HÀM TRONG PYTHON

Hà Nội, 2026

NỘI DUNG

- Khái niệm hàm và ý nghĩa của việc sử dụng hàm
- Sử dụng các hàm có sẵn
- Định nghĩa và sử dụng hàm
- Tham số của hàm
- Biến toàn cục, biến cục bộ
- Hàm đệ quy

VÍ DỤ 8.1

- Bài toán: *Cho số nguyên dương N, hãy tìm và hiển thị các số chính phương nhỏ hơn N*

- Ý tưởng giải thuật:
  - Nhập số N
  - Sử dụng một biến k, nhận giá trị lần lượt từ 1 đến N-1
  - Với mỗi giá trị của k, kiểm tra xem k có là số chính phương không
    - Nếu có thì in ra màn hình
    - Ngược lại, chuyển đến số tiếp theo

- Nhận xét: Thao tác kiểm tra 1 số nguyên k có phải là số chính phương hay không sẽ được sử dụng nhiều lần => Có thể tách riêng đoạn chương trình này và viết thành một chương trình con, khi cần sẽ gọi để thực hiện

Ví dụ 8.1

```python
#Ví dụ 6-1: Tìm các số chính phương
#Cách 1: không sử dụng hàm

#Nhập dữ liệu
N = int(input("Cho số nguyên dương N: "))

print(f"Các số chính phương nhỏ hơn {N} là: ")
#kiểm tra lần lượt các số từ 1 đến N-1
for k in range(1, N):
   #lấy căn bậc 2 của k
   x = k ** 0.5
   #kiểm tra xem x có là số nguyên không,
   #nếu x là số nguyên thì k là số cp
   if (x.is_integer()):
        print(f"{k}", end = " ")

print(f"\nChuong trinh da ket thuc !")
```

```python
#Ví dụ 6-1: Tìm các số chính phương
#Cách 2: có sử dụng hàm

#hàm kiểm tra số chính phương
def so_chinh_phuong(k):
    check = 0  #biến lính canh
    #lấy căn bậc 2 của k
    x = k ** 0.5
    #kiểm tra xem x có là số nguyên không,
    #nếu x là số nguyên thì k là số cp
    if (x.is_integer()):
        check = 1
    return check  #trả về giá trị thông qua biến check

#Nhập dữ liệu
N = int(input("Cho số nguyên dương N: "))
print(f"Các số chính phương nhỏ hơn {N} là: ")
#kiểm tra lần lượt các số từ 1 đến N-1
for k in range(1, N):
  if (so_chinh_phuong(k)):
      print(f"{k}", end = " ")

print(f"\nChuong trinh da ket thuc !")
```

KHÁI NIỆM VÀ Ý NGHĨA CỦA HÀM

KHÁI NIỆM HÀM

- **Hàm:** một khối lệnh trong chương trình giải quyết một vấn đề cụ thể và trọn vẹn.
- Cần phải xác định được dữ liệu vào, kết quả trả về và nhiệm vụ của hàm
- Có hai loại hàm khác nhau về cách gọi để thực thi các hàm: hàm trả về giá trị (function) và hàm không trả về giá trị (procedure).
- Ý nghĩa của việc sử dụng hàm
  - Triển khai việc viết các chương trình theo mô hình chia để trị
  - Tiết kiệm thời gian và công sức vì có thể sử dụng lại các hàm đã có

CÁC LOẠI HÀM TRONG PYTHON

- Hàm có sẵn trong Python
- Hàm được xây dựng trong các thư viện
- Hàm do người lập trình tự xây dựng trong chương trình

HÀM CÓ SẴN TRONG PYTHON

- Là các hàm đã được trình dịch của Python xây dựng sẵn => cho phép dùng ngay mà không cần khai báo.
- Ví dụ: print(), input(), max(), min(), pow(),….

HÀM ĐƯỢC XÂY DỰNG TRONG CÁC THƯ VIỆN

- Python có hệ thống hơn 230 thư viện, mỗi thư viện là một tập hợp các hàm xây dựng cho một lĩnh vực hoặc một mục đích nào đó.
- Các thư viện này được để trong các file modules, cần cài đặt các modules này.
- Để sử dụng hàm của module nào đó trong chương trình, cần khai báo theo một trong hai cách sau:

```
import <tên các modules>
hoặc
from <tên module> import <tên các hàm>
```

Ví dụ:

```
import math, array
from math import sqrt, log10, cos
```

HÀM ĐƯỢC XÂY DỰNG TRONG CÁC THƯ VIỆN

- Một số thư viện phổ biến và được sử dụng nhiều trong lĩnh vực khoa học dữ liệu
  - math - thư viện các hàm toán học
  - string – thư viện các hàm xử lý chuỗi kí tự
  - numpy - thư viện xử lý mảng đa chiều và ma trận
  - matplotlib – thư viện đồ họa 2D
  - scikit-learn – thư viện xử lý ảnh và machine learning
  - keras - thư viện liên quan đến deep learning
  - tensorflow – thư viện liên quan đến machine learning và deep learning
  - …..

VÍ DỤ SỬ DỤNG CÁC HÀM TRONG MODULE

```
>>> import math
>>> print(math.ceil(5.4))
6
>>> print(math.sqrt(4))
2.0
>>> print(math.pi)
3.141592653589793
>>> print(math.cos(1))
0.5403023058681398
>>> math.factorial(6)
720
```

```
#minh họa sử dụng các hàm trong module math
#nhập vào một số nguyên và tính căn bậc hai của số đó
from math import sqrt #nạp hàm sqrt
n = int(input("Cho biết một số nguyên: "))
m = sqrt(n) # sử dụng hàm sqrt
print("Căn bậc hai của", n,"là: ", m)
```

Bài tập *Thực hành 8.1. Sử dụng các hàm có sẵn* trên hệ thống LMS

Thời gian: 10 phút

Từ : 20h15 đến: 20h25

ĐỊNH NGHĨA VÀ SỬ DỤNG HÀM

ĐỊNH NGHĨA HÀM

- Cú pháp:

```
def tên_hàm(<danh sách tham số>):
    <khối lệnh>
```

- Chú ý: sau khi khai báo và triển khai nội dung của hàm, các lệnh trong hàm chỉ hoạt động khi hàm được gọi từ một nơi nào đó trong chương trình.

- def: là từ khóa để khai báo bắt đầu một hàm.
- tên_hàm: được đặt theo quy tắc đặt tên biến, tên hàm thường đặt gợi nhớ, thể hiện nội dung của hàm.
- <danh sách tham số>: là các tham số cần truyền vào và sử dụng trong hàm, hai tham số cách nhau bởi một dấu phẩy.
- <khối lệnh>: là nội dung của hàm. Nếu hàm có trả về giá trị thì trong nội dung của hàm có ít nhất một lệnh return để trả về giá trị.
- Trong Python, hàm có thể trả về nhiều hơn một giá trị sau mỗi lần thực hiện.

VÍ DỤ

```python
#hàm cộng hai số
def tong2so(a, b):
    tong = a + b
    return tong
```

```python
#nhập các hệ số a, b, c
def nhap_du_lieu():
    a = float(input("Cho biết hệ số a: "))
    b = float(input("Cho biết hệ số b: "))
    c = float(input("Cho biết hệ số c: "))
    return a, b, c
```

```python
#hàm thực hiện 4 phép toán
def pheptoan(a, b):
    tong = a + b
    hieu = a - b
    tich = a * b
    thuong = a / b
    return tong, hieu, tich, thuong
```

```python
#phương trình có hai nghiệm phân biệt
def phuong_trinh_hai_nghiem(a, b, delta):
    x1 = (-b + sqrt(delta)) / (2 * a) #tính x1
    x2 = (-b - sqrt(delta)) / (2 * a) #tính x2
    print(f" Phương trình có hai nghiệm: ")
    print(f" x1 = {x1: .2f}, x2 = {x2: .2f}")
```

CÁCH GỌI ĐỂ THỰC THI CÁC HÀM

- Khi một hàm được định nghĩa và triển khai, nó sẽ chưa thực hiện nếu chưa được gọi
  - Khi nào cần hàm thực thi nội dung, chúng ta cần viết lời gọi hàm
- Một hàm có thể được gọi nhiều lần trong chương trình, mỗi lần được gọi, hàm sẽ được thực hiện
- Hàm được gọi thông qua tên hàm và danh sách các đối số truyền vào cho hàm
  - Mỗi tham số của hàm sẽ nhận giá trị của các đối số truyền vào theo thứ tự tương ứng từ trái qua phải
- Lời gọi hàm
  - Đối với hàm có giá trị trả về tên hàm xuất hiện trong một biểu thức hoặc phía bên phải của phép gán
  - Đối với hàm không có giá trị trả về chỉ cần viết tên hàm như một lệnh bình thường

CÁCH GỌI ĐỂ THỰC THI CÁC HÀM

```
34 #chương trình chính
35 a, b, c = nhap_du_lieu()  #gọi hàm nhập dữ liệu
36 #gọi hàm tính delta
37 delta = tinh_delta(a, b, c)
38
39 if delta < 0:
40     #gọi hàm cho trường hợp vô nghiệm
41     phuong_trinh_vo_nghiem()
42 else:
43     if delta == 0:
44         #gọi hàm cho trường hợp nghiệm kép
45         phuong_trinh_nghiem_kep(a, b)
46     else:
47         #gọi hàm cho trường hợp hai nghiệm
48         phuong_trinh_hai_nghiem(a, b, delta)
```

```
18 # lời gọi hàm xuất hiện ở vế phải của phép gán
19 f1 = ham_f(10, 20)
20 print(f'Gia tri cua ham f1 la: {f1}')
21 a = 5
22 # lời gọi hàm xuất hiện trong một biểu thức
23 f2 = f1 + ham_f(a, 25)
24 print(f'Gia tri cua ham f2 la: {f2}')
25 print(f'Gia tri cua ham f la: {ham_f(14, 2)}')
```

- Dòng 19, lời gọi hàm có 3 giá trị trả về
- Dòng 23 lời gọi hàm xuất hiện trong biểu thức
- Dòng 25 lời gọi hàm xuất hiện trong lệnh print()

- Dòng 35, lời gọi hàm có 3 giá trị trả về
- Dòng 37 lời gọi hàm có 1 giá trị trả về
- Dòng 41, 45, 48 lời gọi hàm không có giá trị trả về

CÁCH GỌI ĐỂ THỰC THI CÁC HÀM

- Để gọi được một hàm nào đó, cần phải định nghĩa và triển khai hàm đó phía trên nơi xuất hiện lời gọi hàm.
- Có thể định nghĩa các hàm lồng nhau, nghĩa là trong một hàm lại có thể định nghĩa các hàm con khác.
- Ví dụ: Hàm F1() và F2() là hai hàm lồng trong hàm F().

```
1  def F(...):
2      def F1(...):
3          # nội dung của hàm F1
4          ........
5          ........
6      def F2(...):
7          # nội dung của hàm F2
8          ........
9          ........
10     # nội dung của hàm F
11     .......
12     .......
13     F1(...)  # lời gọi hàm F1
14     F2(...)  # lời gọi hàm F2
```

Bài tập *Thực hành 8.2. Tạo và sử dụng hàm đơn giản* trên hệ thống LMS

Thời gian: 15 phút

Từ : 20h35 đến: 20h50

TRUYỀN GIÁ TRỊ CHO THAM SỐ CỦA HÀM

THAM SỐ VÀ ĐỐI SỐ CỦA HÀM

- Cần xác định được dữ liệu vào và dữ liệu ra của hàm:
  - Dữ liệu vào chính là giá trị của các tham số của hàm
  - Dữ liệu ra là các giá trị trả về của hàm
- Các biến xuất hiện trong cặp dấu ngoặc sau tên hàm trong phần định nghĩa hàm được gọi là các tham số (parameter)
- Các giá trị tương ứng với từng tham số trong lời gọi hàm được gọi là đối số (argument).
- Sự tương ứng giữa tham số và đối số trong lời gọi hàm là 1 – 1, tính từ trái qua phải.
- Trong Python, ngoại trừ các biến có cấu trúc như list, dictonary,… sẽ được truyền vào hàm theo địa chỉ, còn các biến rời rạc chỉ có truyền theo giá trị.

THAM SỐ VÀ ĐỐI SỐ CỦA HÀM

- Dòng 3: x, y là các tham số của ham_f
- Dòng 19, 23, 25: minh họa các cách gọi hàm với các đối số khác nhau
- Với mỗi lời gọi hàm, các tham số sẽ nhận được các giá trị của đối số tương ứng tính từ trái qua phải

```
1  from math import sqrt, pow  # import các hàm sqrt, pow
2  # định nghĩa hàm f(x,y)
3  def ham_f(x, y):
4      if (x > y):
5          #trường hợp x > y
6          f = sqrt(pow(x, 2) - pow(y, 2))
7          return f
8      else:
9          if (x < y):
10             #trường hợp x < y
11             f = sqrt(pow(y, 2) - pow(x, 2))
12             return f
13         else:
14             #trường hợp còn lại, x = y
15             f = sqrt(2 * pow(x, 2)) #x = y
16             return f
17
18 # lời gọi hàm xuất hiện ở vế phải của phép gán
19 f1 = ham_f(10, 20)
20 print(f'Gia tri cua ham f1 la: {f1}')
21 a = 5
22 # lời gọi hàm xuất hiện trong một biểu thức
23 f2 = f1 + ham_f(a, 25)
24 print(f'Gia tri cua ham f2 la: {f2}')
25 print(f'Gia tri cua ham f la: {ham_f(14, 2)}')
```

MINH HỌA CÁCH TRUYỀN THAM SỐ CHO HÀM

```python
def F(x, y):
    x = x + 1
    y = y + 2
    print(f' Trong hàm F: x = {x}, y = {y}')
a = 10
b = 20
print(f' Trước khi gọi hàm: a = {a}, b = {b}')
F(a, b) # gọi hàm F với hai đối số truyền vào là a và b
print(f' Sau khi gọi hàm: a = {a}, b = {b}')
```

```
Trước khi gọi hàm: a = 10, b = 20
Trong hàm F: x = 11, y = 22
Sau khi gọi hàm: a = 10, b = 20
```

```python
#hàm đổi giá trị hai biến cho nhau
def doigiatri(x, y):
    tg = x
    x = y
    y = tg

a = 120
b = 831
print("-" * 36)  #hiển thị đường kẻ
print(f' Trước khi gọi hàm: a = {a}, b = {b}')
doigiatri(a, b) # gọi hàm đổi giá trị hai biến
print("-" * 36)
print(f' Sau khi gọi hàm: a = {a}, b = {b}')
print("-" * 36)
```

Đoạn trình này cho kết quả là gì?

MINH HỌA CÁCH TRUYỀN THAM SỐ CHO HÀM

- Vậy làm thế nào để truyền được giá trị của tham biến (hoặc biến) ra bên ngoài?
- Khi cần truyền giá trị của 1 tham biến (hoặc một biến) của hàm ra ngoài, hãy coi như đó là một giá trị trả về của hàm.
- Ví dụ:

```
5  #nhập các hệ số a, b, c
6  def nhap_du_lieu():
7      a = float(input("Cho biết hệ số a: "))
8      b = float(input("Cho biết hệ số b: "))
9      c = float(input("Cho biết hệ số c: "))
10     return a, b, c
```

```
a, b, c = nhap_du_lieu()  #gọi hàm nhập dữ liệu
```

MINH HỌA CÁCH TRUYỀN THAM SỐ CHO HÀM

```python
#hàm đổi giá trị hai biến cho nhau
def doigiatri(x, y):
    tg = x
    x = y
    y = tg
    #trả về giá trị trong biến x và y
    return x, y

a = 120
b = 831
print("-" * 36)  #hiển thị đường kẻ
print(f' Trước khi gọi hàm: a = {a}, b = {b}')

a, b = doigiatri(a, b) # gọi hàm đổi giá trị hai biến

print("-" * 36)
print(f' Sau khi gọi hàm: a = {a}, b = {b}')
print("-" * 36)
```

```
------------------------------------
 Trước khi gọi hàm: a = 120, b = 831
------------------------------------
 Sau khi gọi hàm: a = 831, b = 120
------------------------------------
```

THAM SỐ CÓ GIÁ TRỊ MẶC ĐỊNH

- Python cho phép gán giá trị mặc định cho một số tham số, khi gọi hàm, nếu các tham số này có đối số tương ứng truyền vào, thì chúng sẽ nhận giá trị là các đối số này, trong trường hợp ngược lại thì sẽ nhận giá trị mặc định đã được gán khi định nghĩa hàm
- Trong cùng một hàm, các tham số có giá trị mặc định phải được khai báo bên phải của tất cả các tham số không có giá trị mặc định.
- Khi gọi hàm có các tham số có giá trị mặc định, thì có thể không cần truyền giá trị cho các tham số này.

THAM SỐ CÓ GIÁ TRỊ MẶC ĐỊNH

```python
#hàm có tham biến có giá trị mặc định
def cong4so(a, b, c = 10, d = 20):
    t = a + b + c + d
    return t

# gọi hàm tính tổng 4 số
print(f'Tổng lần thứ nhất là: {cong4so(15, 8, 5, 9)}')
print("-" * 25)
# gọi hàm sử dụng giá trị mặc định cho d
print(f'Tổng lần thứ hai là: {cong4so(15, 8, 5)}')
print("-" * 25)
# gọi hàm sử dụng giá trị mặc định cho các c và d
print(f'Tổng lần thứ ba là: {cong4so(15, 8)}')
```

a, b là các tham số bình thường

c, d là các tham số có giá trị mặc định lần lượt là 10 và 20; khi lời gọi hàm không truyền giá trị cho các tham số này, thì giá trị mặc định sẽ được sử dụng

```
Tổng lần thứ nhất là: 37
-------------------------
Tổng lần thứ hai là: 48
-------------------------
Tổng lần thứ ba là: 53
```

BÀI TẬP 6-2

- Các chương trình dưới đây cho kết quả là gì?

```
3  def F(n):
4      for i in range(1, n+1, 1):
5          if i % 2 == 0:
6              print('Y')
7  m = 5
8  for i in range(1, m+1, 1):
9      F(i);
```

```
1  #BT6-2b
2  def F(x):
3      x = x + 1
4      return x
5  def G(y):
6      y = y - 1
7      return F(y)
8  def main():
9      print(f'Kết quả: {F(4) + G(8)}')
10 main()
```

Bài tập *Thực hành 8.3. Truyền tham số cho hàm* trên hệ thống LMS

Thời gian: 15 phút

Từ : 21h00 đến: 21h15

BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

KHÁI NIỆM BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

- Khi sử dụng hàm, xuất hiện hai loại biến
  - Biến toàn cục (global variables) là các biến mà giá trị của nó có thể được truy cập và chỉnh sửa trong suốt chương trình.
  - Biến cục bộ (local variables) là các biến được phát sinh trong một hàm và chỉ có thể sử dụng trong phạm vi của hàm đó, khi hàm kết thúc thì các biến cục bộ của nó cũng được giải phóng và không còn tồn tại
- **Chú ý**: Tham biến cũng là một loại biến cục bộ, tuy nhiên tham biến có thể nhận giá trị từ bên ngoài hàm thông qua lời gọi hàm

MINH HỌA BIẾN TOÀN CỤC VÀ BIẾN CỤC BỘ

- Chương trình dưới đây cho kết quả là gì?

```
1  x = 5
2  def nhandoi(y):
3      x = x * 2
4      y = y * 2
5      x = x * y
6
7  print(f"Gía trị x trước khi gọi hàm: {x}")
8
9  nhandoi(3)  #gọi hàm
10
11 print(f"Giá trị x sau khi gọi hàm: {x}")
```

```
1  x = 5
2  def nhandoi(y):
3      global x
4      x = x * 2
5      y = y * 2
6      x = x * y
7
8  print(f"Gía trị x trước khi gọi hàm: {x}")
9
10 nhandoi(3)  #gọi hàm
11
12 print(f"Giá trị x sau khi gọi hàm: {x}")
```

Khi cần sử dụng một biến toàn cục trong một hàm thì trong hàm cần khai báo biến đó là biến toàn cục bằng từ khóa local, giống như dòng 3 ở ví dụ trên.

MỘT SỐ ĐIỂM CẦN LƯU Ý

- Sử dụng hàm có nhiều lợi ích trong lập trình, nhất là đối với các chương trình giải các bài toán lớn, phức tạp
- Hàm có giá trị trả về cần có lệnh return trong thân hàm, hàm không có giá trị trả về không cần lệnh return
- Hàm có thể trả về nhiều giá trị thông qua tên hàm (khác với các ngôn ngữ lập trình khác)
- Chỉ có một cách truyền tham biến cho hàm là truyền theo giá trị; khi cần truyền giá trị của tham biến (hoặc của biến) trong hàm ra ngoài cần coi đó là một giá trị trả về của hàm
- Muốn sử dụng một biến toàn cục trong một hàm nào đó, cần sử dụng từ khóa global để khai báo cho biến đó

Bài tập *Thực hành 8.4. Biến toàn cục và biến cục bộ* trên hệ thống LMS

Thời gian: 15 phút

Từ : 21h15 đến: 21h30

HÀM ĐỆ QUY

ĐỆ QUY

- Một bài toán P được gọi là có tính chất đệ quy khi lời giải của nó có thể đưa về lời giải của bài toán P' nhỏ hơn nó và có dạng giống nó, đồng thời lời giải của P' không cần dùng tới P.
- Lời giải cho những bài toán như vậy được gọi là giải thuật đệ quy.
- Ví dụ:
  - Bài toán tính n!
  - Bài toán tìm số Fibonaxi
  - …

HÀM ĐỆ QUY

- Các lời giải đệ quy cho một bài toán có thể được viết gọn vào trong một hàm, hàm đó được gọi là hàm đệ quy.
- Đặc điểm của một hàm đệ quy là luôn luôn có lời gọi lại tới chính nó (thực tế bài toán đệ quy cũng là đi giải lại chính bài toán đó nhưng với kích thước nhỏ hơn.
- Khôn dạng của hàm đệ quy

```
def  ten_ham(<danh sách tham số>):

    ...
    ...

    return ten_ham(<danh sách tham số>)
```

HÀM ĐỆ QUY

- Các thành phần của hàm đệ quy
  - Phần neo: Chính là lời giải cho bài toán cơ sở, cũng là phần thể hiện tính dừng của thuật toán. Khi hàm đệ quy tự gọi lại chính nó tới một thời điểm nhất định thì sẽ phải đạt được phần neo; lời giải của phần neo có thể tìm được trực tiếp mà không cần thông qua bài toán nhỏ hơn nào cả.
  - Phần đệ quy: Trong trường hợp bài toán chưa thể giải được bằng phần neo, ta sẽ xác định các bài toán con và gọi đệ quy tới các bài toán con đó để giải chúng. Sau khi đã có lời giải của các bài toán con rồi, ta phối hợp chúng lại bằng công thức truy hồi để có được lời giải của bài toán ban đầu. Phần này thể hiện tính quy nạp của thuật toán.

HÀM ĐỆ QUY

- Ví dụ
  - Hàm tính N!

```
#tinh giai thua
def nGiaiThua(n):
    if n == 1:          #phần neo của đệ quy
        return 1
    else:
        return n * nGiaiThua(n - 1)  #gọi đệ quy
```

  - Hàm tìm số Fibonacci

```
#tim so fibonacci
def timFibo(n):
    if (n == 1) or (n == 2):      #phần neo của đệ quy
        return 1
    else:
        return timFibo(n - 2) + timFibo(n - 1) #gọi đệ quy
```

Bài tập *Thực hành 8.5. Hàm đệ quy* trên hệ thống LMS

Thời gian: 1 phút

Từ : 21h40 đến: 21h50