



LẬP TRÌNH PYTHON

HÀM TRONG PYTHON

GIỚI THIỆU VỀ HÀM

ĐỊNH NGHĨA VÀ THIẾT KẾ CHƯƠNG TRÌNH SỬ DỤNG HÀM

BIẾN CỤC BỘ

TRUYỀN ĐỐI SỐ VÀO HÀM

BIẾN TOÀN CỤC VÀ HẲNG SỐ TOÀN CỤC

GIỚI THIỆU VỀ HÀM TRẢ VỀ GIÁ TRỊ: TẠO SỐ NGẪU NHIÊN

VIẾT HÀM TRẢ VỀ GIÁ TRỊ

MÔ ĐUN TOÁN HỌC: *math*

LƯU TRỮ HÀM TRONG MÔ ĐUN

- **FUNCTION:** là tập hợp của các câu lệnh mà trong đó một chương trình sẽ thực hiện một nhiệm vụ cụ thể
- **Mô đun hóa chương trình:** một chương trình lớn mà trong đó, mỗi một hàm sẽ thực hiện đúng chức năng của mình. Hay còn được biết đến với khái niệm **chia để trị**.



This program is one long, complex sequence of statements.

In this program the task has been divided into smaller tasks, each of which is performed by a separate function.

```
def function4():
    statement
    statement
    statement
```

function

- **Lợi ích của việc mô đun hóa một chương trình với Hàm:**
 - + Code đơn giản hơn
 - + Tái sử dụng code: chỉ cần viết một lần và có thể sử dụng lại
 - + Kiểm tra và gỡ lỗi tốt hơn: có thể kiểm tra và gỡ lỗi độc lập với từng hàm
 - + Phát triển nhanh và đơn giản hơn
 - + Dễ dàng khi làm việc nhóm: mỗi nhóm có thể phát triển một hàm độc lập

- Hàm Void và hàm trả về giá trị:

- + Hàm void: chỉ thực thi câu lệnh mà nó chứa rồi kết thúc
- + Hàm trả về giá trị: thực thi câu lệnh mà nó chứa rồi trả về về giá trị cho câu lệnh đã gọi nó

Ví dụ: Hàm input() là một hàm trả về giá trị

Hai hàm dưới đây, hàm nào là hàm void, hàm nào là hàm trả về giá trị ?

```
def TinhTong(a, b):  
    tong = a + b  
    print(tong)
```

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

- Quy tắc đặt tên hàm:

- + Không được sử dụng từ khóa làm tên hàm
- + Không được chứa khoảng trắng
- + Ký tự đầu tiên phải là chữ cái hoặc dấu gạch dưới
- + Tất cả các ký tự khác phải là chữ cái, số hoặc dấu gạch dưới
- + Các ký tự viết hoa và viết thường là khác nhau

⇒ ***Tên của hàm nên được đặt đúng với chức năng của nó***

- Khởi tạo một hàm:

```
def function_name():  
    statement  
    statement
```

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


- Sử dụng *pass* keyword:

- + Sử dụng *pass* keyword để tạo một hàm rỗng
- + Điều này là cần thiết khi chỉ mới khởi tạo một chương trình, lúc này cần nghiên cứu thuật toán trước khi viết code

```
def function_name() :  
    pass
```

- Hàm *main*: được gọi khi chương trình bắt đầu
 - + Gọi đến các hàm khác khi cần sử dụng
 - + Xác định logic chính của chương trình

```
from datetime import date, datetime

def InRaNgayThang():
    today = date.today()
    d1 = today.strftime("%d/%m/%Y")
    print("Hom nay la ngay: ", d1)

def InRaThoiGian():
    now = datetime.now()
    print("Bay gio la: ", now.time())

if __name__ == "__main__":
    InRaThoiGian()
    InRaNgayThang()
```

- Thụt lề trong Python:

- + Các dòng trong khối phải bắt đầu bằng cùng số khoảng trắng
- + Sử dụng Tab hoặc Space để thụt lề các dòng trong khối, nhưng không được dùng cả 2 vì điều này có thể gây nhầm lẫn cho trình thông dịch Python

```
1  from datetime import date, datetime
2
3  def InRaNgayThang():
4      today = date.today()
5      d1 = today.strftime("%d/%m/%Y")
6      print("Hom nay la ngay: ", d1)
7
8
9
10
11
12  InRaThoiGian()
13  InRaNgayThang()
```

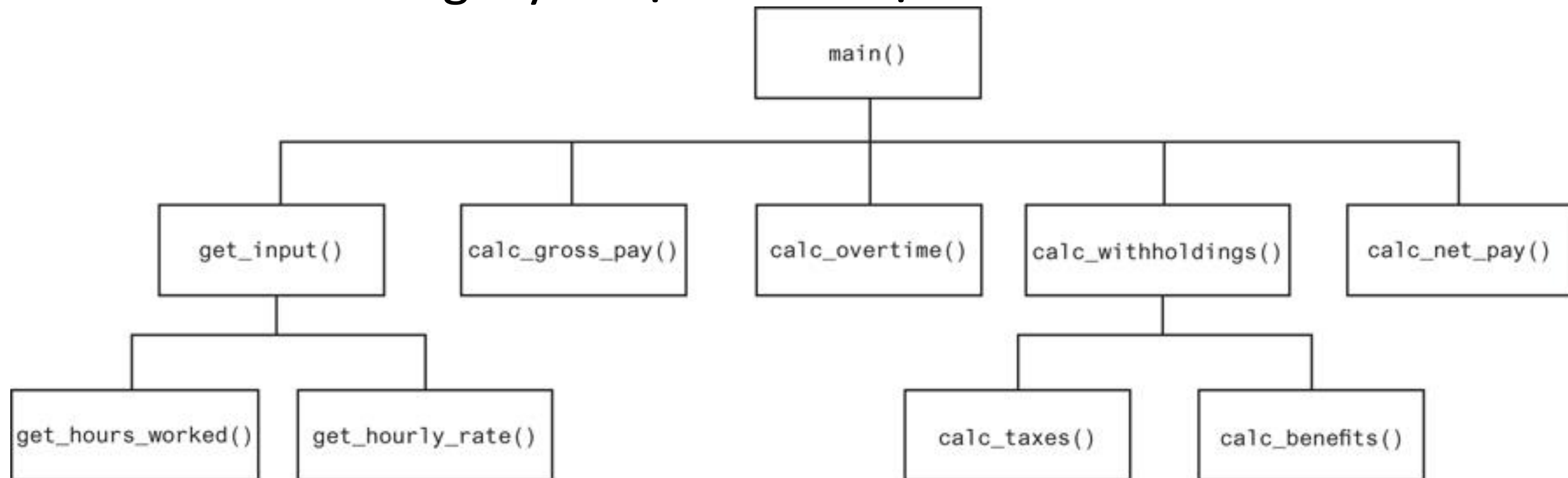
IndentationError: unindent does not match any outer indentation level
(file:///h%3A/Master/Design%20and%20Build%20Software/test.py, line 12) compile

View Problem (Alt+F8) No quick fixes available

Lỗi thụt lề: IndentationError

- Thiết kế chương trình sử dụng Hàm

- + Trong sơ đồ khối, lệnh gọi hàm được thể hiện bằng hình chữ nhật có các thanh dọc ở hai bên, tên hàm viết trong biểu tượng
- + Thường vẽ sơ đồ riêng cho mỗi hàm trong chương trình. Biểu tượng kết thúc thường ghi là Return
- + Thiết kế từ trên xuống: kỹ thuật chia thuật toán thành các hàm.



Một chương trình tính tiền lương thực nhận được xây dựng dựa trên rất nhiều hàm

- Local Variables (biến cục bộ)

Là biến được gán với một giá trị bên trong hàm: thuộc về hàm khi khởi tạo, và ***chỉ có khối lệnh bên trong hàm*** mới có thể truy cập.

=> Thử chạy chương trình dưới đây và đưa ra nhận xét

```
1  from datetime import date, datetime
2
3  def InRaNgayThang():
4      today = date.today()
5      d1 = today.strftime("%d/%m/%Y")
6      print("Hom nay la ngay: ", d1)
7
8  if __name__ == "__main__":
9      print(today)
```

- Local Variables (biến cục bộ)

- + Biến cục bộ không thể được truy cập bởi các câu lệnh bên trong hàm nếu chúng nằm trước khi biến được tạo
- + Các hàm khác nhau có thể có biến cục bộ cùng tên mà không gây nhầm lẫn.

```
1  from datetime import date, datetime
2
3  def InRaNgayThang():
4      today = date.today()
5      today = today.strftime("%d/%m/%Y")
6      print("Hom nay la ngay: ", today)
7
8  def InRaThoiGian():
9      today = datetime.now()
10     print("Bay gio la: ", today.time())
11
12  if __name__ == "__main__":
13     InRaThoiGian()
14     InRaNgayThang()
```

- Argument(đối số): dữ liệu được gửi vào hàm

- + Hàm có thể dùng đối số trong tính toán
- + Khi gọi hàm, đối số được đặt trong ngoặc đơn sau tên hàm
- + Khi truyền đối số vào hàm, nếu đầu vào nhiều đối số, cần truyền chính xác vị trí

```
1  from datetime import date, datetime
2
3  def TinhTong(a,b):
4      tong = a + b
5      return tong
6
7  if __name__ == "__main__":
8      tong = TinhTong(a=4, b=5)
9      print("Tong cua hai so la: ", tong)
```

- Argument(đối số): dữ liệu được gửi vào hàm

Nếu không biết trước được số lượng đối số từ khóa được truyền vào hàm, thêm dấu “**” trước tên

Ví dụ: đối số được truyền vào là một *dictionary*, các bạn sẽ được học sau này

```
def my_function(**kid):  
    print("His last name is " + kid["lname"])  
  
my_function(fname = "Tobias", lname = "Refsnes")
```


- **Argument(đối số):** dữ liệu được gửi vào hàm:

Ví dụ: Chương trình thay đổi tham số

```
def main():  
    value = 99  
    print(f'The value is {value}.')  
    change_me(value)  
    print(f'Back in main the value is {value}.')
```

```
def change_me(arg):  
    print('I am changing the value.')    arg = 0  
    print(f'Now the value is {arg}.')
```

*Nhận xét kết quả trả về của **value** và **arg***

- Global variables (biến toàn cục)

- + Được tạo bằng lệnh gán bên ngoài tất cả các hàm. Có thể được truy cập bởi bất kỳ lệnh nào trong chương trình, kể cả bên trong hàm
- + Nếu một hàm cần gán giá trị cho biến toàn cục, biến đó phải được khai báo lại trong hàm. Cú pháp: `global ten_bien`

```
1  from datetime import date, datetime
2
3  global a
4
5  def TinhTong(a, b):
6      tong = a + b
7      return tong
8
9  def TinhTich(b):
10     return a*b
11
12  if __name__ == "__main__":
13     a = 4
14     tong = TinhTong(a, 5)
15     print("Tong của hai số là: ", tong)
16     tich = TinhTich(5)
17     print("Tich của hai số là: ", tich)
```

```
x = "Toi rat dep trai"

def myfunc():
    global x
    x = "Toi cuc ky dep trai"

myfunc()

print(x)
```

- **Global variables (biến toàn cục)**

Lý do tránh sử dụng biến toàn cục:

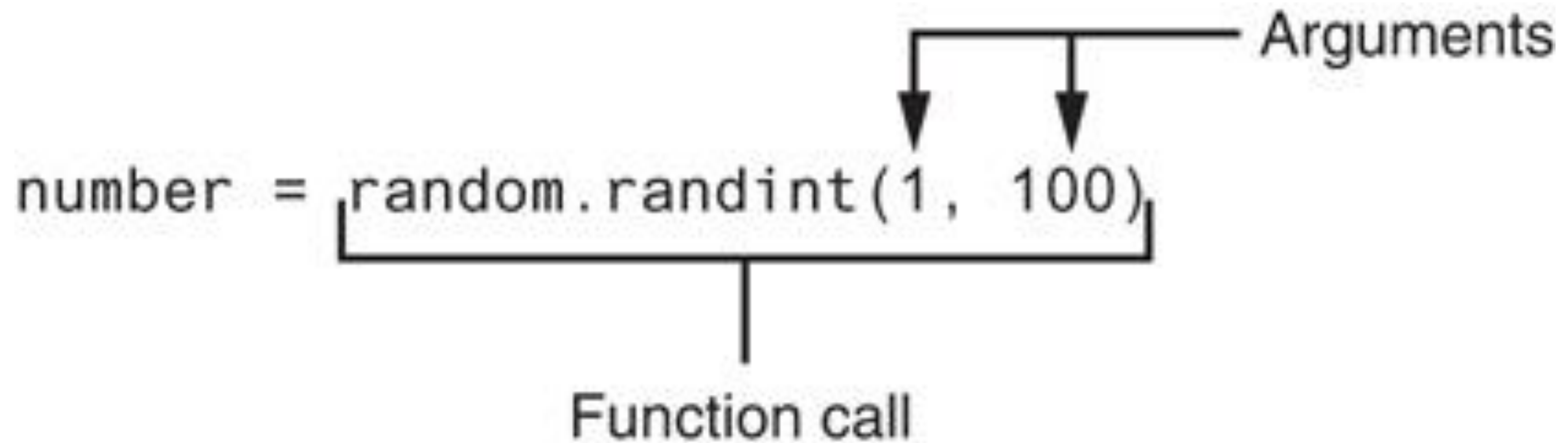
- + Biến toàn cục làm khó khăn trong việc gỡ lỗi
- + Nhiều vị trí trong mã có thể gây ra giá trị biến sai
- + Hàm sử dụng biến toàn cục thường phụ thuộc vào biến đó, làm khó khăn khi chuyển hàm sang chương trình khác.
- + Biến toàn cục làm chương trình khó hiểu

- **Global constants (hằng số toàn cục)**
 - + Tên toàn cục tham chiếu giá trị không thể thay đổi được gọi là hằng số toàn cục.
 - + Có thể sử dụng hằng số toàn cục trong chương trình.
 - + Để giả lập hằng số toàn cục trong Python, tạo biến toàn cục và không khai báo lại nó trong các hàm.

- Tạo số ngẫu nhiên:

- + Số ngẫu nhiên hữu ích trong nhiều tác vụ lập trình.
- + Module `random`: bao gồm các hàm thư viện để làm việc với số ngẫu nhiên.
- + Cú pháp dấu chấm: cú pháp để gọi hàm thuộc về module.

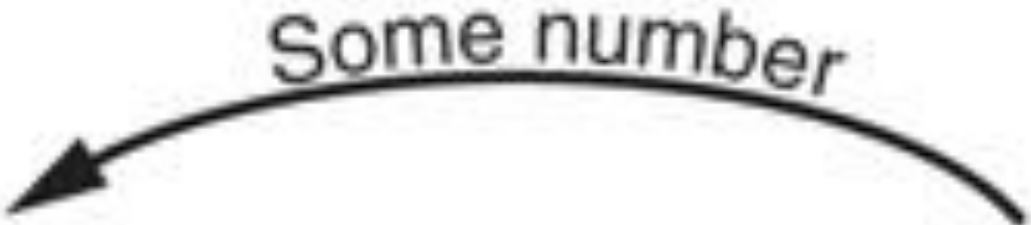
Định dạng: `module_name.function_name()`



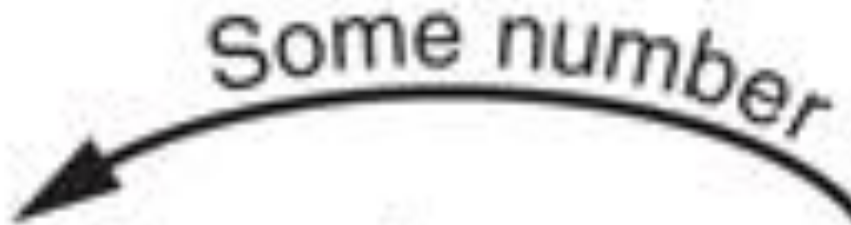
THIỆU VỀ HÀM TRẢ VỀ GIÁ TRỊ: TẠO SỐ NGẪU NHIÊN

- Tạo số ngẫu nhiên:

Hàm random trả về một giá trị:

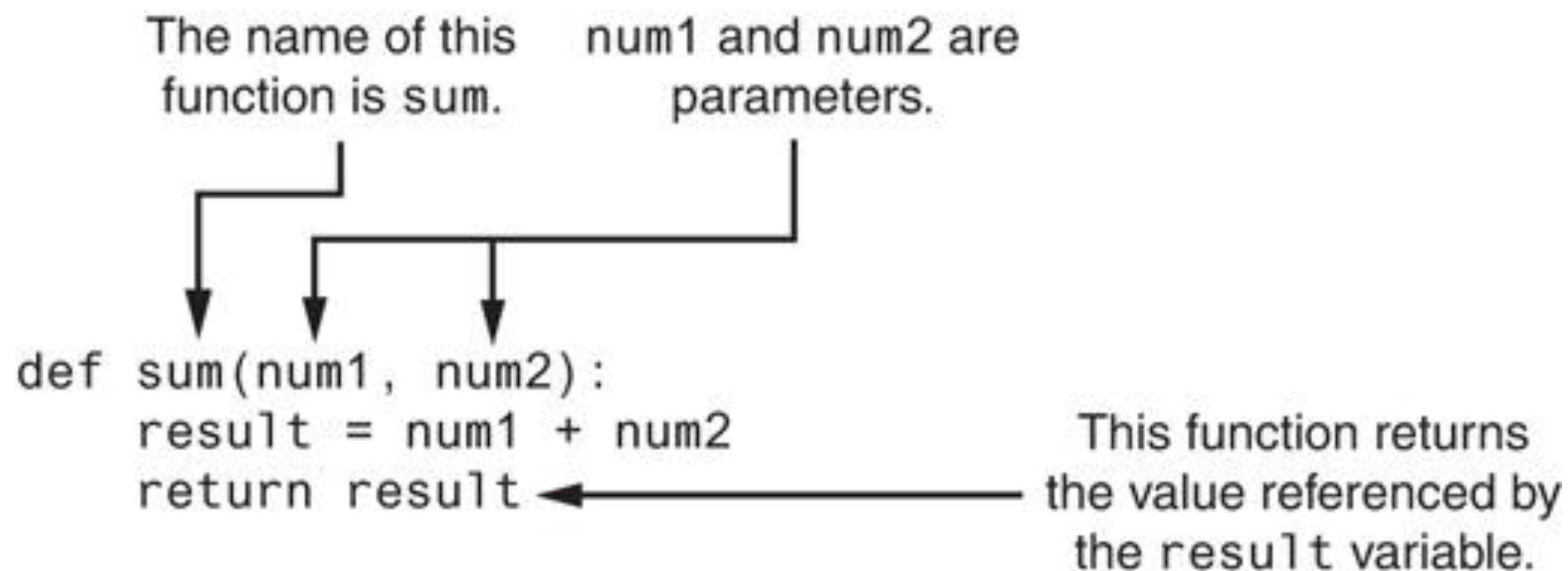

`number = random.randint(1, 100)`

A random number in the range of 1 through 100 will be assigned to the number variable.


`print(random.randint(1, 10))`

A random number in the range of 1 through 10 will be displayed.

- Hàm có thể trả về một, hai hoặc nhiều giá trị
- Các kiểu giá trị trả về có thể khác nhau: *int, float, string, boolean, array, ...*



- Module ``math``: là một phần của thư viện chuẩn chứa các hàm hữu ích cho phép thực hiện các phép toán toán học.
- Thường nhận một hoặc nhiều giá trị làm đối số, thực hiện phép toán và trả về kết quả.
- Sử dụng module cần có lệnh ``import math``.

Module Function	Description
<code>acos (x)</code>	Returns the arc cosine of x , in radians.
<code>asin (x)</code>	Returns the arc sine of x , in radians.
<code>atan (x)</code>	Returns the arc tangent of x , in radians.
<code>ceil (x)</code>	Returns the smallest integer that is greater than or equal to x .
<code>cos (x)</code>	Returns the cosine of x in radians.
<code>degrees (x)</code>	Assuming x is an angle in radians, the function returns the angle converted to degrees.
<code>exp (x)</code>	Returns e^x
<code>floor (x)</code>	Returns the largest integer that is less than or equal to x .
<code>hypot (x, y)</code>	Returns the length of a hypotenuse that extends from $(0, 0)$ to (x, y) .
<code>log (x)</code>	Returns the natural logarithm of x .
<code>log10 (x)</code>	Returns the base-10 logarithm of x .
<code>radians (x)</code>	Assuming x is an angle in degrees, the function returns the angle converted to radians.
<code>sin (x)</code>	Returns the sine of x in radians.
<code>sqrt (x)</code>	Returns the square root of x .
<code>tan (x)</code>	Returns the tangent of x in radians.

- Module là một tệp chứa mã Python.
 - + Chứa định nghĩa hàm nhưng không chứa các lệnh gọi hàm.
 - + Các chương trình nhập khẩu sẽ gọi các hàm.
 - Quy tắc đặt tên module:
 - + Tên tệp phải kết thúc bằng `.py`
 - + Không được trùng với từ khóa Python
- Nhập khẩu module bằng lệnh `import`.

Chương này đã đề cập đến:

- Hàm, bao gồm:
 - + Giới thiệu chung về Hàm
 - + Khởi tạo, gọi Hàm
 - + Thiết kế một chương trình sử dụng Hàm
 - + Biến cục bộ, biến toàn cục, hằng số toàn cục
 - + Truyền đối số vào Hàm
 - + Viết chương trình sinh số ngẫu nhiên
 - + Giới thiệu về mô đun *math*



Cảm ơn đã lắng nghe!