


Hàm trong python

- 01** Cấu trúc hàm
- 02** Cách gọi hàm
- 03** Thêm tài liệu cho hàm
- 04** Biến toàn cục
- 05** Tham số
- 06** Biểu thức lambda
- 07** đệ qui
- 08** Một số hàm có sẵn

1




Mục tiêu

- Hiểu được khái niệm và nguyên tắc hoạt động về hàm
- Biết cách viết hàm, gọi hàm
- Sử dụng được Global variable, Parameter mặc định
- Hiểu và thực hiện được hàm đệ quy
- Sử dụng được một số hàm có sẵn của Python: các hàm toán học, round, time, random, exit, eval...

2

2




1. Cấu trúc hàm

- Hàm là một khối lệnh thực hiện một công việc hoàn chỉnh (module), được đặt tên và được gọi thực thi nhiều lần tại nhiều vị trí trong chương trình.
- Hàm còn gọi là chương trình con (*subroutine*)
- ***Nếu không viết hàm thì sẽ gặp những khó khăn gì?***
 - Rất khó để viết chính xác khi dự án lớn
 - Rất khó debug
 - Rất khó mở rộng
- 2 loại hàm:
 - Hàm thư viện có sẵn
 - Hàm do người dùng định nghĩa

3

3



1. Cấu trúc hàm

- Ví dụ hàm thư viện


```
a, b, c = eval(input('Nhập a, b, c:'))
print('a={0}, b={1}, c={2}'.format(a, b, c))
```

Nhập a, b, c:10 , 29, 100
a=10, b=29, c=100
- Ví dụ hàm người dùng tự định nghĩa


```
def GiaiPTB1(a, b):
    if a == 0:
        if b == 0:
            return '{0}x+{1}=0: vo so nghiem'.format(a, b)
        else:
            return '{0}x+{1}=0: vo nghiem'.format(a, b)
    else:
        x = (-b) / a
        return '{0}x+{1}=0: nghiem x={2}'.format(a, b, x)
```

4

4

1. Cấu trúc hàm

■ Cú pháp: `def FunctionName([parameter list]):`

block

- Hàm có tham số hoặc không có tham số.
- Hàm có thể có giá trị trả về hoặc không.

```
def GiaiPTB1(a, b):
    if a == 0:
        if b == 0:
            return '{0}x+{1}=0: vo so nghiem'.format(a, b)
        else:
            return '{0}x+{1}=0: vo nghiem'.format(a, b)
    else:
        x = (-b) / a
        return '{0}x+{1}=0: nghiem x={2}'.format(a, b, x)
```

```
def XuatChuoi(chuoi):
    print(chuoi)
```

5

5

2. Cách gọi hàm

■ Hàm có giá trị trả về

`val=FunctionName([parameter list])`

Ví dụ:

`s=GiaiPTB1(a, b)`

■ Hàm không có giá trị trả về

`FunctionName([parameter list])`

Ví dụ:

`XuatChuoi('Xin chào Khoa CNTT')`

6

6

2. Cách gọi hàm

- Nguyên tắc hoạt động theo nguyên tắc LIFO (LAST IN FIRST OUT).

```
def double(n):
    return 2 * n
x = double(3)
print(x)
```

7

7

3. Thêm tài liệu cho hàm

- Python hỗ trợ ta bổ sung tài liệu cho hàm, việc này rất thuận lợi cho đối tác sử dụng các function do người khác định nghĩa.
- Ta có thể sử dụng 3 dấu nháy kép hoặc 3 dấu nháy đơn để viết tài liệu cho hàm. Tuy nhiên theo kinh nghiệm thì các bạn nên dùng 3 dấu nháy kép.
- Các ghi chú (tài liệu) phải được viết ở những dòng đầu tiên khi khai báo hàm.

8

8

3. Thêm tài liệu cho hàm

■ Ví dụ:

➤ Vào Tool/Python or debug console
from funcdocument import *

Python Console

```
>>> help(Max2Num)
Help on function Max2Num in module func_ex:

Max2Num(a, b)
    Tìm max 2 số

>>> help(GiaiPTB1)
Help on function GiaiPTB1 in module func_ex:

GiaiPTB1(a, b)
    Giải phương trình bậc 1: ax + b = 0
```

funcdocument.py

```
def GiaiPTB1(a, b):
    """Giải phương trình bậc 1: ax + b = 0"""
    if a == 0:
        if b == 0:
            return '{0}x+{1}=0: vo so nghiem'.format(a, b)
        else:
            return '{0}x+{1}=0: vo nghiem'.format(a, b)
    else:
        x = (-b) / a
        return '{0}x+{1}=0: nghiem x={2}'.format(a, b, x)

def Max2Num(a, b):
    """Tìm max 2 số"""
    return (a if a>b else b)
```

9

9

4. Biến toàn cục

■ Tất cả các biến khai báo trong hàm chỉ có phạm vi ảnh hưởng trong hàm, các biến này gọi là biến local. Khi thoát khỏi hàm thì các biến này không thể truy xuất được.

■ Xem tình huống sử dụng biến toàn cục và cục bộ sau:

```
1 g=5 → Global variable
2 def increment():
3     g=2 → Local variable
4     g=g+1
5     increment()
6     print(g) → Global variable
```

```
1 g=5
2 def increment():
3     global g
4     g=2
5     g=g+1
6     increment()
7     print(g)
```

10

10

5. Tham số mặc định

- Python hỗ trợ tham số mặc định
- Ví dụ:


```
def Max2Num(a, b=100):
    """Tìm max 2 số"""
    return (a if a>b else b)

print('Max2Num(10)={1}'.format(10, Max2Num(10)))
print('Max2Num({0}, {1})={2}'.format(10,20, Max2Num(10, 20)))

Max2Num(10)=100
Max2Num(10, 20)=20
```

11

11

6. Biểu thức lambda

- Python hỗ trợ khai báo hàm nặc danh qua lambda

`lambda` *parameterlist* : *expression*

 - `lambda`: từ khóa
 - *parameterlist*: tập tham số trong hàm muốn định nghĩa
 - *expression*: biểu thức đơn

12

12

6. Biểu thức lambda

■ Ví dụ

1. Định nghĩa hàm

```
def test(f, x, y):
    return f(x, y)
```

2. Gọi hàm: Trước dấu 2 chấm là từ khóa lambda, đằng sau nó là số lượng các biến được khai báo trong **test** (tính sau chữ f). Tức là nếu ta **test(f,x,y)** thì viết lambda x,y:

```
x=3
y=2
sum=test(lambda x, y: x+y, x, y)
print('{0}+{1}={2}'.format(x, y, sum))
pro=test(lambda x, y: x*y, x, y)
print('{0}*{1}={2}'.format(x, y, pro))
```

13

13

6. Biểu thức lambda


■ Vì lambda chỉ cho phép biểu thức đơn, do đó nếu muốn complex thì phải tự định nghĩa riêng. Ví dụ như sau:

```
1 def handle(f, x):
2     return f(x)
3 def soChan(x):
4     return x%2==0
5 def soLe(x):
6     return x%2==1
7 def soNguyenTo(x):
8     dem=0
9     for i in range(1,x+1):
10         if x % i is 0:
11             dem=dem+1
12     return dem==2
```


```
ret1=handle(soChan,6)
print(ret1)
ret2=handle(lambda x:soChan(x),6)
print(ret2)
ret3=handle(soLe,6)
print(ret3)
ret4=handle(lambda x:soLe(x),7)
print(ret4)
ret5=handle(soNguyenTo,5)
print(ret5)
ret6=handle(lambda x:soNguyenTo(x),9)
print(ret6)
```

14

14




7. Đệ quy




1. Một khái niệm X gọi là được định nghĩa đệ quy (*recursion*) nếu trong định nghĩa của X có sử dụng ngay chính khái niệm X.
Ví dụ:
 Giai thừa có thể định nghĩa đệ quy như sau:
 - Giai thừa của 1 bằng 1.
 - Giai thừa của n ($n > 1$) là tích của n với giai thừa $n-1$.
2. Một định nghĩa đệ quy bao gồm 2 thành phần:
 - a. Thành phần dừng: không chứa khái niệm đang định nghĩa.
 - b. Thành phần đệ quy: Có chứa khái niệm đang định nghĩa.**Trong ví dụ trên thì:**
 - Thành phần dừng: Giai thừa của 1 bằng 1.
 - Thành phần đệ quy: Giai thừa của n ($n > 1$) là tích của n với giai thừa $n-1$.

15

15



7. Đệ quy



3. Thuật toán đệ quy là thuật toán có chứa thao tác gọi đến nó.
 Thuật toán đệ quy chứa các thao tác mà trong đó có chứa thao tác gọi lại thuật toán (gọi đệ quy).
4. Một hàm được gọi từ chính nó, thì đó cũng là một hình thức đệ quy.
 Sự tương ứng giữa các lần gọi hàm và các lần ra khỏi hàm được thực hiện theo thứ tự ngược lại, nghĩa là:
 - Lần ra đầu tiên ứng với lần vào cuối cùng,
 - Và lần ra khỏi hàm cuối cùng ứng với lần đầu tiên gọi hàm

16

16

7. Đệ qui

```
def factorial(n):
    """
    Hàm tính n!
    Trả về giai thừa của n
    """
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

kq=factorial(6)
print(kq)
```

factorial(6) = 6 * factorial(5)
 = 6 * 5 * factorial(4)
 = 6 * 5 * 4 * factorial(3)
 = 6 * 5 * 4 * 3 * factorial(2)
 = 6 * 5 * 4 * 3 * 2 * factorial(1)
 = 6 * 5 * 4 * 3 * 2 * 1 * factorial(0)
 = 6 * 5 * 4 * 3 * 2 * 1 * 1
 = 6 * 5 * 4 * 3 * 2 * 1
 = 6 * 5 * 4 * 3 * 2
 = 6 * 5 * 4 * 6
 = 6 * 5 * 24
 = 6 * 120
 = 720

17

17

8. Một số hàm có sẵn

■ **math:**

- Sqrt-Căn bậc 2
- Pow - lũy thừa
- Log-> $\log(x) = \log_e x = \ln x$
- Log10- Logarit cơ số 10 của x, $\log_{10}(x) = \log_{10} x$
- Exp-tính e^x
- Degrees-Đổi radian ra độ
- Radians- Tính radian $180/\pi \cdot x$
- Fabs- tính giá trị tuyệt đối
- Sin
- Cos
- tan

```
from math import *
print('sqrt(25)=', sqrt(25))
print('pow(5,3)=', pow(5,3))
print('log(2)=', log(2))
print('log10(100)=', log10(100))
print('exp(2)=', exp(2))
print(degrees(0.5235987755982988))
print(radians(30))
```

↓

```
sqrt(25)= 5.0
pow(5,3)= 125.0
log(2)= 0.6931471805599453
log10(100)= 2.0
exp(2)= 7.38905609893065
29.999999999999996
0.5235987755982988
```

18

18

8. Một số hàm có sẵn

- **round**(số gốc, đơn vị làm tròn): làm tròn


```
a=3
b=11
c=b/a
print(c) → 3.6666666666666665
print(round(c,2)) → 3.67
```
- **random.randrange(x,y)** → lấy số ngẫu nhiên $\geq x$ và $< y$

```
import random
a=random.randrange(10,20)
```
- **exit()**: thoát khỏi chương trình

19

19

8. Một số hàm có sẵn

- **eval(chuỗi tính toán)**: có thể tự tính toán chuỗi phép toán


```
from math import sin
result=eval('10+30+sin(30)')
print(result)

x1,x2=eval(input("Nhập x1,x2:"))#nhập giá trị cách nhau dấu ,
print("x1=",x1,"x2=",x2)
print("{0}+{1}={2}".format(x1,x2,x1+x2))
```

20

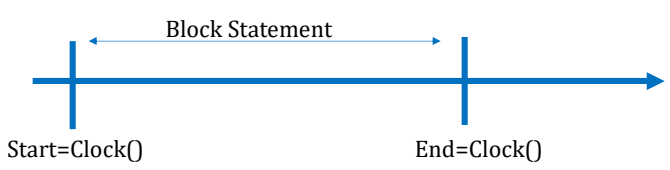
20

8. Một số hàm có sẵn

■ Time cung cấp:

- **Clock:** trả về số giây khi ta gọi hàm clock (đối với HĐH windows)
- **Sleep:** giúp ta tạm dừng quá trình chạy trong một đơn vị thời gian nào đó.

```
from time import sleep
for count in range(10, -1, -1): #
    Range 10, 9, 8, ..., 0
    print(count) # Display the
    count
    sleep(1) # Suspend execution
    for 1 second
```



Thời gian thực thi=End-Start

```
from time import clock
print("Enter your name: ", end="")
start_time = clock()
name = input()
elapsed = clock() - start_time
print(name, "it took you", elapsed,
      "seconds to respond")
```

21

21

Bài thực hành số 3

Viết chương trình tổ chức thành hàm chức năng:

Bài 1: Tìm Max 2 số

Bài 2: Tìm Max 3 số

Bài 3: Tính n!

Bài 4: Nhập số nguyên dương

Bài 5: Tính trung bình cộng của n số được nhập từ bàn phím

Bài 6: Phương trình bậc nhất 1
Viết chương trình giải phương trình bậc nhất một ẩn có dạng $ax + b = 0$.


Bài 7: Phương trình bậc 2
Viết chương trình giải phương trình bậc 2 một ẩn có dạng $ax^2 + bx + c = 0$.

Bài 8: Tính $H_N = \sum_{i=1}^n \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ (còn gọi là các số Harmonic)


Bài 9: Tính $S_n = \sum_{i=1}^n \frac{i+1}{i^2} = 2 + \frac{3}{4} + \frac{4}{9} + \dots + \frac{n+1}{n^2}$

22

22



Bài thực hành số 3



Bài 10: Phép toán số học

Viết chương trình cho phép người dùng nhập vào hai số thực x, y khác 0 và ký tự k (thuộc 1 trong 4 ký tự: +, -, *, /). Tùy thuộc vào ký tự k , hãy xuất ra tổng, hiệu, tích, thương của x và y .

Hướng dẫn:

Dùng lệnh rẽ nhánh để kiểm tra ký tự k .

Nếu $k = '+'$ thì return $x + y$. Tương tự cho các trường hợp còn lại.

Nếu $k \notin \{+, -, *, /\}$ thì return 0.

23

23



THẢO LUẬN





24

24



25