



TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ

CHƯƠNG 3

LẬP TRÌNH HÀM TRONG PYTHON

Nghe An, 2022

Chương 3:

LẬP TRÌNH HÀM TRONG PYTHON



NỘI DUNG GIẢNG DẠY:

- 3.1. Định nghĩa hàm trong Python
- 3.2. Các loại hàm trong Python
- 3.3. Tham số của hàm
- 3.4. Hàm vô danh
- 3.5. Các loại biến trong Python

Chương 3:

LẬP TRÌNH HÀM TRONG PYTHON



NỘI DUNG GIẢNG DẠY:

3.1. Định nghĩa hàm trong Python

3.2. Các loại hàm trong Python

3.3. Tham số của hàm

3.4. Hàm vô danh

3.5. Các loại biến trong Python

ĐỊNH NGHĨA HÀM TRONG PYTHON

- Trong Python, hàm là một nhóm các lệnh có liên quan đến nhau được dùng để thực hiện một tác vụ, nhiệm vụ cụ thể nào đó.
- Hàm giúp chia chương trình Python thành những khối/phần/mô-đun nhỏ hơn.
- Khi chương trình Python quá lớn, hoặc cần mở rộng, thì các hàm giúp chương trình có tổ chức và dễ quản lý hơn.
- Hàm còn có một tác dụng vô cùng quan trọng nữa là tránh việc phải lặp lại code để thực thi những tác vụ tương tự nhau, giúp code gọn hơn và có thể tái sử dụng.

ĐỊNH NGHĨA HÀM TRONG PYTHON

- **Cú pháp của hàm Python**

```
def ten_ham(các tham số/đối số):
```

```
    """Chuỗi văn bản để mô tả cho hàm (docstring)"""
```

```
    Các câu lệnh
```

Về cơ bản, một định nghĩa hàm Python sẽ bao gồm các thành phần sau:

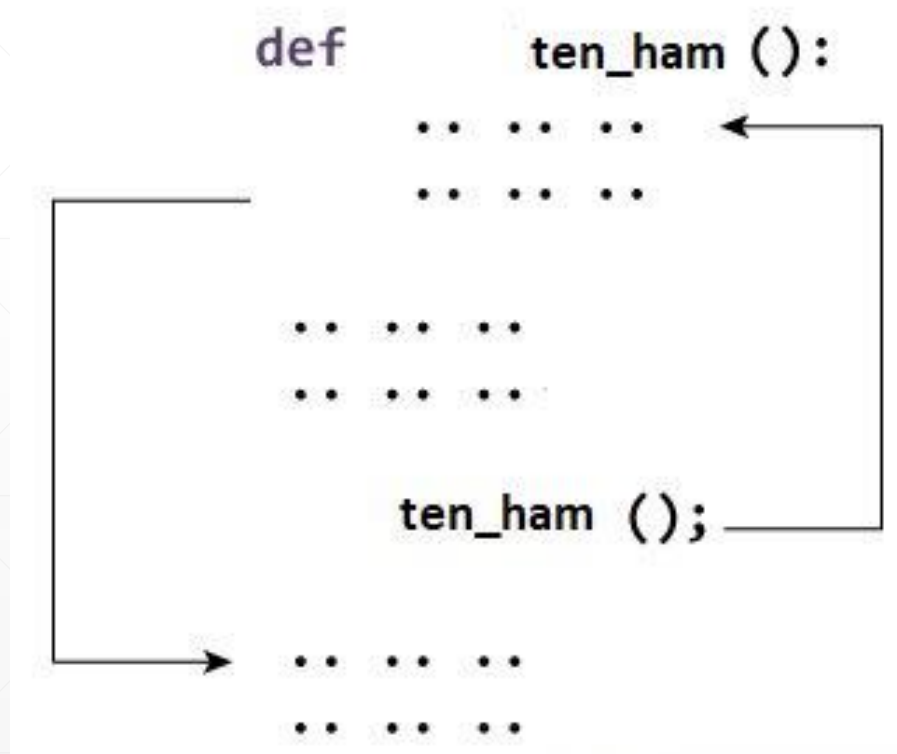
1. Từ khóa **def**: Đánh dấu sự bắt đầu của tiêu đề hàm.
2. **ten_ham**: Là định danh duy nhất dành cho hàm. Việc đặt tên hàm phải tuân thủ theo quy tắc viết tên và định danh trong Python.

ĐỊNH NGHĨA HÀM TRONG PYTHON

3. **Các tham số/đối số**: Chúng ta truyền giá trị cho hàm thông qua các tham số này. Chúng là tùy chọn.
4. **Dấu hai chấm (:)**: Đánh dấu sự kết thúc của tiêu đề hàm.
5. **docstring**: Chuỗi văn bản tùy chọn để mô tả chức năng của hàm.
6. **Các câu lệnh**: Một hoặc nhiều lệnh Python hợp lệ tạo thành khối lệnh. Các lệnh này phải có cùng một mức thụt đầu dòng (thường là 4 khoảng trắng).
7. Lệnh **return**: Lệnh này là tùy chọn, dùng khi cần trả về giá trị từ hàm.

ĐỊNH NGHĨA HÀM TRONG PYTHON

- Cách thức làm việc của hàm trong Python



ĐỊNH NGHĨA HÀM TRONG PYTHON

- Ví dụ về hàm Python

Dưới đây là một định nghĩa hàm đơn giản, gồm tên hàm, tham số của hàm, mô tả hàm và một câu lệnh:

```
def chao(ten):  
    """Hàm này dùng để  
    chào một người được truyền  
    vào như một tham số"""  
    print("Xin chào, " + ten + "!" )
```


ĐỊNH NGHĨA HÀM TRONG PYTHON

- **Gọi hàm trong Python**

Khi một hàm đã được định nghĩa, bạn có thể gọi nó từ một hàm khác, chương trình khác hoặc thậm chí tại dấu nhắc lệnh.

Để gọi hàm chúng ta chỉ cần nhập tên hàm với những tham số thích hợp là được.

Ví dụ để gọi hàm `chao()` vừa định nghĩa bên trên, ta gõ lệnh sau ngay tại dấu nhắc:

```
>>> chao ("Sinh vien lop Ky thuat lap trinh")
```

ĐỊNH NGHĨA HÀM TRONG PYTHON

- **Docstring trong Python**

Chuỗi đầu tiên ngay sau tiêu đề hàm được gọi là docstring (documentation string), nó được dùng để giải thích chức năng cho hàm.

Mặc dù docstring là không bắt buộc, nhưng việc giải thích ngắn gọn về chức năng của hàm sẽ giúp người dùng sau, thậm chí là bản thân người viết chương trình, khi gọi hàm có thể hiểu ngay hàm sẽ làm gì mà không cần phải tìm lại định nghĩa hàm để xem xét.

ĐỊNH NGHĨA HÀM TRONG PYTHON

- **Lệnh return trong hàm Python**

Lệnh return thường được dùng để thoát hàm và trở về nơi mà tại đó hàm được gọi.

Cú pháp của lệnh return:

```
return [danh_sach_bieu_thuc]
```

- Lệnh này có thể chứa biểu thức được tính toán và giá trị trả về.
- Nếu không có biểu thức nào trong câu lệnh hoặc không có lệnh return trong hàm thì hàm sẽ trả về None.

ĐỊNH NGHĨA HÀM TRONG PYTHON

- Ví dụ về lệnh return:

```
def gia_tri_tuyet_doi(so):  
    """Hàm này trả về giá trị tuyệt đối  
    của một số nhập vào"""  
    if so >= 0:  
        return so  
    else:  
        return -so
```

ĐỊNH NGHĨA HÀM TRONG PYTHON

- **Phạm vi và thời gian tồn tại của các biến**
 - Phạm vi của biến là đoạn chương trình mà ở đó biến được thừa nhận. Các tham số và biến được xác định bên trong một hàm không thể "nhìn thấy" từ bên ngoài. Do đó, những biến và tham số này chỉ có phạm vi trong hàm.
 - Thời gian tồn tại của biến là khoảng thời gian mà biến đó xuất hiện trong bộ nhớ. Khi hàm được thực thi thì biến sẽ tồn tại.
 - Biến bị hủy khi chúng ta thoát khỏi hàm. Hàm không nhớ giá trị của biến trong những lần gọi hàm trước đó.

ĐỊNH NGHĨA HÀM TRONG PYTHON

- Ví dụ:

```
def ham_in():  
    x = 15  
    print("Giá trị bên trong hàm:",x)  
x = 30  
ham_in()  
print("Giá trị bên ngoài hàm:",x)
```

Chương 3:

LẬP TRÌNH HÀM TRONG PYTHON



NỘI DUNG GIẢNG DẠY:

3.1. Định nghĩa hàm trong Python

3.2. Các loại hàm trong Python

3.3. Tham số của hàm

3.4. Hàm vô danh

3.5. Các loại biến trong Python

CÁC LOẠI HÀM TRONG PYTHON

Về cơ bản, Python có 2 loại hàm chính:

- Hàm được tích hợp sẵn trong Python: là các hàm có sẵn trong trình thông dịch của Python.
- Hàm do người dùng định nghĩa: định nghĩa để thực hiện một số công việc cụ thể.

CÁC LOẠI HÀM TRONG PYTHON

- **Hàm được tích hợp sẵn trong Python**

Trong phiên bản Python 3.6 có 68 hàm Python được tích hợp sẵn.

Hàm	Mô tả
abs()	Trả về giá trị tuyệt đối của một số
all()	Trả về True khi tất cả các phần tử trong iterable là đúng
any()	Kiểm tra bất kỳ phần tử nào của iterable là True
ascii()	Tả về string chứa đại diện (representation) có thể in
bin()	Chuyển đổi số nguyên sang chuỗi nhị phân
bool()	Chuyển một giá trị sang Boolean

CÁC LOẠI HÀM TRONG PYTHON

- **Hàm được tích hợp sẵn trong Python**

Hàm	Mô tả
<code>bytearray()</code>	Trả về mảng kích thước byte được cấp
<code>bytes()</code>	Trả về đối tượng byte không đổi
<code>callable()</code>	Kiểm tra xem đối tượng có thể gọi hay không
<code>chr()</code>	Trả về một ký tự (một chuỗi) từ Integer
<code>classmethod()</code>	Trả về một class method cho hàm
<code>compile()</code>	Trả về đối tượng code Python
<code>complex()</code>	Tạo một số phức

CÁC LOẠI HÀM TRONG PYTHON

- **Hàm được tích hợp sẵn trong Python**

Nếu muốn biết hàm này cụ thể làm gì, có đối số nào, chúng ta chỉ cần nhập lệnh:

```
print(ten_ham.__doc__)
```

→ Python sẽ giải thích khá đầy đủ về hàm.

CÁC LOẠI HÀM TRONG PYTHON

- **Hàm do người dùng định nghĩa:**

Việc định nghĩa hàm và gọi hàm đã được đề cập đến trong bài định nghĩa hàm Python (mục 3.1).

- Nếu ta sử dụng những hàm được người dùng khác viết dưới dạng thư viện, thì những hàm này gọi là hàm thư viện (library function). Như vậy, hàm ta tự định nghĩa có thể trở thành một hàm thư viện đối với người dùng nào đó.

CÁC LOẠI HÀM TRONG PYTHON

- **Ưu điểm khi sử dụng hàm Python do người dùng định nghĩa**
 - Hàm do người dùng định nghĩa giúp phân tích một chương trình lớn thành những phần nhỏ, khiến chương trình dễ hiểu, dễ duy trì và gỡ lỗi hơn.
 - Khi một đoạn code bị lặp lại trong chương trình, thì có thể sử dụng hàm để gom đoạn code này lại và chạy khi cần bằng cách gọi hàm.
 - Các lập trình viên cùng làm việc trong một dự án lớn, có thể phân chia công việc cho nhau bằng cách tạo các hàm khác nhau.

Chương 3:

LẬP TRÌNH HÀM TRONG PYTHON



NỘI DUNG GIẢNG DẠY:

3.1. Định nghĩa hàm trong Python

3.2. Các loại hàm trong Python

3.3. Tham số của hàm

3.4. Hàm vô danh

3.5. Các loại biến trong Python

THAM SỐ CỦA HÀM

Hàm có 0, 1 hoặc nhiều tham số. Ngăn cách nhau bởi dấu phẩy.
Tham số có 4 loại:

- Tham số bắt buộc
- Tham số có mặc định (Default parameter)
- Tham số có độ dài biến đổi (Variable-Length Parameter)
- Tham số từ khóa (Keyword Parameter)

THAM SỐ CỦA HÀM

- **Tham số bắt buộc**

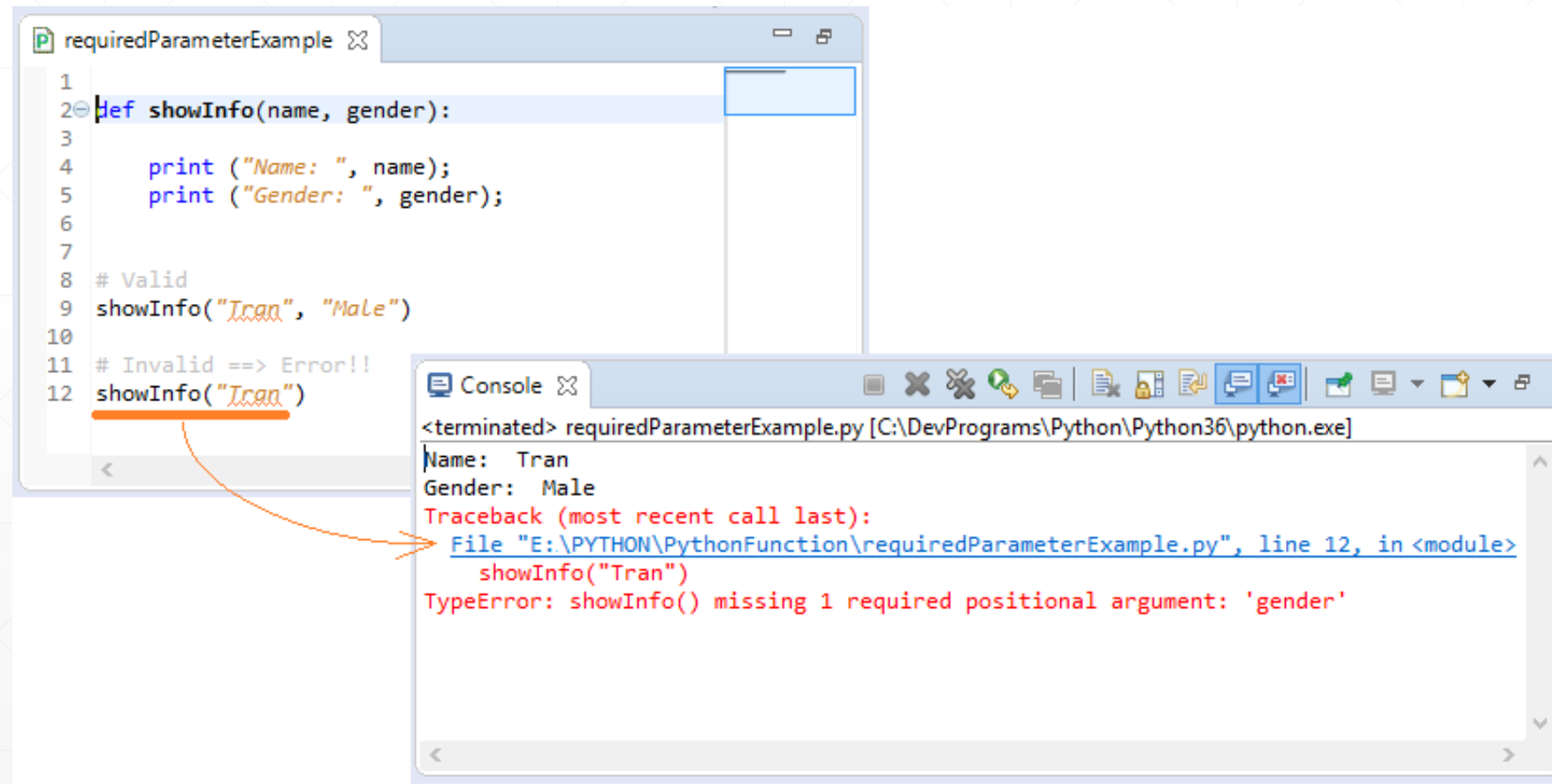
Ví dụ: Định nghĩa hàm showInfo, có 2 tham số, cả hai tham số này đều là bắt buộc.

- Khi gọi hàm này cần phải truyền 2 tham số vào cho hàm.
- Ngược lại chương trình sẽ xảy ra lỗi.

```
def showInfo(name, gender):  
    print ("Name: ", name);  
    print ("Gender: ", gender);
```


THAM SỐ CỦA HÀM

- Tham số bắt buộc

The image shows a screenshot of a Python IDE with two windows. The top window, titled 'requiredParameterExample', contains the following code:

```
1
2 def showInfo(name, gender):
3
4     print ("Name: ", name);
5     print ("Gender: ", gender);
6
7
8 # Valid
9 showInfo("Tran", "Male")
10
11 # Invalid ==> Error!!
12 showInfo("Tran")
```

Line 12 is highlighted with an orange underline. An orange arrow points from this line to the bottom window. The bottom window, titled 'Console', shows the output of the program:

```
<terminated> requiredParameterExample.py [C:\DevPrograms\Python\Python36\python.exe]
Name: Tran
Gender: Male
Traceback (most recent call last):
  File "E:\PYTHON\PythonFunction\requiredParameterExample.py", line 12, in <module>
    showInfo("Tran")
TypeError: showInfo() missing 1 required positional argument: 'gender'
```

THAM SỐ CỦA HÀM

- **Hàm với tham số mặc định**

- Hàm có thể có nhiều tham số, bao gồm các tham số bắt buộc và các tham số có giá trị mặc định.

- Ví dụ: Hàm `showInfo` có 3 tham số (`name`, `gender = "Male"`, `country = "US"`):

- `name` là tham số bắt buộc.
 - `gender` là tham số có giá trị mặc định `"Male"`.
 - `country` là tham số có giá trị mặc định `"US"`.

THAM SỐ CỦA HÀM

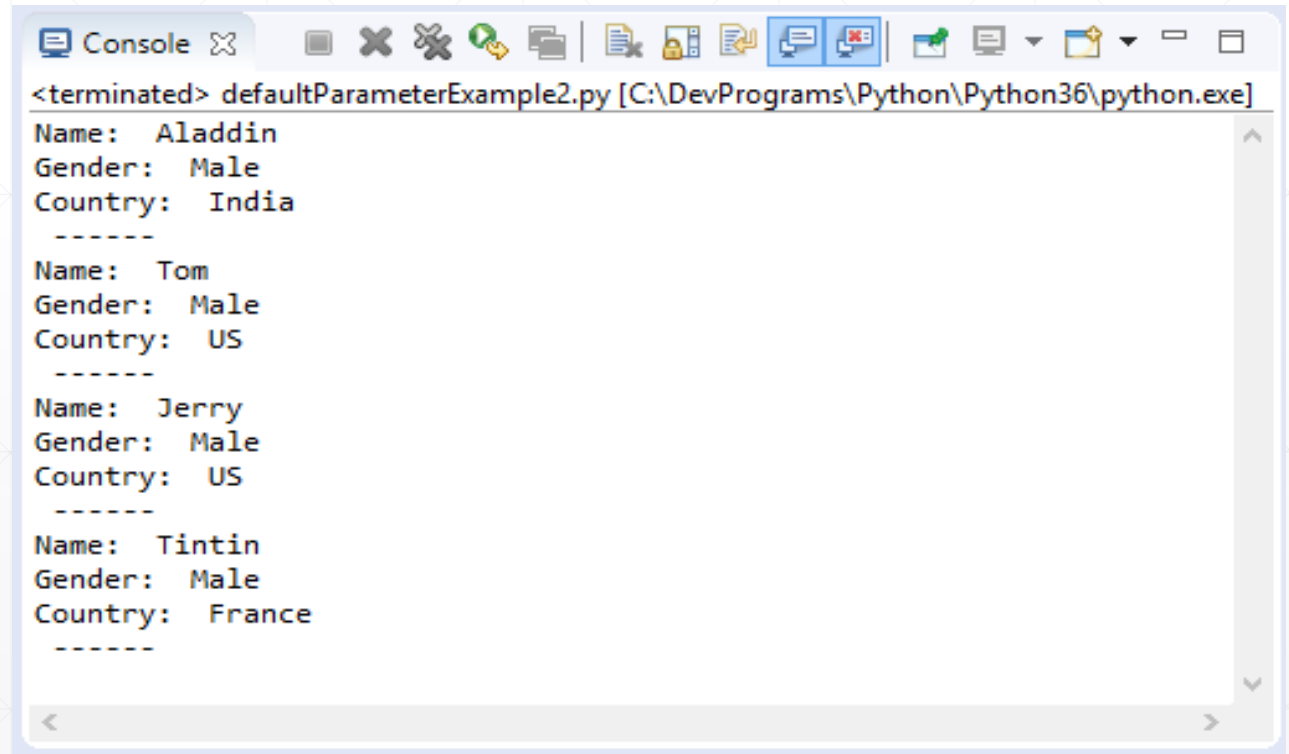
- **Hàm với tham số mặc định**

```
def showInfo(name, gender = "Male", country ="US"):
```

```
    print ("Name: ", name)
```

```
    print ("Gender: ", gender)
```

```
    print ("Country: ", country)
```

A screenshot of a Python console window showing the output of the showInfo function. The window title is "Console". The output shows three calls to the function with different names, all using the default gender "Male" and country "US". Each call is followed by a separator line of dashes.

```
<terminated> defaultParameterExample2.py [C:\DevPrograms\Python\Python36\python.exe]  
Name:  Aladdin  
Gender:  Male  
Country:  India  
-----  
Name:  Tom  
Gender:  Male  
Country:  US  
-----  
Name:  Jerry  
Gender:  Male  
Country:  US  
-----  
Name:  Tintin  
Gender:  Male  
Country:  France  
-----
```

THAM SỐ CỦA HÀM

- **Hàm có tham số với độ dài thay đổi**

Tham số với độ dài thay đổi (Variable-length Parameter) là một tham số đặc biệt. Khi gọi hàm, bạn có thể truyền (pass) 0, 1 hoặc nhiều giá trị ứng với tham số đó.

Chú ý: "Variable-length Parameter" luôn phải là tham số cuối cùng của hàm.

Ví dụ: Hàm `sumValues` có 3 tham số:

Tham số `a`, `b` là bắt buộc.

Tham số `*others` là "Variable-Length Parameter".

THAM SỐ CỦA HÀM

- **Hàm với tham số có độ dài thay đổi**

```
def sumValues(a, b, *others):
```

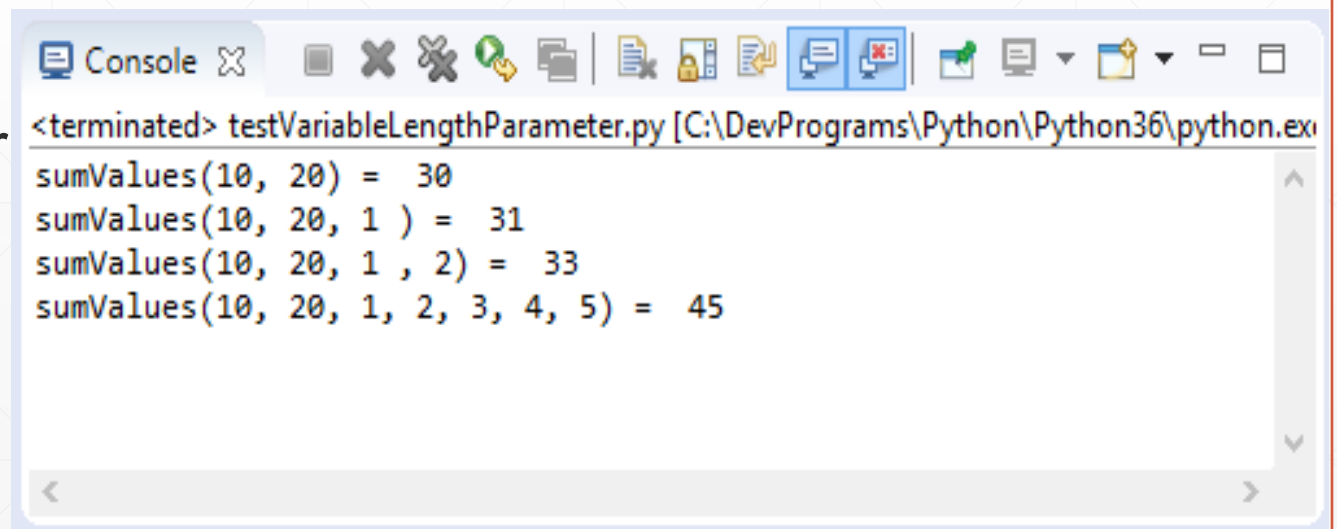
```
    retValue = a + b
```

```
    # Tham số 'others' giống như một mảng.
```

```
    for other in others :
```

```
        retValue = retValue + other
```

```
    return retValue
```

A screenshot of a Python console window titled "Console". The window shows the execution of a script named "testVariableLengthParameter.py". The output displays four function calls to "sumValues" with varying arguments, showing the cumulative sum of the arguments. The first call is "sumValues(10, 20) = 30", the second is "sumValues(10, 20, 1) = 31", the third is "sumValues(10, 20, 1, 2) = 33", and the fourth is "sumValues(10, 20, 1, 2, 3, 4, 5) = 45". The console window has a standard Windows-style title bar and a toolbar with various icons for file operations and debugging.

```
<terminated> testVariableLengthParameter.py [C:\DevPrograms\Python\Python36\python.exe]
sumValues(10, 20) = 30
sumValues(10, 20, 1 ) = 31
sumValues(10, 20, 1 , 2) = 33
sumValues(10, 20, 1, 2, 3, 4, 5) = 45
```

THẢO LUẬN NHÓM

NỘI DUNG:

1. Các phương thức truyền tham số giữa các hàm trong ngôn ngữ lập trình Python.
2. Viết hàm tìm giá trị lớn nhất của 3 số nhập vào từ bàn phím.
3. Viết hàm tìm giá trị nhỏ nhất của một mảng dữ liệu số cho trước.

BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 3.4; 3.5.
2. Tìm hiểu về các hàm trong lập trình Python.

Chương 3:

LẬP TRÌNH HÀM TRONG PYTHON



NỘI DUNG GIẢNG DẠY:

3.1. Định nghĩa hàm trong Python

3.2. Các loại hàm trong Python

3.3. Tham số của hàm

3.4. Hàm vô danh

3.5. Các loại biến trong Python

HÀM VÔ DANH

Các hàm được gọi là vô danh (**anonymous**) nếu chúng không được định nghĩa theo cách thông thường bởi từ khóa **def**, mà sử dụng từ khóa **lambda**.

- Hàm vô danh có thể có 0 hoặc nhiều tham số, nhưng trong thân hàm chỉ có duy nhất một biểu thức (expression). Giá trị của biểu thức chính là giá trị trả về của hàm. Nhưng không được sử dụng từ khóa 'return' ngay trước biểu thức.
- Danh sách các tham số cách nhau bởi dấu phẩy, và không được đặt trong cặp dấu ngoặc tròn ().

HÀM VÔ DANH

- Trong thân của hàm vô danh, chúng ta không thể truy cập các biến bên ngoài, chỉ có thể truy cập các tham số của nó.
- Hàm vô danh không thể gọi trực tiếp hàm print, bởi vì lambda đòi hỏi một biểu thức.

lambda tham_so: bieu_thuc

- Thường thì hàm Lambda được sử dụng khi cần một hàm vô danh trong thời gian ngắn, ví dụ như dùng làm đối số cho một hàm bậc cao hơn. Hàm Lambda thường được sử dụng cùng với các hàm Python tích hợp sẵn như filter() hay map(),...

HÀM VÔ DANH

Khai báo một biến: hello = một hàm nặc danh và không có tham số.

```
hello = lambda : "Hello"
```

Khai báo một biến: mySum = một hàm nặc danh có 2 tham số.

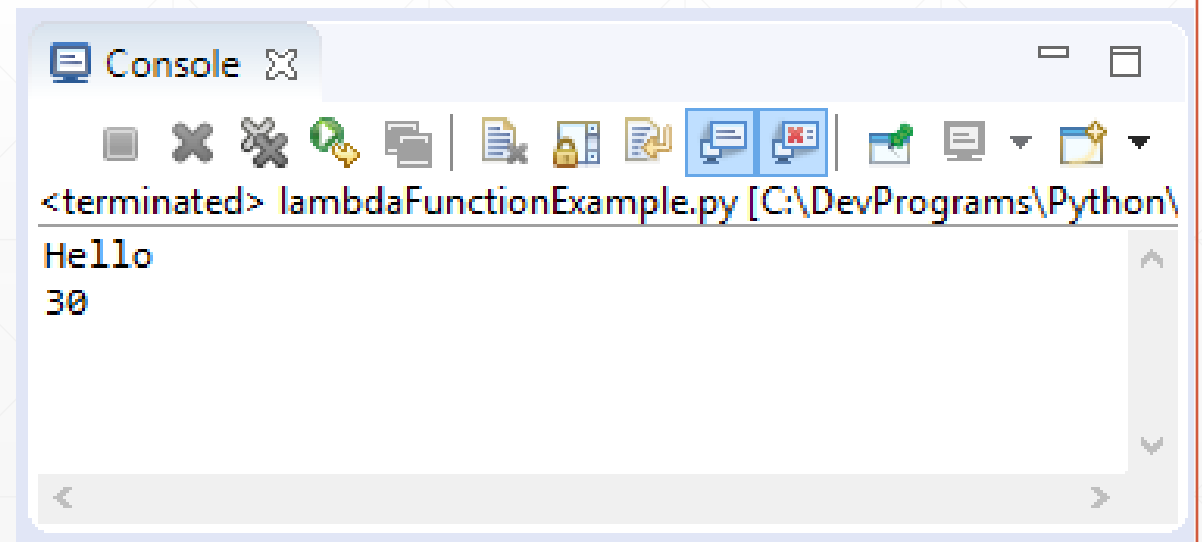
```
mySum = lambda a, b : a + b
```

```
a = hello()
```

```
print (a)
```

```
a = mySum(10, 20)
```

```
print (a)
```

A screenshot of a Python console window titled "Console". The window shows the execution of a script named "lambdaFunctionExample.py" located at "C:\DevPrograms\Python\". The output of the script is displayed as "Hello" followed by "30" on the next line. The console window includes standard Windows window controls (minimize, maximize, close) and a toolbar with various icons for file operations and debugging.

Chương 3:

LẬP TRÌNH HÀM TRONG PYTHON



NỘI DUNG GIẢNG DẠY:

3.1. Định nghĩa hàm trong Python

3.2. Các loại hàm trong Python

3.3. Tham số của hàm

3.4. Hàm vô danh

3.5. Các loại biến trong Python

CÁC LOẠI BIẾN TRONG PYTHON

Trong Python tồn tại các loại biến: biến toàn cục (global), biến cục bộ (local), biến nonlocal.

- **Biến toàn cục (global)**

- Trong ngôn ngữ lập trình Python, một biến được khai báo bên ngoài hàm hoặc trong phạm vi toàn cục được gọi là biến toàn cục hay biến global.
- Biến toàn cục có thể được truy cập từ bên trong hoặc bên ngoài hàm.

CÁC LOẠI BIẾN TRONG PYTHON

- **Biến toàn cục (global)**

```
x = "Biến toàn cục"    # Khai báo biến x
```

```
# Gọi x từ trong hàm vidu()
```

```
def vidu():
```

```
    print("x trong hàm vidu() :", x)
```

```
vidu()
```

```
# Gọi x ngoài hàm vidu()
```

```
print("x ngoài hàm vidu():", x)
```

CÁC LOẠI BIẾN TRONG PYTHON

- **Biến toàn cục (global)**

Chuyện gì sẽ xảy ra nếu chúng ta thay đổi giá trị của biến toàn cục trong hàm?

Ví dụ:

```
x = 2
```

```
def vidu():
```

```
    x=x*2
```

```
    print(x)
```

```
vidu()
```

CÁC LOẠI BIẾN TRONG PYTHON

- **Biến toàn cục (global)**

Nếu chạy code này sẽ nhận được thông báo lỗi:

UnboundLocalError: local variable 'x' referenced before assignment

- Lỗi này xuất hiện là do Python xử lý x như một biến cục bộ và x không được định nghĩa trong vidu().
- Để thay đổi biến toàn cục trong một hàm bạn sẽ phải sử dụng từ khóa global.

CÁC LOẠI BIẾN TRONG PYTHON

- **Biến cục bộ (local)**

Biến được khai báo bên trong một hàm hoặc trong phạm vi cục bộ được gọi là biến cục bộ hay biến local.

Ví dụ:

```
def vidu():  
    y = "Biến cục bộ"  
  
vidu()  
print(y)
```

CÁC LOẠI BIẾN TRONG PYTHON

- **Biến cục bộ (local)**

Khi chạy code trên chúng ta sẽ nhận được thông báo lỗi:

NameError: name 'y' is not defined

→ Lỗi này xuất hiện là do chúng ta đã cố truy cập vào biến cục bộ y trong phạm vi toàn cục, nhưng y chỉ làm việc trong hàm vidu() hoặc phạm vi cục bộ.

CÁC LOẠI BIẾN TRONG PYTHON

- **Biến cục bộ (local)**

Thông thường, để tạo một biến cục bộ, chúng ta sẽ khai báo nó trong một hàm như ví dụ dưới đây:

Ví dụ:

```
def vidu():  
    y = "Biến cục bộ"  
    print(y)  
vidu()
```

CÁC LOẠI BIẾN TRONG PYTHON

- **Biến nonlocal**

Trong Python, biến nonlocal được sử dụng trong hàm lồng nhau nơi mà phạm vi cục bộ không được định nghĩa.

→ Nói dễ hiểu thì biến nonlocal không phải biến local, không phải biến global, bạn khai báo một biến là nonlocal khi muốn sử dụng nó ở phạm vi rộng hơn local, nhưng chưa đến mức global.

Để khai báo biến nonlocal ta cần dùng đến từ khóa nonlocal.

CÁC LOẠI BIẾN TRONG PYTHON

- **Biến nonlocal**

Trong code trên có một hàm lồng là `ham_trong()`, ta dùng từ khóa `nonlocal` để tạo biến `nonlocal`.

Hàm `ham_trong()` được định nghĩa trong phạm vi của `hamngoai()`.

Lưu ý: Nếu chúng ta thay đổi giá trị của biến `nonlocal`, sự thay đổi sẽ xuất hiện trong biến cục bộ.

```
def hamngoai():  
    x = 10  
    def ham_trong():  
        nonlocal x  
        #x = 15  
        print(x)  
    ham_trong()  
    print(x)  
hamngoai()
```

BÀI TẬP

NỘI DUNG:

1. Viết một hàm số tính giai thừa của một số cho trước. Kết quả được in thành chuỗi trên một dòng, phân tách bởi dấu phẩy.
2. Viết hàm số chuyển chuỗi ký tự được nhập vào từ bàn phím từ chữ thường sang chữ hoa.

BÀI TẬP

CHUẨN BỊ CHO BUỔI HỌC TIẾP THEO:

1. Đọc các tài liệu về nội dung mục 4.1; 4.2; và 4.3.
2. Tìm hiểu về khái niệm kiểu dữ liệu có cấu trúc trong lập trình Python.
3. Các phương thức và các hàm xây dựng sẵn để xử lý String, List trong Python.