BÀI GIẢNG

CƠ SỞ LẬP TRÌNH

CHƯƠNG 4. HÀM

NGUYỄN THÀNH THỦY

BỘ MÔN TIN HỌC QUẢN LÝ

TRƯỜNG ĐẠI HỌC KINH TẾ, ĐẠI HỌC ĐÀ NẴNG

THUYNT@DUE.EDU.VN

NỘI DUNG

- Khái niệm về Hàm
- Khai báo Hàm
- Tham số của Hàm
- Nhận giá trị từ Hàm
- Lời gọi Hàm
- Phạm vi của biến
- Hàm rỗng
- Xử lý ngoại lệ

KHÁI NIỆM VỀ HÀM

- **Ví dụ 4.1:**

  Tính tổng của n số nguyên dương đầu tiên:

  - Nhập từ bàn phím một số nguyên **n**
  - Tính tổng **S** của **n** số nguyên đầu tiên

    S=1+2+…+n
  - In lên màn hình tổng **S**

```
Nhap mot so nguyen:
n=5
Tong cua 5 so nguyen duong dau tien=15
```

KHÁI NIỆM VỀ HÀM

- **Lời giải không sử dụng hàm**

Nhap(n)

```
#Nhap du lieu
print("Nhap mot so nguyen:")
n=int(input("n="))
```

Tinh(n)

```
#Tinh tong
S=0
for x in range(1,n+1):
    S=S+x
```

InKQ(S)

```
#In ket qua
print("Tong cua ",n," so nguyen duong dau tien=",S,sep="")
```

KHÁI NIỆM VỀ HÀM

❑ Phân tích bài toán để chuyển vào hàm

- Hàm **Nhap(n):**
  - **Input**: không có
  - **Output:** số nguyên **n**
  - **Process:** cho phép người dùng nhập từ bàn phím một số nguyên và lưu vào biến n;
- Hàm **Tinh(n):**
  - **Input**: số nguyên **n**
  - **Output**: trả kết quả về qua tên hàm
  - **Process:** thực hiện tính tổng của **n** số nguyên dương đầu tiên;
- Hàm **InKQ(S):**
  - **Input:** số nguyên **S**
  - **Output**: kết quả được in lên màn hình
  - **Process:** In lên màn hình giá trị của biến **S**;

KHÁI NIỆM VỀ HÀM

- **Chương trình hoàn chỉnh sử dụng hàm**

❶
```
def Nhap():
    print("Nhap mot so nguyen:")
    n=int(input("n="))
    return n
```

❷
```
def Tinh(n):
    S=0
    for x in range(1,n+1):
        S=S+x
    return S
```

❸
```
def InKQ(n,S):
    print("Tong cua ",n," so nguyen duong dau tien=",S,sep="")
```

❹
```
n=Nhap()
S=Tinh(n)
InKQ(n,S)
```

Trong đó:

❶ Hàm nhập dữ liệu
❷ Hàm tính tổng
❸ Hàm in kết quả
❹ Lời gọi hàm

KHÁI NIỆM VỀ HÀM

❑ Ví dụ 4.2:

```
❶ def hello():
❷     print('Howdy!')
      print('Howdy!!!')
      print('Hello there.')

❸ hello()
  hello()
  hello()
```

```
Howdy!
Howdy!!!
Hello there.
Howdy!
Howdy!!!
Hello there.
Howdy!
Howdy!!!
Hello there.
```

Trong đó:

❶ Phần Header

❷ Phần thân

❸ Lời gọi hàm

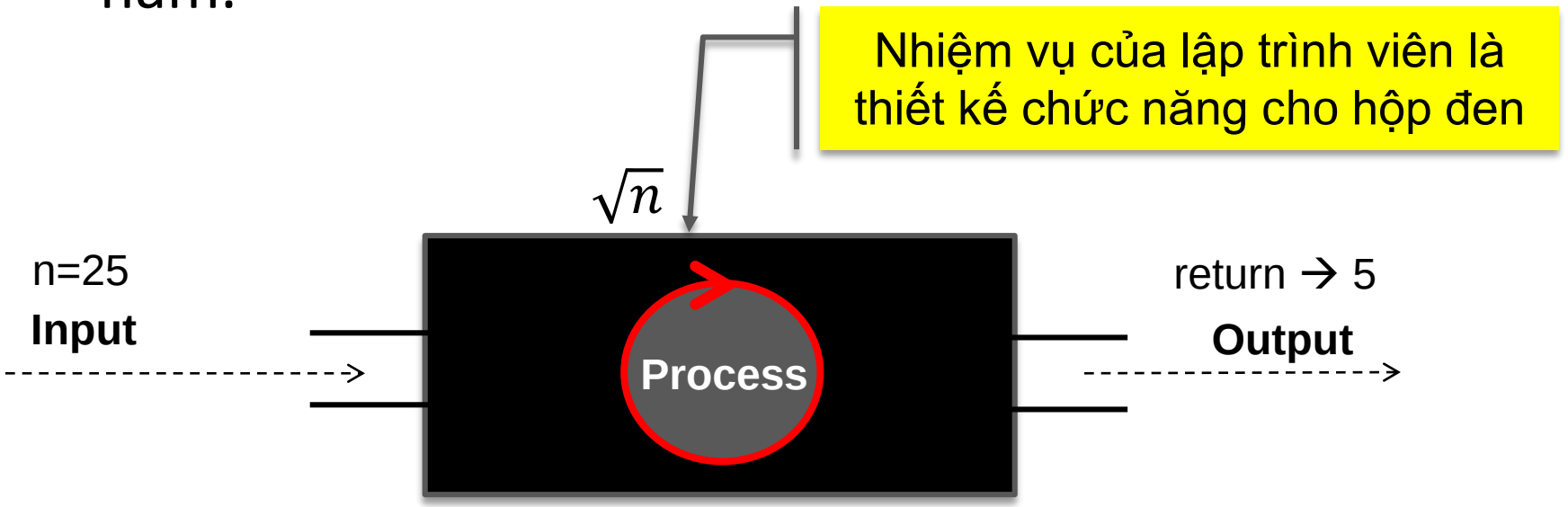
Nguồn: Automate the Boring Stuff with Python Book – Page 61

KHÁI NIỆM VỀ HÀM

- Ngoài các hàm chuẩn (do Python định nghĩa sẵn), hệ thống còn cho phép người dùng tự thiết kế ra những hàm theo mục đích riêng;
- Hàm chia các bài toán lớn thành các công việc nhỏ hơn, giúp thực hiện những công việc lặp lại nào đó một cách nhanh chóng mà không phải viết lại đoạn chương trình;
- Hàm phải được định nghĩa trước khi sử dụng;

KHÁI NIỆM VỀ HÀM

- **Ví dụ 4.3:** hàm **math.sqrt(n)**, thực hiện tính căn bậc hai của số nguyên n. Trong đó **n** là tham số của hàm.



Hàm được ví như một cái hộp đen (**Black Box**) đối với người sử dụng:
- Có hoặc không có dữ liệu đầu vào (**input**)
- Thực hiện (**process**) một yêu cầu cụ thể gì đó khi được gọi (Call)
- Kết quả xử lý được trả về gọi là đầu ra (**output**)

KHAI BÁO HÀM

- **Cú pháp khai báo hàm:**

```
def <Tên_Hàm>([<DS/Tham số>]):
        Lệnh/Khối lệnh
```

  - Trong đó:
    - **def**: là từ khóa bắt buộc khi khai báo hàm
    - <**Tên_Hàm**>: do người lập trình tự đặt, theo quy tắc đặt tên cho các đối tượng
    - <**DS/Tham số**>: các tham số (nếu có)

KHAI BÁO HÀM

❑ Ví dụ 4.4

```
❶ def hello(name):
❷     print('Hello ' + name)

❸ hello('Alice')
  hello('Bob')
```

When you run this program, the output looks like this:

```
Hello Alice
Hello Bob
```

Trong đó:

❶ Định nghĩa tên hàm, hàm có tên **hello**, tham số **name**

❷ Phần thân của hàm

❸ Lời gọi hàm, với tham số được gán giá trị '**Alice**'

Nguồn: Automate the Boring Stuff with Python Book – Page 63

THAM SỐ

- **Khái niệm**
  - **Tham số** (Parameter/ Argument - Đối số), là công cụ của hàm, cho phép đưa dữ liệu từ ngoài vào trong hàm;
  - Có loại hàm **không có tham số** và loại hàm **có tham số**;

- **Có 2 loại tham số**
  - **Tham số vị trí**
    - Là sự ánh xạ tương ứng giữa vị trí khai báo tham số và lời gọi hàm;
  - **Tham số từ khóa**
    - Sử dụng tên tham số để truyền giá trị cho hàm;

SV tham khảo thêm: https://www.geeksforgeeks.org/default-arguments-in-python/

THAM SỐ – HÀM KHÔNG CÓ THAM SỐ

- **Ví dụ 4.5a: Hàm KHÔNG có tham số**

```
def hello():
    print('Hello Alice')

hello()
```

```
Hello Alice
```

- **Ví dụ 4.5b: Hàm KHÔNG có tham số**

```
def show_number():
    for i in range(1,5):
        print(i)

show_number()
```

```
1
2
3
4
```

THAM SỐ – HÀM CÓ THAM SỐ

- **Ví dụ 4.6a: Hàm CÓ tham số**

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

hello('Alice')
hello('Bod')
```

```
Hello Alice
Hello Bod
```

THAM SỐ – HÀM CÓ THAM SỐ

- **Ví dụ 4.6b: Hàm CÓ tham số**

```
def show_number(n):
    for i in range(1,n+1):
        print(i)

show_number(4)
```

```
1
2
3
4
```

```
show_number(10)
```

THAM SỐ – HÀM CÓ THAM SỐ

- **Ví dụ 4.6c: Hàm CÓ tham số**

```
def show_number(m,n):
    for i in range(m,n+1):
        print(i)

show_number(1,4)
```

```
1
2
3
4
```

```
show_number(5,10)
```

THAM SỐ - GIÁ TRỊ MẶC ĐỊNH CỦA THAM SỐ

❑ Giá trị mặc định của tham số

- **Cú pháp:**

def <**Tên_Hàm**>(<para1>=<**value1**>, <para2>=<**value2**>,…):
    Lệnh/Khối lệnh

- **Ví dụ 4.7a**

```
def show_number(n=10):
    for i in range(1,n+1):
        print(i)

❶ show_number(5)
❷ show_number()
```

❶ n=5
➡ ❷ n=**10**

SV tham khảo thêm: https://www.geeksforgeeks.org/default-arguments-in-python/

THAM SỐ - GIÁ TRỊ MẶC ĐỊNH CỦA THAM SỐ

❑ Giá trị mặc định của tham số

- **Ví dụ 4.7b:**

```
def show_number(m=1,n=10):
    for i in range(m,n+1):
        print(i)

❶ show_number(3,7)
❷ show_number(3,)
❸ show_number()
```

❶ m=3, n=7
❷ m=3, n=**10**
❸ m=**1**, n=**10**

SV tham khảo thêm: https://www.geeksforgeeks.org/default-arguments-in-python/

THAM SỐ - GIÁ TRỊ NONE

❑ Giá trị None

- **None** là một giá trị trong Python thuộc kiểu **NoneType**;
- Biểu diễn **giá-trị-không-có-giá-trị** (value-without-a-value);
- Trong các ngôn ngữ lập trình khác được gọi là **null**, **nil** hoặc **undefined**;
- None thường được sử dụng để kết thúc hàm(với cú pháp return None) hoặc để khai báo giá trị mặc định cho tham số trong hàm;

Nguồn: Automate the Boring Stuff with Python Book – Page 65

THAM SỐ - GIÁ TRỊ NONE

- **Nếu đặt giá trị None cho tham số**, thì nó trở thành tham số không bắt buộc:
  - **Ví dụ 4.8a**

```
def PhepCong(x,y=None):
    if y==None:
        return x
    else:
        return x+y

kq1=PhepCong(5)
kq2=PhepCong(5,2)
print(kq1, kq2)
```

5 7

THAM SỐ - GIÁ TRỊ NONE

- **Nếu tất cả các tham số được đặt giá trị None**, thì có thể sử dụng tên tham số để truyền giá trị:
  - **Ví dụ 4.8b**

```
def PhepCong(x=None,y=None):
    return x+y

kq1=PhepCong(5,2)
kq2=PhepCong(x=5,y=2)
kq3=PhepCong(y=2,x=5)
print(kq1, kq2, kq3)
```

7 7 7

THAM SỐ

❑ Phân biệt giữa 2 loại tham số

Tham số vị trí	Tham số từ khóa

Khai báo hàm:
def PhepCong(x,y):
  return x+y
 	
Gọi hàm:
kq=PhepCong(5,2)
	Khai báo hàm:
	def PhepCong(x=None,y=None):
	  return x+y
	
	Gọi hàm:
	kq=PhepCong(5,2)
	
	Hoặc:
	kq=PhepCong(x=5,y=2)
	
	Hoặc:
	kq=PhepCong(y=2,x=5)

NHẬN GIÁ TRỊ TỪ HÀM – TỪ KHÓA RETURN

- **Từ khóa return sẽ kết thúc và trả giá trị về cho hàm**
  - **Ví dụ 4.9:**

```
def Nhap():
    n=int(input('Nhap mot so nguyen='))
    return n

n=Nhap()
print('So nguyen da nhap=',n)
```

```
Nhap mot so nguyen=5
So nguyen da nhap= 5
```

NHẬN GIÁ TRỊ TỪ HÀM – TỪ KHÓA RETURN

- **Từ khóa return có thể trả về 1 hoặc nhiều giá trị**
  - **Ví dụ 4.10:**

```
def Nhap():
    x=int(input('x='))
    y=int(input('y='))
    return x, y

a, b=Nhap()
print('Hai so nguyen da nhap=',a,b)
```

```
x=5
y=10
Hai so nguyen da nhap= 5 10
```

NHẬN GIÁ TRỊ TỪ HÀM – TỪ KHÓA RETURN

- **Từ khóa return có thể trả về giá trị của một biểu thức**
  - **Ví dụ 4.11:**

```
def Tong_HieuHaiSo(x,y):
    return x+y,x-y

x=10
y=7
a,b=Tong_HieuHaiSo(x,y)
print('Tong va hieu=',a,b)
```

```
Tong va hieu= 17 3
```

NHẬN GIÁ TRỊ TỪ HÀM – TỪ KHÓA RETURN

- **Có thể sử dụng nhiều từ khóa return trong hàm**
  - **Ví dụ 4.12a:**

```
❶ import random

❷ def getAnswer(answerNumber):
❸     if answerNumber == 1:
          return 'It is certain'
      elif answerNumber == 2:
          return 'It is decidedly so'
      elif answerNumber == 3:
          return 'Yes'
      elif answerNumber == 4:
          return 'Reply hazy try again'
      elif answerNumber == 5:
          return 'Ask again later'
      elif answerNumber == 6:
          return 'Concentrate and ask again'
      elif answerNumber == 7:
          return 'My reply is no'
      elif answerNumber == 8:
          return 'Outlook not so good'
      elif answerNumber == 9:
          return 'Very doubtful'

❹ r = random.randint(1, 9)
❺ fortune = getAnswer(r)
❻ print(fortune)
```

Nguồn: Automate the Boring Stuff with Python Book – Page 64

NHẬN GIÁ TRỊ TỪ HÀM

- **Ví dụ 4.12b:** một cách viết tối ưu hóa code

```
r = random.randint(1, 9)
fortune = getAnswer(r)
print(fortune)
```

to this single equivalent line: 

```
print(getAnswer(random.randint(1, 9)))
```


Nguồn: Automate the Boring Stuff with Python Book – Page 65

NHẬN GIÁ TRỊ TỪ HÀM

- Có thể sử dụng cú pháp sau để kết thúc chu kỳ vòng lặp hoặc kết thúc hoạt động của hàm nhưng không trả về giá trị:

```
return None
```

NHẬN GIÁ TRỊ TỪ HÀM

❑ Giá trị None

- **Ví dụ 4.13:**

```
def PhepChia(x,y):
    if y==0:
        return None
    else:
        return x/y

kq1=PhepChia(5,0)
kq2=PhepChia(5,2)
print(kq1, kq2)
```

➡ None 2.5

LỜI GỌI HÀM

❑ Ý nghĩa:

- Khi cần sử dụng một hàm đã định nghĩa, ta cần thực hiện lời gọi hàm;
- Hàm được gọi thông qua tên hàm, theo sau là danh sách tham số (nếu có);
- Sử dụng cú pháp hai ngôi (có phép gán =) khi hàm trả kết quả qua tên hàm;
- Một hàm có thể được gọi trong một hàm khác (gọi là phép đệ quy);

BÀI TẬP

- Sử dụng hàm để thực hiện các yêu cầu sau:
  - Nhập từ bàn phím một số nguyên n;
    - Hàm **Nhap()**
  - Nhập liên tục từ bàn phím n số nguyên;
  - Đếm và in lên màn hình có bao nhiêu chữ số chẵn đã được nhập vào.
    - Hàm **NhapVaDem(n)**
    - Hàm **InKQ**(kq)

Chương trình mẫu:

```
n=7
Nhap 7 so nguyen:
6
5
8
7
0
2
3
So chu so chan la: 3
```

PHẠM VI CỦA BIẾN

❑ Các loại biến

- **Biến cục bộ**
  - Là biến được khai báo trong một HÀM và chỉ ảnh hưởng trong phạm vi của HÀM đó;
  - Khi ra khỏi HÀM, biến sẽ không còn giá trị sử dụng;

- **Biến toàn cục**
  - Biến toàn cục được khai báo ngoài tất cả các hàm;
  - Biến toàn cục có giá trị sử dụng trong toàn bộ chương trình;

PHẠM VI CỦA BIẾN

Phạm vi **TOÀN CỤC**

```
def NhapBanKinh():
    x=int(input("Ban kinh="))
    return x
```

Phạm vi **CỤC BỘ**

```
def TinhChuVi(y):
    PI = 3.14
    ChuVi=y*PI
    return ChuVi
```

Vị trí khai báo **BIẾN CỤC BỘ** trong chương trình con

Vị trí khai báo **BIẾN TOÀN CỤC**

```
BanKinh=NhapBanKinh()
ChuVi=TinhChuVi(BanKinh)
print(ChuVi)
```

PHẠM VI CỦA BIẾN

❑ Cú pháp global

- Sử dụng từ khóa **global** để khai báo 1 biến cục bộ thành toàn cục
- **Ví dụ 4.14**

```
def spam():
❶     global eggs
❷     eggs = 'spam'

eggs = 'global'
spam()
print(eggs)
```

When you run this program, the final `print()` call will output this:

```
spam
```

Nguồn: Automate the Boring Stuff with Python Book – Page 70

HÀM RỖNG – TỪ KHÓA PASS

- Có thể định nghĩa một **hàm rỗng** với từ khóa **pass**

Sai

```
def myfunc1():

myfunc1()
```

IndentationError: expected an indented block

Đúng

```
def myfunc2():
    pass

myfunc2()
```

XỬ LÝ NGOẠI LỆ

- Khi thực hiện chương trình, người lập trình có thể chủ động kiểm soát lỗi và bỏ qua khi phát sinh;
  - **Ví dụ 4.15**

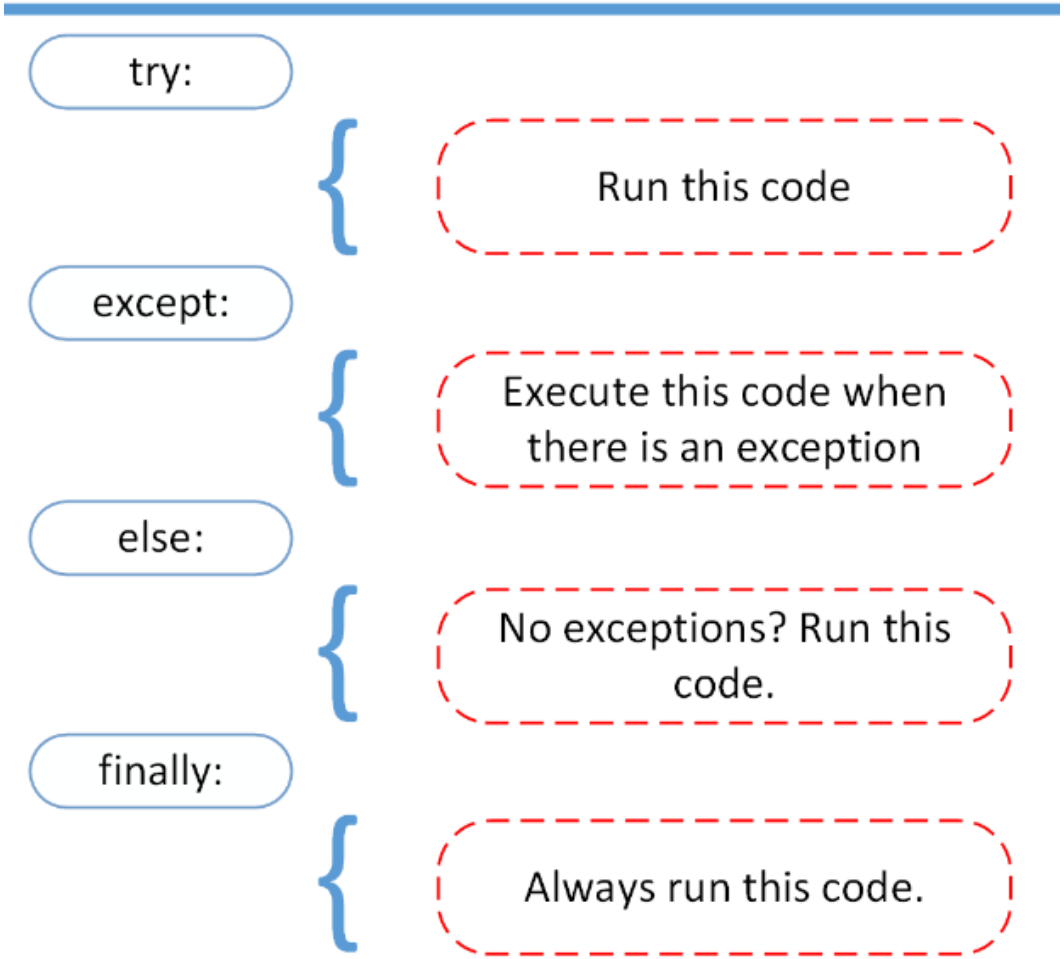
```
def spam(divideBy):
    return 42 / divideBy

print(spam(2))
print(spam(12))
print(spam(0))
print(spam(1))
```

```
21.0
3.5
Traceback (most recent call last):
  File "C:/zeroDivide.py", line 6, in <module>
    print(spam(0))
  File "C:/zeroDivide.py", line 2, in spam
    return 42 / divideBy
ZeroDivisionError: division by zero
```

XỬ LÝ NGOẠI LỆ

- Cú pháp:



Nguồn: https://realpython.com/python-exceptions/

XỬ LÝ NGOẠI LỆ

- **Ví dụ 4.16:**

```
def spam(divideby):
    try:
        result = 42 / divideby
    except:
        print("Sorry ! You are dividing by zero ")
    else:
        print("Yeah ! Your answer is :", result)

print(spam(1))
print(spam(0))
```


Nguồn: https://realpython.com/python-exceptions/