MÔN HỌC

LẬP TRÌNH CƠ BẢN

Giảng viên: Trần Bảo Trung

PYTHON

HÀM TRONG PYTHON

Part 1

Giảng viên: Trần Bảo Trung

MỤC TIÊU BÀI:

- Cách hàm hoạt động trong Python
- Cách khởi tạo và gọi hàm
- Cơ chế truyền đối số trong hàm
- Cách nhận lại kết quả từ hàm

HÀM LÀ GÌ?

HÀM LÀ GÌ?

HÀM LÀ GÌ?

Hàm – function: Hàm là một loại ánh xạ giữa hai tập hợp số liên kết mọi phần tử của tập số đầu tiên với đúng một phần tử của tập số thứ hai

HÀM LÀ GÌ?

Hàm – function: Hàm là một loại ánh xạ giữa hai tập hợp số liên kết mọi phần tử của tập số đầu tiên với đúng một phần tử của tập số thứ hai

HÀM LÀ GÌ?

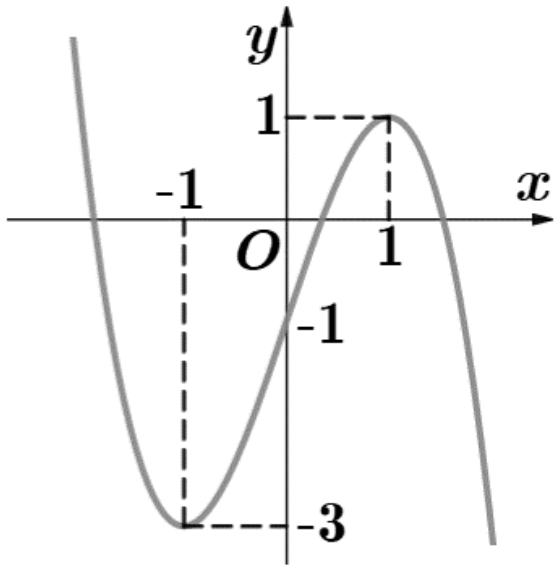
Hàm – function: Hàm là một loại ánh xạ giữa hai tập hợp số liên kết mọi phần tử của tập số đầu tiên với đúng một phần tử của tập số thứ hai

$$z = f(x,y)$$

HÀM LÀ GÌ?

Hàm – function: Hàm là một loại ánh xạ giữa hai tập hợp số liên kết mọi phần tử của tập số đầu tiên với đúng một phần tử của tập số thứ hai

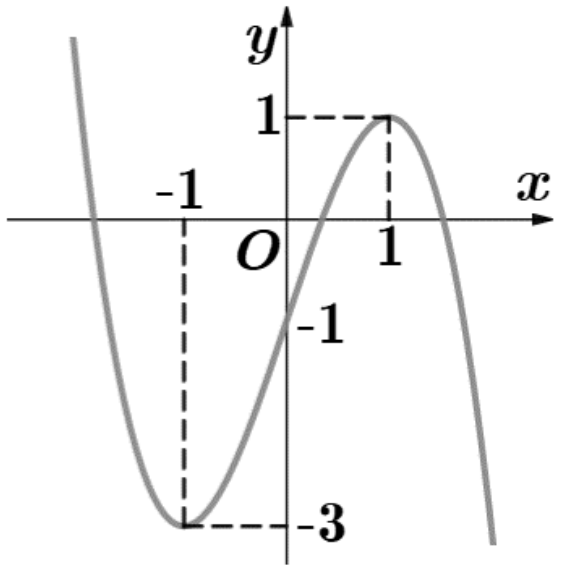
```
z = f(x,y)
```

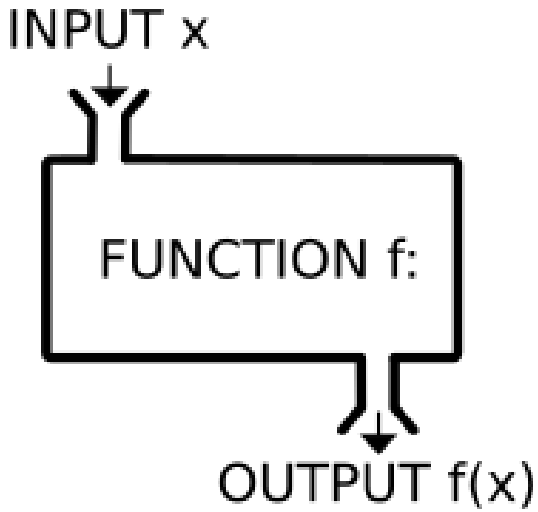


HÀM LÀ GÌ?

Hàm – function: Hàm là một loại ánh xạ giữa hai tập hợp số liên kết mọi phần tử của tập số đầu tiên với đúng một phần tử của tập số thứ hai

```
z = f(x,y)
```





HÀM LÀ GÌ?

Hàm – function: Trong lập trình, một hàm là một khối mã độc lập đóng gói một tác vụ cụ thể hoặc nhóm tác vụ liên quan

HÀM LÀ GÌ?

Hàm – function: Trong lập trình, một hàm là một khối mã độc lập đóng gói một tác vụ cụ thể hoặc nhóm tác vụ liên quan

Ví dụ các hàm đã được tiếp cận trong Python:

- Hàm len(): Thực hiện nhiệm vụ đếm số lượng phần tử trong một kiểu dữ liệu tuần tự
- Hàm int(): Thực hiện ép kiểu dữ liệu sang dạng số nguyên
- Hàm sort(): Thực hiện sắp xếp thứ tự các phần tử trong một kiểu dữ liệu tuần tự

TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

Giảng viên: Trần Bảo Trung

TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

Trong quá trình lập trình thường sẽ gặp phải nhiều vấn đề:
- Khối lượng các dòng lệnh lớn
- Lặp lại các thao tác thực thi
- Khó truy vết, khó khắc phục lỗi
- Quá tải tên biến

TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

Trong quá trình lập trình thường sẽ gặp phải nhiều vấn đề:
- Khối lượng các dòng lệnh lớn
- Lặp lại các thao tác thực thi
- Khó truy vết, khó khắc phục lỗi
- Quá tải tên biến

=> Xây dựng hàm sẽ giúp giải quyết các vấn đề này

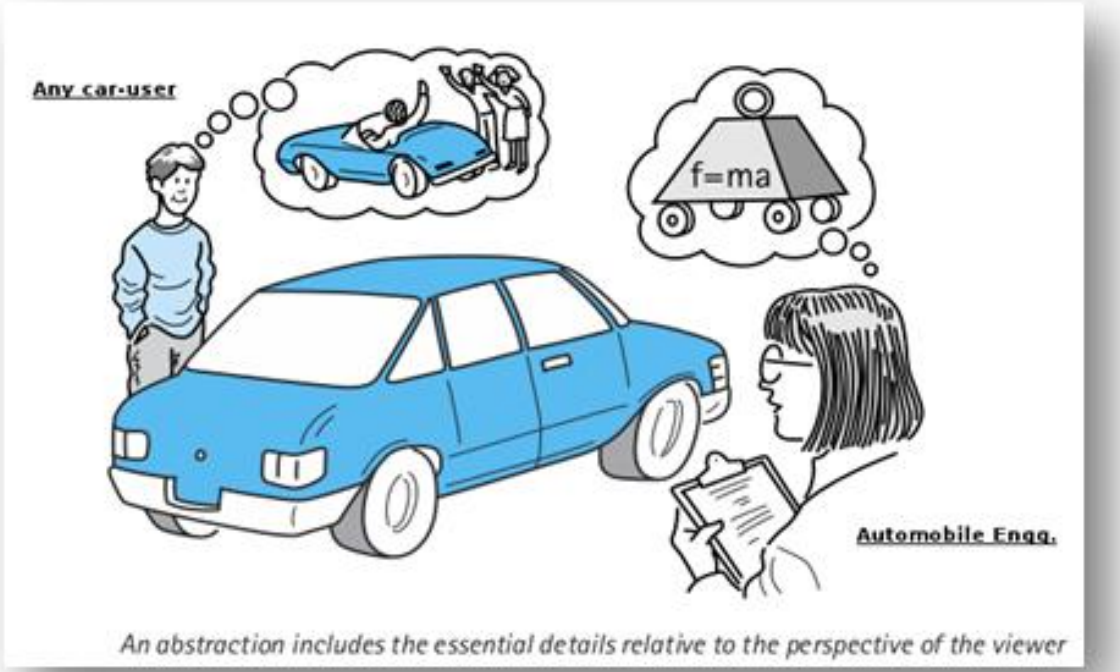
TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

Hàm trong lập trình có các tính chất:

- Abstraction and Reusability: trừu tượng và tái sử dụng
- Modularity: tính mô-đun
- Namespace Separation: tách biệt không gian tên

TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

Abstraction and Reusability:



An abstraction includes the essential details relative to the perspective of the viewer

TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

Abstraction and Reusability:

Tính trừu tượng (Abstraction) thể hiện rõ nhất khi sử dụng hàm. Người dùng không cần quan tâm đến nội dung các khối lệnh và cách chúng thực hiện trong hàm như thế nào, mà họ chỉ cần quan tâm:

- Hàm yêu cầu những **tham số** nào? (nếu có)
- Hàm trả về **giá trị** như thế nào? (nếu có)

TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

Abstraction and Reusability:

Tính tái lặp (Reusability) phản ánh bởi công việc mà hàm đó thực thi. Thông thường trong lập trình sẽ có những công việc phải thực hiện đi thực hiện lại với các giá trị dữ liệu khác nhau. Việc đưa nhiệm vụ này vào hàm sẽ giúp rút ngắn thời gian lập trình do người dùng không cần phải viết lại các câu lệnh thao tác cho công việc này nữa.

TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

Modularity:

TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

Modularity:

Tính mô-đun (Modularity) được thể hiện qua việc hàm là các khối lệnh riêng biệt. Do nó là khối lệnh riêng biệt, nó có thể đưa sang các tệp tin, tệp lưu trữ khác hoặc xây dựng thành một package hoặc thư viện

TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

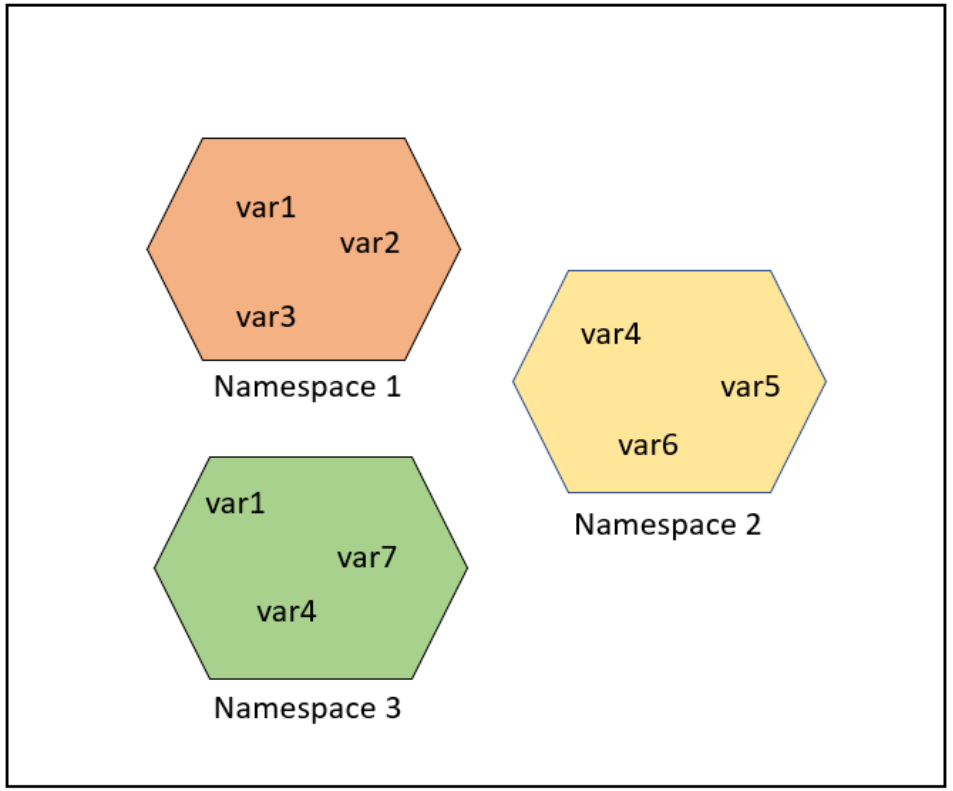
Modularity:

Tính mô-đun (Modularity) được thể hiện qua việc hàm là các khối lệnh riêng biệt. Do nó là khối lệnh riêng biệt, nó có thể đưa sang các tệp tin, tệp lưu trữ khác hoặc xây dựng thành một package hoặc thư viện

Khi xảy ra lỗi, người dùng chỉ cần quan tâm đến mô-đun bị lỗi mà không cần phải đi vào ra soát toàn bộ chương trình

TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

Namespace Separation:



TẦM QUAN TRỌNG CỦA HÀM TRONG PYTHON

Namespace Separation:

Mỗi hàm được coi là một khối lệnh riêng biệt, do vậy mỗi hàm sẽ chứa một không gian tên (namespace) khác nhau. Các biến được khai báo trong không gian tên khác nhau sẽ không ảnh hưởng lên nhau, do vậy việc tái sử dụng lại tên được cho phép, giúp giảm bớt khó khăn trong quá trình lập trình

GỌI HÀM VÀ KHỞI TẠO HÀM

Giảng viên: Trần Bảo Trung

GỌI HÀM VÀ KHỞI TẠO HÀM

GỌI HÀM VÀ KHỞI TẠO HÀM

```
def <function_name>(<parameters>):
    <statement(s)>
```

GỌI HÀM VÀ KHỞI TẠO HÀM

```
def <function_name>(<parameters>):
    <statement(s)>
```

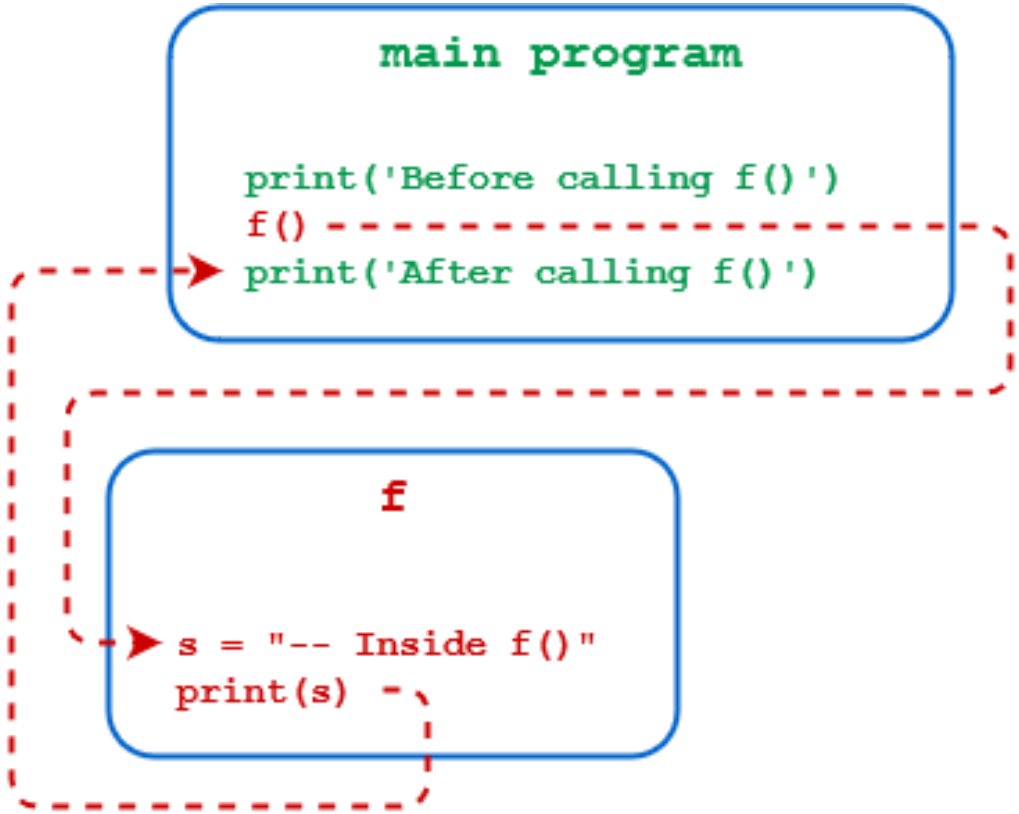
Thành phần	Ý nghĩa
def	Từ khóa để Python hiểu đây là khởi tạo hàm
<function_name>	Tên của hàm (tuân thủ đúng theo nguyên tắc đặt tên)
<parameters>	Các tham số đầu vào của hàm (có thể có hoặc không), các tham số cách nhau bởi dấu “,”
:	Dấu hai chấm thông báo kết thúc khởi tạo và bắt đầu tiến hành viết lệnh trong hàm
<statement(s)>	Các dòng lệnh xử lý trong hàm

GỌI HÀM VÀ KHỞI TẠO HÀM

```
1   def f():
2       s = '-- Inside f()'
3       print(s)
4
5   print('Before calling f()')
6   f()
7   print('After calling f()')
```

GỌI HÀM VÀ KHỞI TẠO HÀM

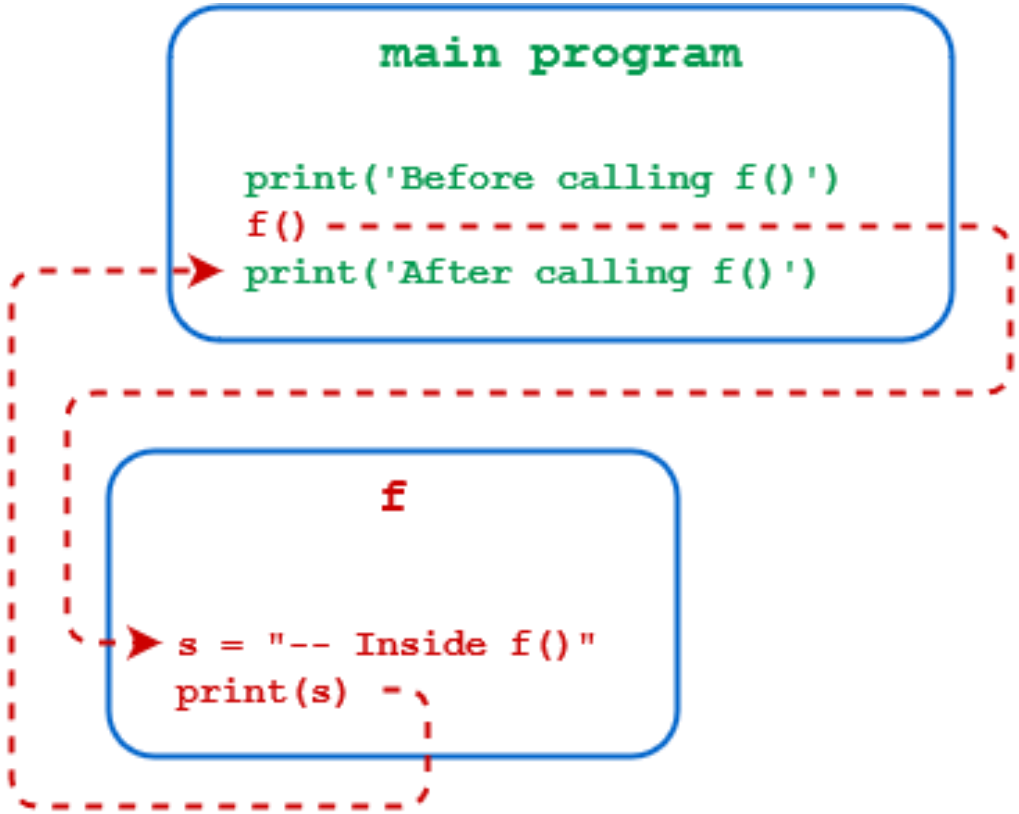
```
1    def f():
2        s = '-- Inside f()'
3        print(s)
4
5    print('Before calling f()')
6    f()
7    print('After calling f()')
```



GỌI HÀM VÀ KHỞI TẠO HÀM

Lưu ý:

- Hàm chỉ hoạt động khi chúng được gọi đến tên
- Hàm thực hiện công việc tại vị trí mà nó được gọi tên
- Hàm chỉ nên thực hiện một công việc duy nhất, không thực hiện nhiều công việc một lúc



HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

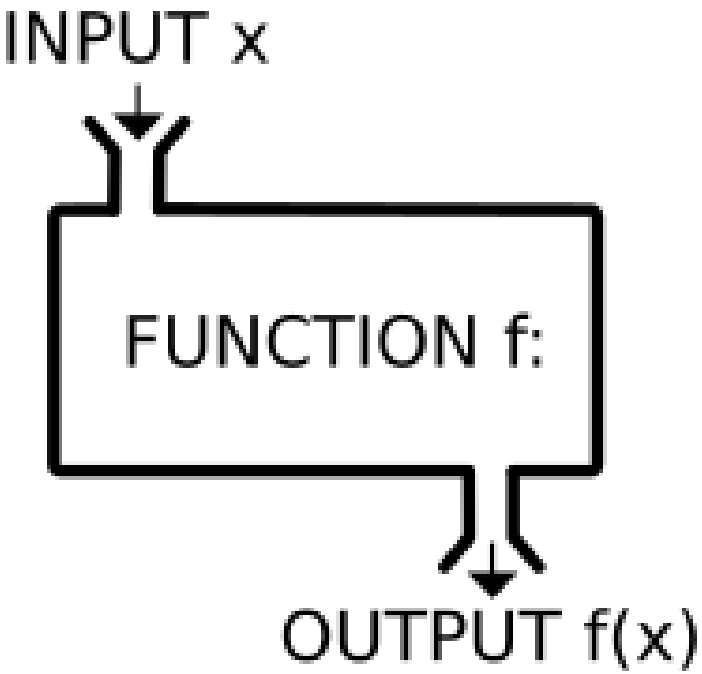
Giảng viên: Trần Bảo Trung

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

Giảng viên: Trần Bảo Trung

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

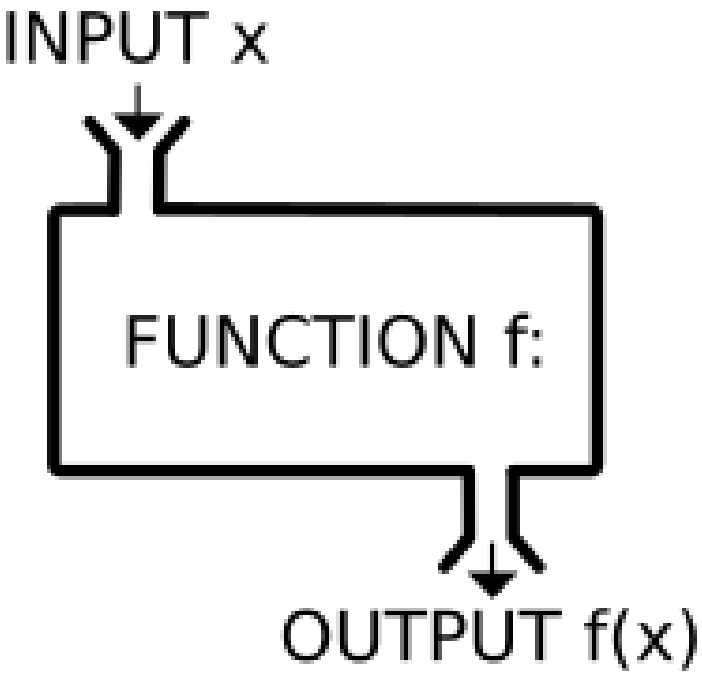
Hàm có tham số là những hàm cho phép nhận giá trị bên ngoài vào và xử lý tác vụ dựa trên những giá trị đó, các giá trị này được gọi là đối số



HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

Hàm có đối số là những hàm cho phép nhận giá trị bên ngoài vào và xử lý tác vụ dựa trên những giá trị đó , các giá trị này được gọi là đối số

```
def <function_name>(<parameters>):
    <statement(s)>
```



HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

Hàm nhận các loại đối số sau:

- Posisinal Arguments: đối số theo vị trí
- Keyword Arguments: đối số theo từ khóa
- Default Parameters: tham số mặc định

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

Posisinal Arguments:

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

Posisinal Arguments:

Đối số theo vị trí hay còn gọi là đối số bắt buộc (required arguments) là đối số mà khi người dùng sử dụng hàm, họ bắt buộc phải truyền các đối số vào vị trí các tham số khởi tạo trong hàm sao cho:

- Đủ số lượng đối số (số lượng đối số tương ứng với tham số)
- Đúng thứ tự đối số (đối số truyền vào đúng thức tự với tham số)

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

```
1   def f(qty, item, price):
2       print(f'{qty} {item} cost ${price:.2f}')
3
4   f(6, 'bananas', 1.74)
```

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

```
1   def f(qty, item, price):
2       print(f'{qty} {item} cost ${price:.2f}')
3
4   f(6, 'bananas', 1.74)
```

Function Call → **Function Definition**

`f(6, 'bananas', 1.74)` → `def f(qty, item, price):`

arguments (actual parameters) — *parameters (formal parameters)*

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

Keyword Arguments:

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

Keyword Arguments:

Khi gọi đến các hàm có chứa tham số, người dùng có thể gọi tên trực tiếp đến các tham số để truyền đối số vào. Khi sử dụng cách này, người dùng không cần quan tâm đến vị trí của các tham số được gọi tên nữa

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

Keyword Arguments:

Khi gọi đến các hàm có chứa tham số, người dùng có thể gọi tên trực tiếp (khóa) đến các tham số để truyền đối số vào. Khi sử dụng cách này, người dùng không cần quan tâm đến vị trí của các tham số được gọi tên nữa

Lưu ý:

- Đối số truyền vào không đi kèm khóa sẽ phải được gọi đúng vị trí
- Đối số kèm khóa không được nằm giữa các đối số không có khóa

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

```
1   def f(qty, item, price):
2       print(f'{qty} {item} cost ${price:.2f}')
3
4   f(6, 'bananas', 1.74)
5   f(qty=6, item='bananas', price=1.74)
6   f(item='bananas', price=1.74, qty=6)
7   f(6, price=1.74, item='bananas')
8   f(6, 'bananas', price=1.74)
```

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

```
1    def f(qty, item, price):
2        print(f'{qty} {item} cost ${price:.2f}')
3
4    f(6, item='bananas', 1.74)
5    #SyntaxError: positional argument follows keyword argument
```

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

```
1    def f(qty, item, price):
2        print(f'{qty} {item} cost ${price:.2f}')
3
4    f(6, item='bananas', 1.74)
5    #SyntaxError: positional argument follows keyword argument
```

Lỗi xảy ra do đối số không có khóa xuất hiện sau đối số có khóa

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

Default parameters:

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

Default parameters:

Hàm chứa tham số mặc định là những hàm mà tham số được khởi tạo có kèm theo giá trị. Khi sử dụng hàm này, người dùng không cần truyền hết các đối số cho tham số, những tham số nào không được truyền đối số sẽ tự động sử dụng các giá trị mặc định thay thế

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

```
1   def f(qty=6, item='bananas', price=1.74):
2       print(f'{qty} {item} cost ${price:.2f}')
3
4   f(4, 'apples', 2.24)
5   f(4, 'apples')
6   f(4)
7   f()
8   f(item='kumquats', qty=9)
9   f(price=2.29)
```

HÀM CÓ THAM SỐ VÀ CÁCH TRUYỀN ĐỐI SỐ

Tóm lại:

- Posisinal Arguments: tuân thủ theo vị trí và số lượng tham số được khởi tạo trong hàm
- Keyword Arguments: tuân thủ theo số lượng tham số được khởi tạo trong hàm, vị trí tham số không quan trọng
- Default Parameters: loại bỏ hoàn toàn yêu cầu về số lượng và vị trí khi truyền vào đối số

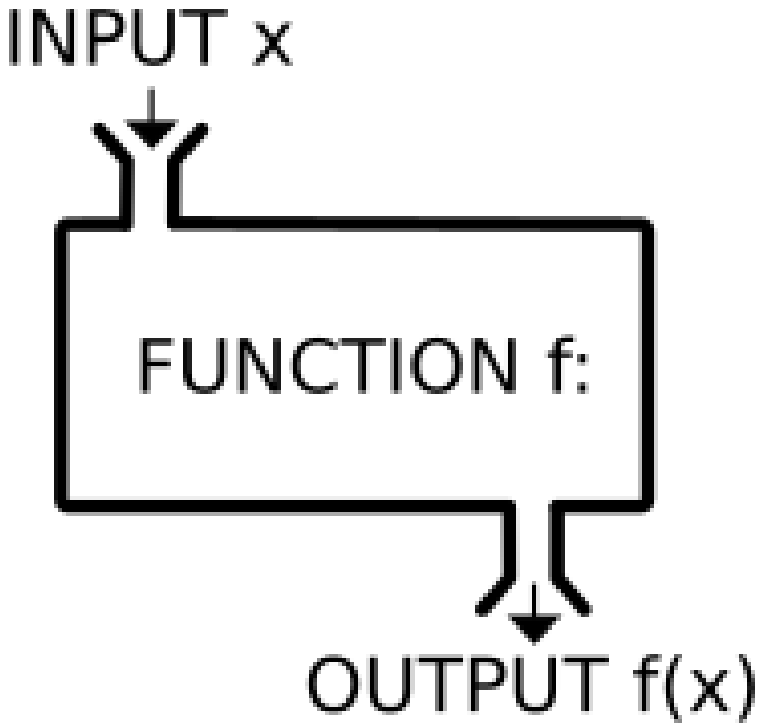
HÀM TRẢ VỀ GIÁ TRỊ

Giảng viên: Trần Bảo Trung

HÀM TRẢ VỀ GIÁ TRỊ

HÀM TRẢ VỀ GIÁ TRỊ

Khi khởi tạo và sử dụng hàm, kết quả trả về từ hàm là yêu cầu quan trọng nhất và các kết quả trả về phải sử dụng cho công việc khác được



HÀM TRẢ VỀ GIÁ TRỊ

Lệnh return:

Lệnh return là lệnh được sử dụng trong hàm, nó thực hiện hai nhiệm vụ sau:

- Nó ngay lập tức kết thúc hàm và chuyển quyền điều khiển thực thi trở lại cho người dùng.
- Nó cung cấp một cơ chế mà hàm có thể chuyển dữ liệu trở lại cho người dùng.

HÀM TRẢ VỀ GIÁ TRỊ

Kết thúc thực thi hàm:

HÀM TRẢ VỀ GIÁ TRỊ

Kết thúc thực thi hàm:

Khi chạy các câu lệnh trong hàm, nếu return được thực thi, các câu lệnh sau return sẽ không được thực thi

HÀM TRẢ VỀ GIÁ TRỊ

Kết thúc thực thi hàm:

Khi chạy các câu lệnh trong hàm, nếu return được thực thi, các câu lệnh sau return sẽ không được thực thi

```
1    def f():
2        print('foo')
3        print('bar')
4        return
5
6    f()
```

HÀM TRẢ VỀ GIÁ TRỊ

Kết thúc thực thi hàm:

Khi chạy các câu lệnh trong hàm, nếu return được thực thi, các câu lệnh sau return sẽ không được thực thi

```
1    def f():
2        print('foo')
3        print('bar')
4        return
5
6    f()
```

```
1    def f(x):
2        if x < 0:
3            return
4        if x > 100:
5            return
6        print(x)
7
8    f(-3)
9    f(105)
10   f(64) #64
```

HÀM TRẢ VỀ GIÁ TRỊ

Kết thúc thực thi hàm:

Khi chạy các câu lệnh trong hàm, nếu return được thực thi, các câu lệnh sau return sẽ không được thực thi

```
def f():
    if error_cond1:
        return
    if error_cond2:
        return
    if error_cond3:
        return
    <normal processing>
```

HÀM TRẢ VỀ GIÁ TRỊ

Trả về kết quả sau khi thực thi hàm:

HÀM TRẢ VỀ GIÁ TRỊ

Trả về kết quả sau khi thực thi hàm:

Lệnh return có thể trả về kết quả về chương trình chính nếu như return được nhận vào một giá trị (thông thường giá trị này sẽ là kết quả của công việc thực thi)

HÀM TRẢ VỀ GIÁ TRỊ

Trả về kết quả sau khi thực thi hàm:

Lệnh return có thể trả về kết quả về chương trình chính nếu như return được nhận vào một giá trị (thông thường giá trị này sẽ là kết quả của công việc thực thi)

```
1    def f():
2        return 'foo'
3
4    s = f()
5    print(s) #'foo'
```

```
1    def f():
2        str_1 = 'foo'
3        return str_1
4
5    s = f()
6    print(s) #'foo'
```

CẢM ƠN ĐÃ LẮNG NGHE

Giảng viên: Trần Bảo Trung