PERL & PYTHON

Course Code: CS466

No. of Credit Hours: 2 credits (LEC)

Lecturer: Dang, Tran Huu Minh

Mobile/ Zalo: 0918.763.367

Email: tranhminhdang@dtu.edu.vn

GIỚI THIỆU
THÔNG TIN SLIDE

Chủ đề:

Function & Module In Perl & Python

Thời lượng: 120 phút

Lecturer: Dang, Tran Huu Minh
Mobile/ Zalo: 0918.763.367
Email: tranhminhdang@dtu.edu.vn

Reference materials

Textbooks:

1. Randal L. Schwartz, Tom Phoenix, Brian D Foy (2021). *Learning Perl: Making Easy Things Easy and Hard Things Possible, 8th Edition*. O'Reilly Media.
2. Eric Chou (2023). *Mastering Python Networking, Fourth Edition*. Packt Publishing.

Reference Materials:

1. Florian Dedov (2020). *The Python Bible 7 in 1: Volumes One To Seven (Beginner, Intermediate, Data Science, Machine Learning, Finance, Neural Networks, Computer Vision)*. Independently published.

Function in Perl

Hàm trong Perl là một khối mã thực hiện một tác vụ cụ thể. Nó cho phép bạn chia nhỏ chương trình thành các phần nhỏ hơn, dễ quản lý và tái sử dụng.

Tái sử dụng mã

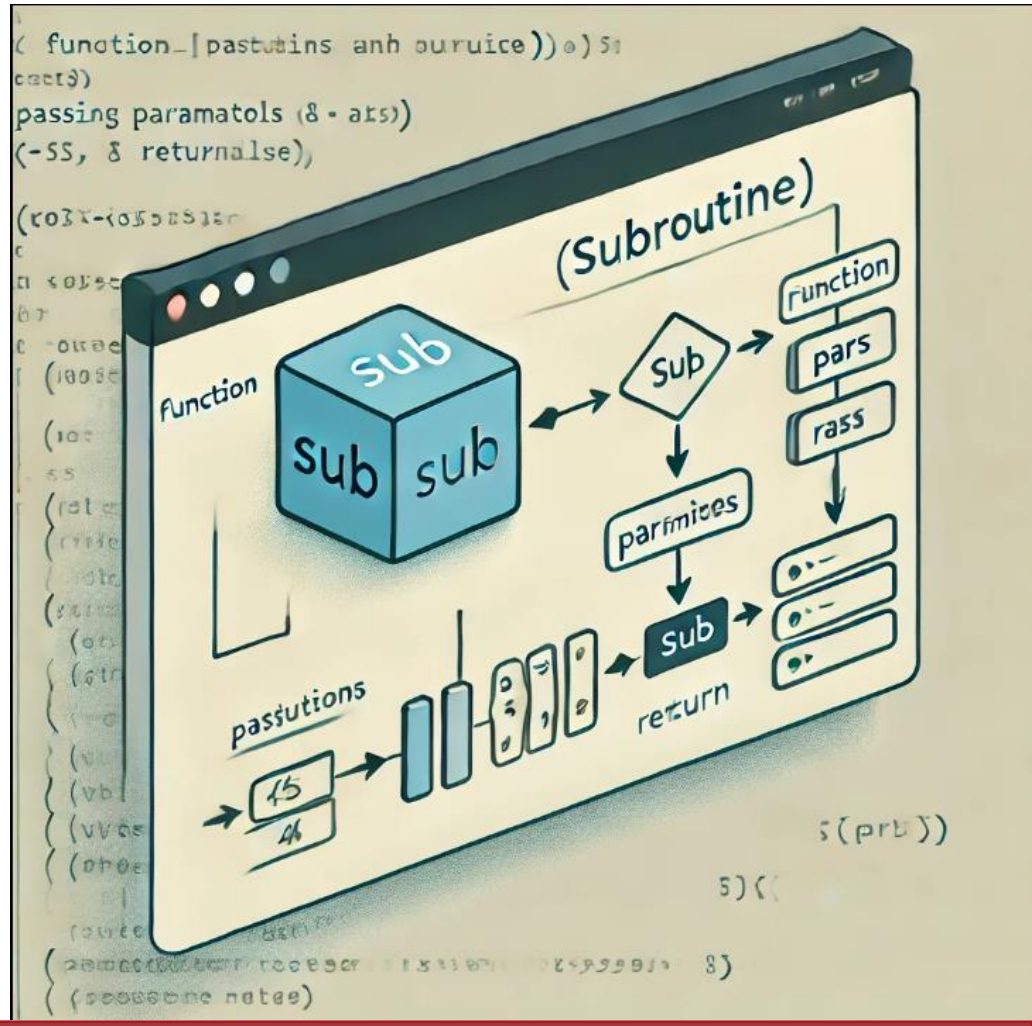
Thay vì viết lại cùng một đoạn mã nhiều lần, bạn có thể đặt nó vào một hàm và gọi nó khi cần.

Tổ chức mã

Hàm giúp tổ chức mã một cách rõ ràng, dễ đọc và dễ bảo trì.

Cải thiện khả năng đọc

Hàm làm cho mã dễ hiểu hơn bằng cách tách biệt các phần logic khác nhau.



Cú pháp khai báo hàm

Để khai báo hàm trong Perl, bạn sử dụng từ khóa sub theo sau là tên hàm và khối mã được bao bọc bởi dấu ngoặc nhọn {}.

1. **Khai báo hàm**
   sub ten_ham { ... }
2. **Thân hàm**
   Mã được thực hiện bởi hàm
3. **Kết thúc hàm**
   Dấu ngoặc nhọn đóng }

Ví dụ:

```
sub factorial {
    my ($n) = @_;
    if ($n == 0) {
        return 1;
    } else {
        return $n * factorial($n - 1);
    }
}
my $number = 5;
my $result = factorial($number);
print "Giai thừa của $number là: $result\n";
# Kết quả: Giai thừa của 5 là: 120
```

Các loại hàm trong Perl

Perl có hai loại hàm chính: hàm tích hợp sẵn (built-in) và hàm do người dùng định nghĩa (user-defined).

Haàm tích hợp sẵn

Được cung cấp sẵn bởi ngôn ngữ Perl.

1. print()
2. length()
3. substr()

Hàm do người dùng định nghĩa

Được tạo ra bởi lập trình viên để thực hiện các tác vụ cụ thể.

1. tinh_tong()
2. kiem_tra_so_nguyen()
3. xu_ly_du_lieu()

Phạm vi biến trong hàm

Phạm vi biến xác định nơi biến có thể được truy cập. Biến được khai báo trong hàm chỉ có thể được sử dụng trong hàm đó.

Biến toàn cục

```
$global_var = 10;
sub print_global {
    print "Giá trị của biến toàn cục:
$global_var\n";
}
print_global();
# Kết quả: Giá trị của biến toàn cục: 10
```

Biến cục bộ

```
sub local_variable_example {
    my $local_var = 20;  # Biến cục bộ
    print "Giá trị của biến cục bộ:
$local_var\n";
}
local_variable_example();
# Kết quả: Giá trị của biến cục bộ: 20
print $local_var;
# Lỗi: $local_var không tồn tại ngoài phạm vi hàm
```

Ví dụ:

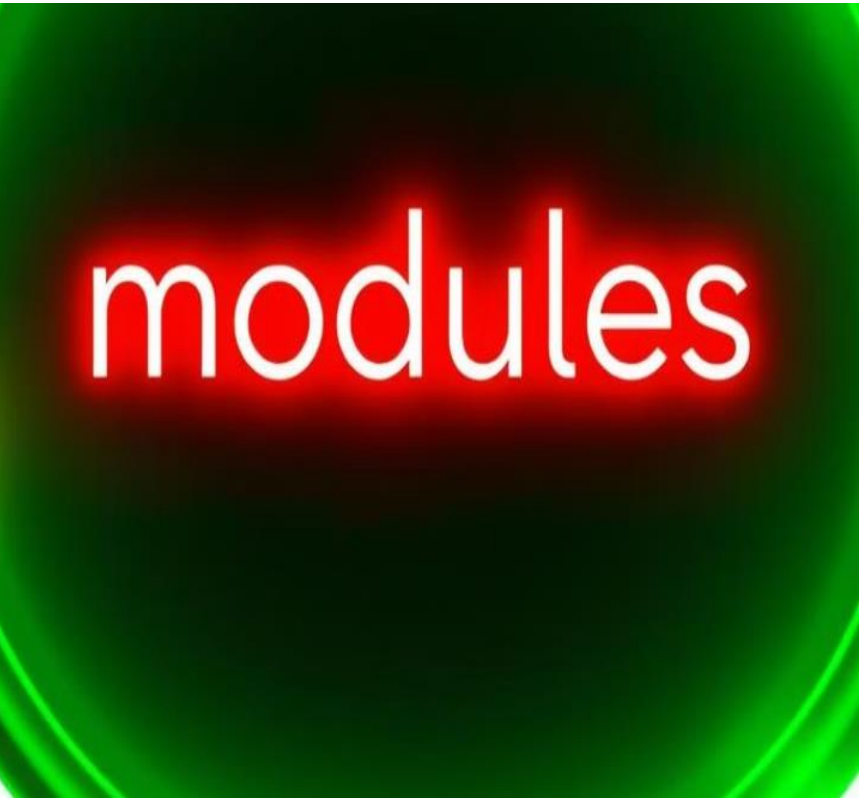
Kết hợp biến toàn cục và cục bộ


```
# Biến toàn cục
$var = 100;
sub test_variable_scope {
    my $var = 50;  # Biến cục bộ
    print "Giá trị biến cục bộ bên trong hàm: $var\n";
}
test_variable_scope();
# Kết quả: Giá trị biến cục bộ bên trong hàm: 50
print "Giá trị biến toàn cục ngoài hàm: $var\n";
# Kết quả: Giá trị biến toàn cục ngoài hàm: 100
```

Module in Perl

Module là một tập hợp các mã Perl được thiết kế để thực hiện một nhiệm vụ cụ thể, giúp tổ chức và tái sử dụng code hiệu quả hơn.

Lợi ích của việc sử dụng module



1 **Tăng năng suất**

Module giúp tái sử dụng code, tránh viết lại các chức năng đã có sẵn.

2 **Cải thiện tính tổ chức**

Code được chia nhỏ thành các module, dễ quản lý và bảo trì.

3 **Hỗ trợ cộng đồng**

Perl có một kho module phong phú do cộng đồng phát triển và chia sẻ.

4 **Nâng cao chất lượng code**

Module đã được kiểm tra và sử dụng rộng rãi, đảm bảo chất lượng cao hơn.

Các loại module phổ biến trong Perl

Module xử lý văn bản

Chẳng hạn như Text::CSV, xử lý file CSV, hoặc Regexp::Common, hỗ trợ tạo biểu thức chính quy phức tạp.

Module thao tác hệ thống

Ví dụ như File::Find, tìm kiếm file theo quy tắc, hoặc Sys::Hostname, lấy tên máy tính.

Module mạng

Bao gồm các module như Net::HTTP, thực hiện các yêu cầu HTTP, hoặc LWP, cung cấp giao diện để tương tác với web.

Cách tìm và cài đặt module

Sử dụng CPAN

CPAN (Comprehensive Perl Archive Network) là kho lưu trữ module lớn nhất cho Perl.

Cài đặt module

Sử dụng lệnh 'cpan' để cài đặt module từ CPAN.

Tìm kiếm module

Sử dụng công cụ tìm kiếm trên CPAN để tìm module phù hợp với nhu cầu.

Cách tạo và chia sẻ module của riêng bạn

1
Xác định mục đích
Hãy rõ ràng về chức năng của module bạn muốn tạo.

2
Thiết kế cấu trúc
Tổ chức code một cách logic và dễ hiểu.

3
Viết mã nguồn
Sử dụng các tiêu chuẩn mã hóa tốt nhất.

4
Kiểm tra và ghi chú
Đảm bảo code hoạt động chính xác và đầy đủ tài liệu.

5
Chia sẻ module
Đăng module lên CPAN để cộng đồng sử dụng.

Sử dụng module trong các dự án Perl

Import module

Sử dụng lệnh 'use' để import module vào chương trình.

Sử dụng hàm

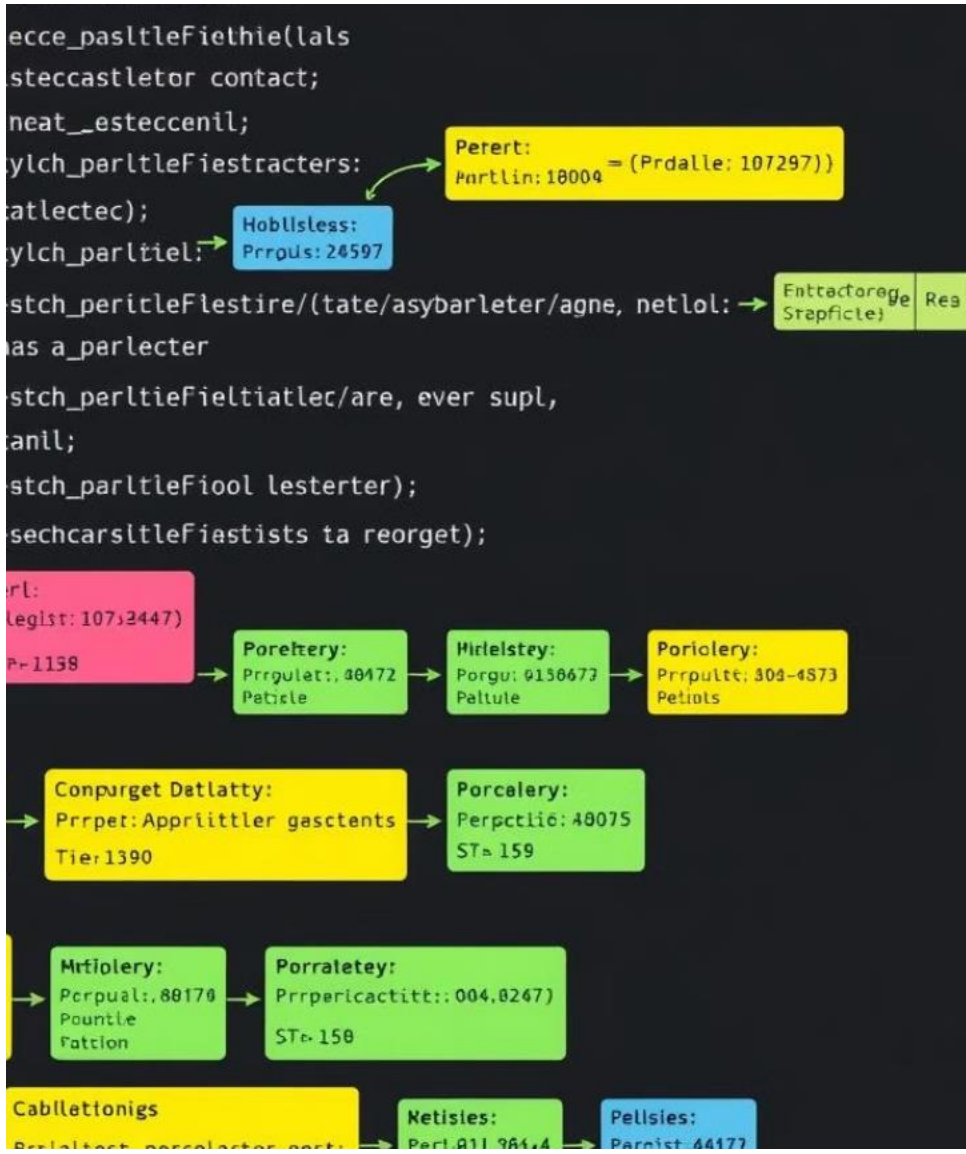
Gọi các hàm và biến được cung cấp bởi module.

Quản lý phụ thuộc

Đảm bảo các module cần thiết được cài đặt và quản lý.

Tái sử dụng code

Tận dụng lợi thế của các module đã có sẵn để tiết kiệm thời gian và công sức.



Quản lý phụ thuộc module

Công cụ quản lý phụ thuộc	Mô tả
CPAN	Kho lưu trữ module lớn nhất, hỗ trợ quản lý phụ thuộc.
cpanfile	File định nghĩa danh sách các module cần thiết cho dự án.
Module::Build	Công cụ xây dựng và quản lý module.

Ví dụ:

Bước 1. Tạo một module có tên MyModule.pm.

```
package MyModule;

# Hàm xuất hiện trong module
sub greet {
    my ($name) = @_;
    return "Hello, $name!";
}
# Trả về 1 để xác nhận module được tải thành công
Kết quả: 1;
```

Bước 2: Sử dụng module trong chương trình chính

```
# Tải module MyModule
use MyModule;

# Gọi hàm greet từ module MyModule
my $greeting = MyModule::greet("Đăng Trần");
print "$greeting\n";
# Kết quả: Hello, Đăng Trần!
```

Ví dụ:

Xuất khẩu function để sử dụng trực tiếp.

```
package MyModule;
use Exporter 'import';

# Xuất khẩu hàm greet
our @EXPORT = ('greet');

sub greet {
    my ($name) = @_;
    return "Hello, $name!";
}
```

```
use MyModule;

# Không cần MyModule:: để gọi hàm greet
my $greeting = greet("Đăng Trần");
print "$greeting\n";
# Kết quả: Hello, Đăng Trần!
```

Function in Python

Hàm là một khối mã có thể được sử dụng lại để thực hiện một nhiệm vụ cụ thể. Chúng đóng vai trò quan trọng trong việc tổ chức và đơn giản hóa mã, giúp cho việc lập trình Python trở nên hiệu quả hơn.

Cú pháp khai báo hàm

Để khai báo một hàm, bạn sử dụng từ khóa def theo sau là tên hàm và dấu ngoặc tròn. Nếu hàm có đối số, bạn khai báo chúng trong ngoặc tròn. Cấu trúc cơ bản của hàm là:

```
def ten_ham(doi_so1, doi_so2, ...):
```

Dòng mã sau từ khóa def là phần thân hàm, chứa các câu lệnh thực hiện nhiệm vụ của hàm.

Ví dụ hàm đơn giản

```
def in_choi():
 print("Xin chào!")
```

Ví dụ hàm có đối số

```
def tinh_tong(a, b):
 tong = a + b
 return tong
```

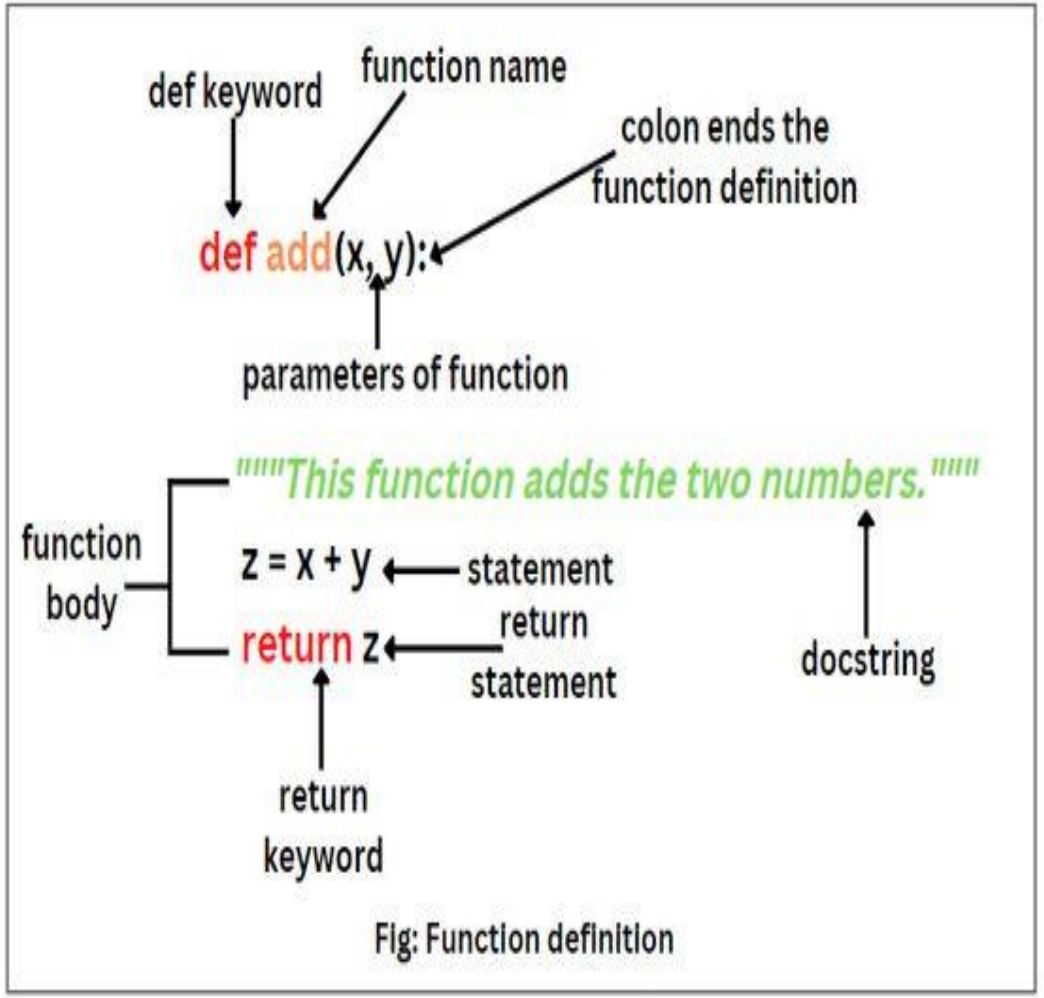


Fig: Function definition

Các loại đối số trong hàm

Hàm có thể nhận các đối số (arguments) để điều khiển hành vi của chúng. Có ba loại đối số chính:

1. **Đối số bắt buộc**

   Phải được truyền vào hàm khi gọi hàm, theo thứ tự đã định nghĩa.

2. **Đối số tùy chọn**

   Có giá trị mặc định, nếu không được truyền vào, hàm sẽ sử dụng giá trị mặc định.

3. **Đối số từ khóa**

   Được truyền vào theo tên, cho phép truyền các đối số theo bất kỳ thứ tự nào.

Trả về giá trị từ hàm

```
myFuunction();

aid  ==: ;

ateetu = sun;;
```

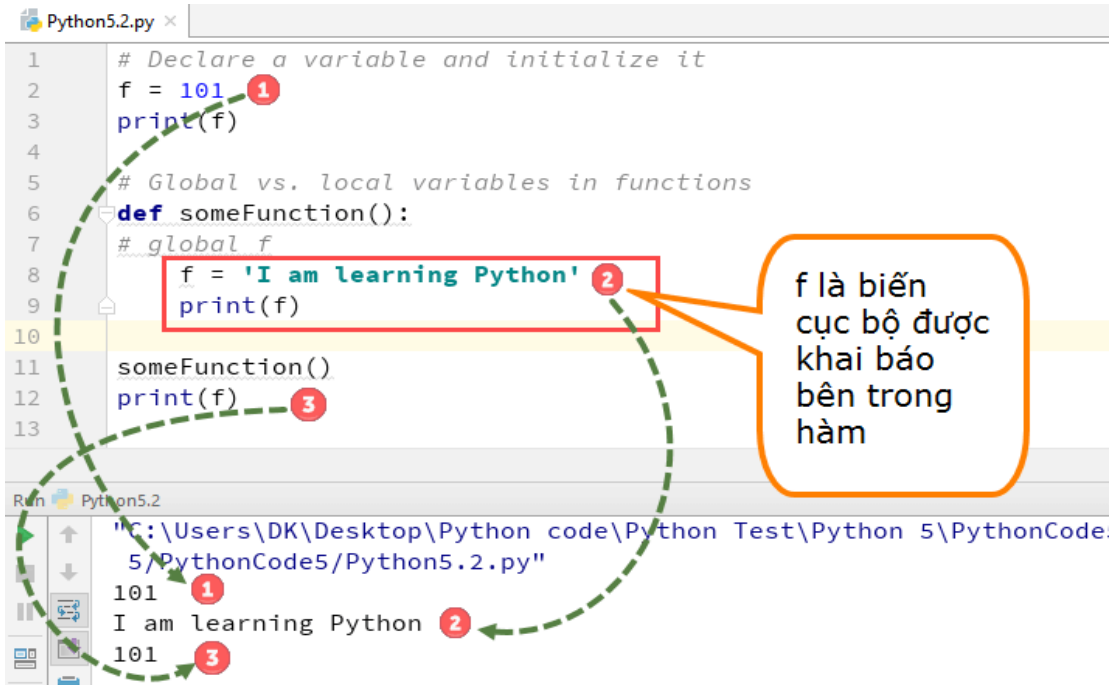
Hàm có thể trả về một giá trị sau khi hoàn thành nhiệm vụ. Từ khóa return được sử dụng để trả về giá trị.

Ví dụ:

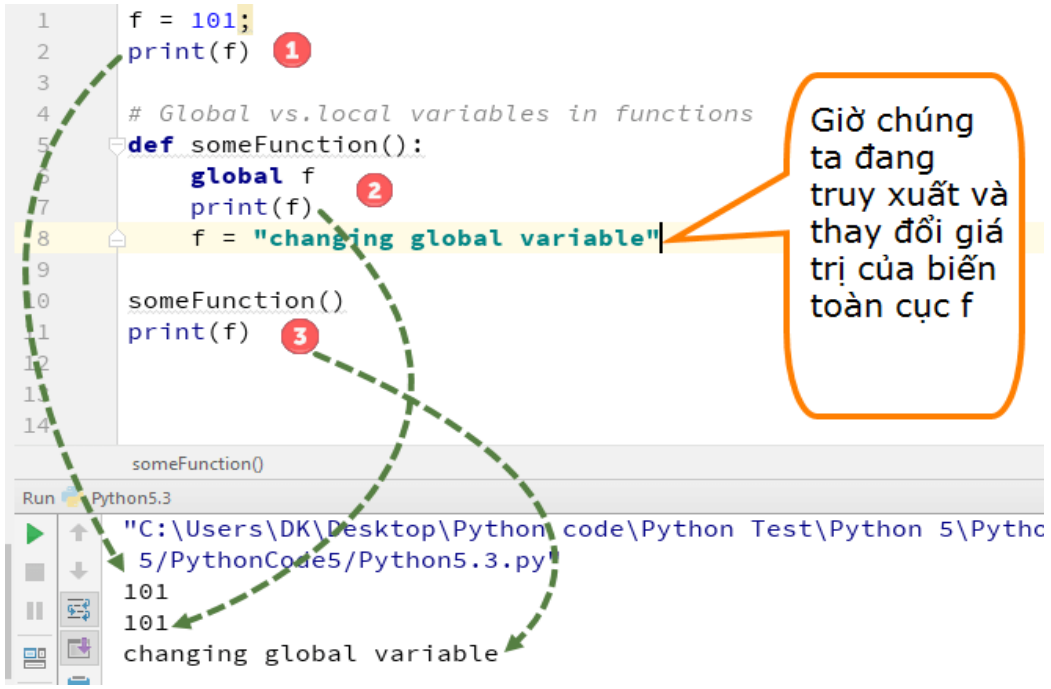
```
def tinh_tong(a, b):
 tong = a + b
 return tong
```

Hàm này nhận hai đối số a và b, tính tổng của chúng và trả về kết quả.

Phạm vi biến số trong hàm



Biến được khai báo bên trong hàm có phạm vi cục bộ (local scope), chỉ có thể truy cập được từ bên trong hàm đó.



Biến được khai báo bên ngoài hàm có phạm vi toàn cục (global scope), có thể truy cập được từ mọi nơi trong chương trình.

Ví dụ:

```
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)
```

Hàm lambda

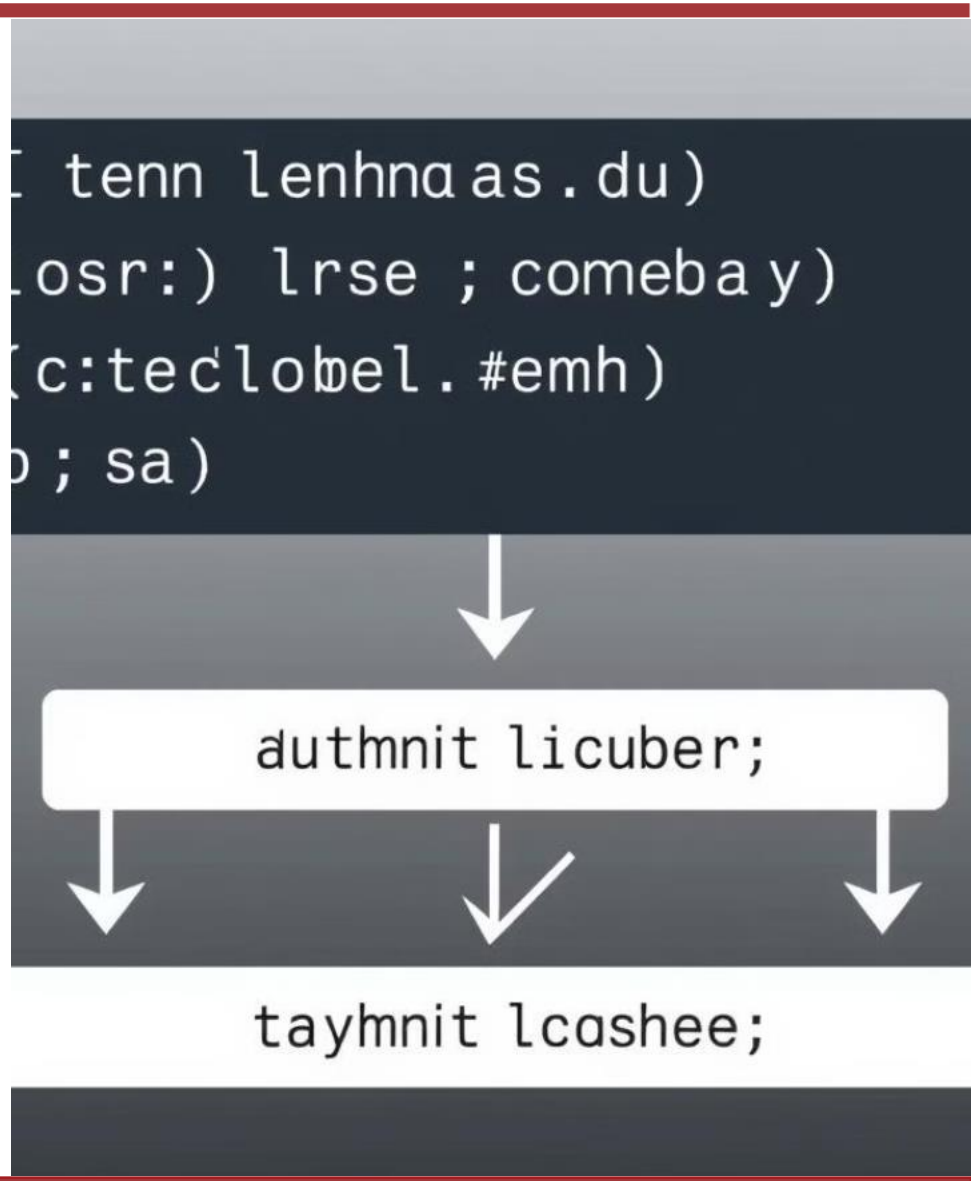
Hàm lambda là một loại hàm ẩn danh, được định nghĩa trong một dòng mã.

Cú pháp của hàm Lambda:

lambda arguments: expression

Trong đó:

- arguments là các tham số đầu vào của hàm Lambda.
- expression là biểu thức được thực hiện và trả về kết quả



Ví dụ:

```
# Hàm lambda để cộng hai số
add = lambda x, y: x + y
# Sử dụng hàm lambda
result = add(3, 5)
print(result)
```

Cú pháp của hàm Lambda như sau:

lambda arguments: expression

Trong đó:

- arguments là các tham số đầu vào của hàm Lambda.
- expression là biểu thức được thực hiện và trả về kết quả

```
# Sử dụng lambda với filter để lọc các số chẵn trong danh sách
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)
```

Lambda functions vs Traditional functions

Lambda functions

```
square = lambda x: x * x
print(square(4))
```

Traditional functions

```
def square(x):
    return x * x
print(square(4))
```

- **Hàm Lambda**: Được sử dụng khi cần một hàm ngắn gọn, thường chỉ chứa một biểu thức duy nhất.
- **Hàm truyền thống**: Thích hợp cho các trường hợp phức tạp hơn, có thể bao gồm nhiều biểu thức và câu lệnh, có tên để dễ dàng sử dụng lại trong nhiều nơi.

Module trong Python

1

Filename: **addition.py**

```
def sum(a,b):
    c=a+b
    print (c)
Return c
```

2

import addition

```
addition.sum(10,20)
addition.sum(30,40)
```

3

Filename: **area.py**

```
def square(l):
    print (l*l)
    return
def rectangle(l,b):
    print (l*b)
    return
def triangle(b,h):
    print (0.5*b*h)
    Return
```

4

from area import rectangle, triangle

```
rectangle(5,3)
triangle(7,2)
```

Trong Python, có thể sử dụng bất cứ source file nào dưới dạng như một Module bằng việc thực thi một lệnh import trong source file khác.

Cú pháp của lệnh import là:

import module1[, module2[,... moduleN]

Cách tạo và sử dụng module

Tạo module

Bạn có thể tạo một module bằng cách tạo một tệp Python (.py) với các định nghĩa và lệnh cần thiết.

Ví dụ: tạo một tệp tên là mymodule.py với nội dung sau:

```
# mymodule.py
def Ten(name):
    return f"Hello, {name}!“
def add(a, b):
    return a + b
```

Cách tạo và sử dụng module

Sử dụng module

Bạn có thể nhập và sử dụng module trong một tệp Python khác bằng cách sử dụng từ khóa import.

Ví dụ: tạo một tệp khác tên là main.py và nhập module mymodule:

 # main.py

```
import mymodule

print(mymodule.Ten("Minh Dang"))
# Output: Hello, Minh Dang!
print(mymodule.add(5, 3))
# Output: 8
```

Cách khác để nhập module.

Nhập toàn bộ module:

```
import math
print(math.sqrt(16))
# Output: 4.0
```

Nhập tất cả các thành phần từ module:

```
from math import *
print(sqrt(16))
# Output: 4.0
```

Nhập một phần của module:

```
from math import sqrt
print(sqrt(16))
# Output: 4.0
```

Đổi tên module khi nhập:

```
import math as m
print(m.sqrt(16))
# Output: 4.0
```

Module chuẩn của Python

Python đi kèm với một thư viện tiêu chuẩn lớn, chứa nhiều module hữu ích có thể được sử dụng mà không cần cài đặt thêm.

Ví dụ: **math**: Cung cấp các hàm toán học cơ bản.

datetime: Cung cấp các lớp để làm việc với ngày và giờ.

os: Cung cấp các hàm để tương tác với hệ điều hành.

```
import math
print(math.factorial(5))
```

```
import datetime
now = datetime.datetime.now()
print(now)
```

```
import os
print(os.getcwd())
```

Cài đặt và sử dụng module bên ngoài

Sử dụng pip, trình quản lý gói của Python.

Ví dụ:

Cài đặt và sử dụng module requests để thực hiện các yêu cầu HTTP:

```
# Cài đặt module:
pip install requests

# Sử dụng module:
import requests
response =
requests.get("https://api.github.com")
print(response.status_code)
```

Function In Perl & Python

Câu1: Xác thực địa chỉ IP

Viết một function **(bằng Perl và Python)** có tên validate_ip để kiểm tra xem một chuỗi đầu vào có phải là một địa chỉ IP hợp lệ hay không. Sử dụng biểu thức chính quy để kiểm tra định dạng địa chỉ IP (IPv4).

Yêu cầu:

- Input: Một chuỗi chứa địa chỉ IP.
- Output: Trả về 1 nếu địa chỉ IP hợp lệ, 0 nếu không hợp lệ.

Câu 2: Trích xuất tên miền từ URL

Viết một function **(bằng Perl và Python)** có tên extract_domain để trích xuất tên miền từ một URL. Sử dụng biểu thức chính quy để tìm và trích xuất phần tên miền (domain) từ URL.

Yêu cầu:
- Input: Một chuỗi chứa URL.
- Ouput: Chuỗi chứa tên miền.

Câu 3: Kiểm tra định dạng email

Viết một function **(bằng Perl và Python)** có tên validate_email để kiểm tra xem một chuỗi đầu vào có phải là email hợp lệ hay không. Sử dụng biểu thức chính quy để kiểm tra định dạng email.

Yêu cầu:
- Input: Một chuỗi chứa địa chỉ email.
- Output: Trả về 1 nếu email hợp lệ, 0 nếu không hợp lệ.

Câu 4: Kiểm tra URL có chứa giao thức HTTPS

Viết một function **(bằng Perl và Python)** có tên check_https để kiểm tra xem một URL có sử dụng giao thức HTTPS hay không. Sử dụng biểu thức chính quy để xác định xem URL bắt đầu bằng "https".

Yêu cầu:
- Input: Một chuỗi chứa URL.
- Output: Trả về 1 nếu URL sử dụng HTTPS, 0 nếu không.

Câu 5: Trích xuất các tham số truy vấn từ URL

Viết một function **(bằng Perl và Python)** có tên extract_query_params để trích xuất các tham số truy vấn từ một URL. Sử dụng biểu thức chính quy để tìm và trích xuất các cặp key=value từ phần truy vấn của URL.

Yêu cầu:
- Input: Một chuỗi chứa URL.
- Output: Một hash chứa các cặp key=value từ phần truy vấn.

References

[1] Lê Văn Cường, *Python cơ bản và ứng dụng*, NXB Đại học Quốc gia Hà Nội, 2019

[2] Randal L. Schwartz, Tom Phoenix & Brian D Foy (2008). *Learning Perl, 5th Edition*. O'Reilly Media.

[3] Mark Lutz (2013). *Learning Python, 5th Edition*. O'Reilly Media.Stein,

[4] Lincoln D. (2001). *Network Programming with Perl*. Addison-Wesley.

[5] Rhodes, Brandon & Goerzen, John (2014). *Foundations of Python Network Programming, 3rd Edition*. U.S.: Apress.

Website & Youtobe

[6] www.python.org

[7] https://perldoc.perl.org /

[8] https://www.youtube.com/c/TheNetNinja/playlists

[9] https://www.youtube.com/watch?v=T11QYVfZoD0

[10] https://www.youtube.com/watch?v=xbT7Pvh_5LQ