UnboundLocalError: local variable 'a' referenced before assignment
```

3.2.2. THAM SỐ VÀ HÌNH THỨC TRUYỀN THAM SỐ CHO HÀM

- Tham số trong Python có hai loại:
  - ✓ Tham số hình thức
  - ✓ Tham số thực sự (đối số)
- Tham số hình thức (Parameter): là tham số viết trong định nghĩa hàm
- Tham số thực sự - đối số (Argument) là tham số viết trong lời gọi hàm.
- Tham số thực sự cần cùng kiểu và bằng số lượng tham số hình thức (trừ một số trường hợp đặc biệt: tham số mặc định)

3.2.2. THAM SỐ VÀ HÌNH THỨC TRUYỀN THAM SỐ CHO HÀM

- Tham số hình thức (Parameter): là tham số viết trong định nghĩa hàm
- Cú pháp:

```
def equation (a, b, c):
    from math import sqrt
    d = b * b - 4 * a * c
    if d >= 0:
        x1 = (- b + sqrt(d)) / (2 * a)
        x2 = (- b - sqrt(d)) / (2 * a)
        return (x1, x2)
```

3.2.2. THAM SỐ VÀ HÌNH THỨC TRUYỀN THAM SỐ CHO HÀM

- Tham số hình thức (Parameter): là tham số viết trong định nghĩa hàm
- Cú pháp: tham số hình thức kết hợp với kiểu

```
def sum_range(start: int, stop: int, step: int = 1) -> int:
    sum = 0
    for i in range(start, stop, step):
        sum += i
    return sum
# lời gọi
sum_range(1, 100)
```

3.2.2. THAM SỐ VÀ HÌNH THỨC TRUYỀN THAM SỐ CHO HÀM

- Tham số hình thức (Parameter): là tham số viết trong định nghĩa hàm
- Cú pháp: tham số ngầm định

```
def equation (a =1 , b= 2, c= 3):
    from math import sqrt
    d = b * b - 4 * a * c
    if d >= 0:
        x1 = (- b + sqrt(d)) / (2 * a)
        x2 = (- b - sqrt(d)) / (2 * a)
        return (x1, x2)
```

3.2.2. THAM SỐ VÀ HÌNH THỨC TRUYỀN THAM SỐ CHO HÀM

Ví dụ: Hoán vị

Ds tham số

```
a=5
b=7
def hoanvi(a,b):
    tam=a
    a=b
    b=tam
    print("Trong ham: a= %d b= %d" %(a,b))
print("Truoc loi goi ham a= {} b= {}".format(a, b))
print("Goi ham: hoan vi")
hoanvi(a,b)
print("Ngoai ham a= {} b= {}".format(a,b))
```

Biến cục bộ

Ds đối số thực

Kết quả chạy

```
Truoc loi goi ham a= 5 b= 7
Goi ham: hoan vi
Trong ham: a= 7 b= 5
Ngoai ham a= 5 b= 7
```

3.2.2. THAM SỐ VÀ HÌNH THỨC TRUYỀN THAM SỐ CHO HÀM

Khi có tham số ngầm định

```
def equation (a =1 , b= 2, c= 3):
    from math import sqrt
    d = b * b - 4 * a * c
    if d >= 0:
        x1 = (- b + sqrt(d)) / (2 * a)
        x2 = (- b - sqrt(d)) / (2 * a)
        return (x1, x2)
```

```
# lời gọi với tham số định vị
(x1, x2)= equation(2, 5, 2)
print('nghiệm', x1, x2)
# lời gọi với tên tham số
(x1, x2)= equation(a =2, b=5, c=2)
print('nghiệm', x1, x2)
# lời gọi với tham số ngầm định
(x1, x2) = equation(a =2, b=5)
# c ngầm định bằng 0
print('nghiệm', x1, x2)
```

3.2.2. THAM SỐ VÀ HÌNH THỨC TRUYỀN THAM SỐ CHO HÀM

- **Truyền tham số:**
  - Truyền giá trị.
  - Truyền biến.

```
def equation (a =1 , b= 2, c= 3):
    from math import sqrt
    d = b * b - 4 * a * c
    if d >= 0:
        x1 = (- b + sqrt(d)) / (2 * a)
        x2 = (- b - sqrt(d)) / (2 * a)
        return (x1, x2)
```

```
# truyền giá trị
(x1, x2)= equation(2, 5, 2)
print('nghiệm', x1, x2)
# truyền biến
a= 2
b= 5
c=2
(x1, x2)= equation(a,b,c)
print('nghiệm', x1, x2)
```

3.2.2. THAM SỐ VÀ HÌNH THỨC TRUYỀN THAM SỐ CHO HÀM

- **Ví dụ:**

```
def hello(name, msg):
"""Say hello to a person with message"""
    print("Hello ", name, ", ", msg, sep='')
```

- **Truyền tham số:**

```
# truyền theo tên tham số hình thức
hello(name = "John", msg = "How are you?")
# có thể đảo thứ tự trong truyền tham số với tên tham số hình thức
hello(msg = "How are you?", name = "John")
# 1 tham số truyền theo vị trí, 1 tham số truyền theo tên
hello("John", msg = "How are you?")
```

3.2.2. THAM SỐ VÀ HÌNH THỨC TRUYỀN THAM SỐ CHO HÀM

- **Ví dụ:**

```
def hello(name, msg):
"""Say hello to a person with message"""
    print("Hello ", name, ", ", msg, sep='')
```

- **Truyền tham số không đúng:** Khi gọi hàm mà sử dụng tên tham số trước thứ tự tham số sẽ gây ra lỗi SyntaxError

```
# lỗi truyền tham số
hello(name ="John", "How are you?")
```

3.2.2. THAM SỐ VÀ HÌNH THỨC TRUYỀN THAM SỐ CHO HÀM

- Python cho phép xây dựng hàm đệ quy (là hàm mà trong định nghĩa hàm có thể gọi đến chính nó

```
def func3(n):
    if (n==1):
        return 1
    else:
        return n*func3(n-
1)
n=5
print(func3(n))
```

3.2.2. THAM SỐ VÀ HÌNH THỨC TRUYỀN THAM SỐ CHO HÀM

- **Tham số biến động:**
  - Tham số có số thành phần biến động.

```
def sum(start, *numbers):
    for n in numbers:
        start += n
    return start

print(sum(0, 1, 2)  )# = 3
sum(1, 2, 3)  # = 6
sum(0, 1, 2, 3, 4, 5, 6)  # = 21
```

3.2.2. THAM SỐ VÀ HÌNH THỨC TRUYỀN THAM SỐ CHO HÀM

- Hàm chỉ thay đổi giá trị của các đối số mà bản thân nó có thể thay đổi như List hoặc Dictionary còn tuple, set thì không. Khi đó ta chuyển chúng thành chuyển thành List.
- **Ví dụ:**

```
def func1(x):
        x = x + 1
def func2(L):
        L.append(1)
a=3
M=[1,2,3]
func1(a)  #a không thay đổi
func2(M)
```

```
a = 3
M=  [1, 2, 3, 1]
```

3.3. Các thư viện thường dùng

- Có 3 loại thư viện thường dùng là:
  - ✓ Viết bằng Python: có phần mở rộng là .py
  - ✓ Các thư viện liên kết động: có phần mở rộng là .dll, .pyd , .so , .sl ,...
  - ✓ C-Module liên kết với trình phiên dịch.

3.3. Các thư viện thường dùng

- 3.3.1 Các hàm tích hợp sẵn (built-in function)
- 3.3.2 Thư viện toán học
- 3.3.3 Thư viện xử lý chuỗi
- 3.3.4 Cài đặt và sử dụng thư viện mở rộng

3.3.1 Các hàm tích hợp sẵn (built-in function)

- Trong Python có một số hàm được tích hợp sẵn, đó là các hàm có thể được sử dụng mà không cần import thư viện.
- Một số hàm dựng sẵn như:
  - **abs();** aiter(); all(); any(); anext(); ascii()
  - bin(); bool(); breakpoint(); bytearray(); bytes()
  - callable(); chr(); classmethod(); compile(); complex()
  - eval(); exec(); float(); hash(); **help();** id(); **input();** iter();
  - **len();** list(); locals(); map(); **max(); min(); next();**
  - **open(); pow(); print()**
  - **range();** round(); set(); **sum();** super(); tuple(); type(); zip()

3.3.1 Các hàm tích hợp sẵn (built-in function)

- Hàm vào ra dữ liệu: **input(); print()**
- **Hàm input():**
  - Sử dụng để nhận dữ liệu từ người dùng (nhập vào từ bàn phím)
- Hàm print():
  - Sử dụng để hiện thị dữ liệu
- Ví dụ:

```
n= input("nhập số : ")
print('n = ',n)

s= input("nhập chuỗi : ")
print('s = ',s)
```

3.3.1 Các hàm tích hợp sẵn (built-in function)

- Các hàm tính toán: **abs(); max(); min(); pow(); round(); sum();**

```
print(abs(-5))
print('min = ', min(2,4,5,9))
print('max = ',max(2, 4, 5, 9))
print(pow(2,3))
print(round(2.345,2))
print('sum = ', sum(2, 4, 5, 9))
```

```
5
min =  2
max =  9
8
2.35
sum =  20
```

3.3.1 Các hàm tích hợp sẵn (built-in function)

- Các hàm tính toán:
  - **len(); next();**
  - **range();**
- Ví dụ: cách thức duyệt và hiển thị một mảng

```
a = [2, 4, 5, 9]
for i in range(len(a)):
    print(" a[",i,"] = ", a[i])
```

```
a[ 0 ] =  2
a[ 1 ] =  4
a[ 2 ] =  5
a[ 3 ] =  9
```

3.3.2 Thư viện toán học

- Khi làm việc với các phép toán toán học, Python cung cấp thư viện math và cmath.
- Cú pháp

```
import math
```

- Một số hàm trong thư viện **math**:
  - fabs(x); factorial(n); floor(x); fmod(a,b); exp(x); exp2(x)
  - fsum();
  - Isfinite(x); isinf(x); isnan(x);
  - log(x); log2(x); pow(x,y)
  - sqrt(x)
  - ….

3.3.2 Thư viện toán học

- Ví dụ:

```
import math
print( math.exp(3))
print( math.sqrt(4))
print( math.log(10) )
print( math.pow(2,3))
```

```
20.085536923187668
2.0
2.302585092994046
8.0
```

3.3.3 Thư viện xử lý chuỗi

- Python cung cấp thư viện string.
- Cú pháp sử dụng

```
import string
```

- Một số hàm trong thư viện **string**:
  - capitalize( ); casefold(); lower(); upper();
  - count()
  - find();
  - format();
  - isdigit(); isnumeric();
  - Join(); removeprefix); removesuffix)
  - ….

3.3.3 Thư viện xử lý chuỗi

- Ví dụ: Khai báo chuỗi

```
# Các khai báo sau đều tương tự nhau
my_string = 'Hello'
print(my_string)
my_string = "Hello"
print(my_string)
my_string = '''Hello'''
print(my_string)

# sử dụng dấu '''__''' cho phép tạo chuỗi có nhiều dòng
my_string = """Hello, welcome to
           the world of Python"""
print(my_string)
```

3.3.3 Thư viện xử lý chuỗi

- Ví dụ: Truy cập các giá trị trong chuỗi

```
#khởi tạo
str = 'programiz'
print('str = ', str)
#Ký tự đầu tiên
print('str[0] = ', str[0])
#Ký tự cuối cùng
print('str[-1] = ', str[-1])
#cắt đoạn từ ký tự 1-5
print('str[1:5] = ', str[1:5])
#cắt đoạn từ ký tự thứ 5- cuối
print('str[5:-2] = ', str[5:-2])
```

```
str =  programiz
str[0] =  p
str[-1] =  z
str[1:5] =  rogr
str[5:-2] =  am
```

3.3.3 Thư viện xử lý chuỗi

- Ví dụ: Kết nối chuỗi

```
# Python String Operations
str1 = 'Hello'
str2 ='World!'
# using +
print('str1 + str2 = ', str1 + str2)
# format
str=str1 + str2
txt = str.center(20)
print(txt)
```

```
str1 + str2 =  HelloWorld!
HelloWorld!
    HelloWorld!
```

3.3.4 Cài đặt và sử dụng thư viện mở rộng

- Python cung cấp rất nhiều thư viện hữ ích
  - Numpy, Pandas, sk-learnt, Matplotlib, …
- Cú pháp sử dụng

```
import numpy as np
```

- Cần cài đặt vào môi trường lập trình Python
  - sử dụng lệnh pip/pip3

```
pip install numpy
```

3.3.4 Cài đặt và sử dụng thư viện mở rộng

- Ví dụ: tạo mảng với thư viện numpy

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
print(type(arr))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

3.3.4 Cài đặt và sử dụng thư viện mở rộng

- Ví dụ: tạo dataframe với thư viện pandas

```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

df = pd.DataFrame(data)
print(df)
```

	calories	duration
0	420	50
1	380	40
2	390	45

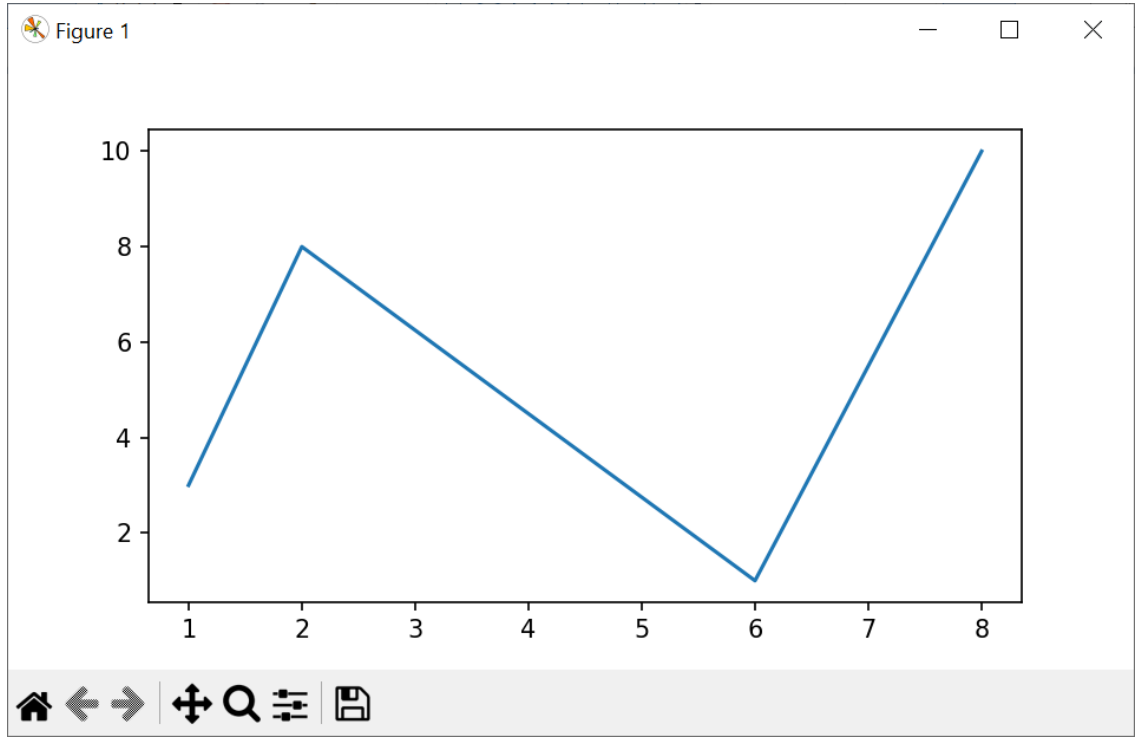
3.3.4 Cài đặt và sử dụng thư viện mở rộng

- Ví dụ: tạo biểu đồ với thư viện matplotlib

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
plt.show()
```



TỔNG KẾT CHƯƠNG 3

- Định nghĩa, cú pháp hàm trong Python
- Nguyên tắc hoạt động của hàm
- Tham số, đối số
- Giá trị trả về của hàm
- Biến toàn cục, biến cục bộ

GIẢI THÍCH THUẬT NGỮ - GLOSSARY

Thuật ngữ	Ý nghĩa
def	Từ khóa định nghĩa hàm
parameter	Tham số
argument	Đối số
return	Lệnh trả về
pass	Bỏ qua thân hàm
lambda	Hàm tự định nghĩa
docstring	Mô tả hàm

HẾT CHƯƠNG 3
CHÚC CÁC EM HỌC TỐT!!!