

SAU BÀI NÀY EM SẼ:

- Biết được chương trình con là hàm.
- Biết cách tạo hàm.



Các chương trình giải những bài toán thực tế phức tạp thường có rất nhiều dòng lệnh, trong đó có không ít những khối lệnh tương ứng với một số thao tác được lặp đi lặp lại nhiều lần ở những vị trí khác nhau. Để đỡ công viết đi viết lại các khối lệnh đó, trong tổ chức chương trình viết bằng ngôn ngữ lập trình bậc cao, người ta thường gom các khối lệnh như vậy thành những chương trình con. Khi đó, trong chương trình người ta chỉ cần thay cả khối lệnh bằng một lệnh gọi chương trình con tương ứng. Trong Python, các hàm chính là các chương trình con.

Em có thể kể tên một số hàm trong số các lệnh đã học hay không? Các hàm đó có những đặc điểm chung gì?

## 1. MỘT SỐ HÀM THIẾT KẾ SẴN CỦA PYTHON

### Hoạt động 1 Tìm hiểu một số hàm của Python

Quan sát một số câu lệnh trong Bảng 26.1 và cho biết những câu lệnh này có điểm chung gì.

**Bảng 26.1.** Một số lệnh trong Python

<code>abs()</code>	<code>len()</code>	<code>range()</code>	<code>bool()</code>	<code>float()</code>
<code>list()</code>	<code>round()</code>	<code>chr()</code>	<code>input()</code>	<code>ord()</code>
<code>str()</code>	<code>divmod()</code>	<code>int()</code>	<code>print()</code>	<code>type()</code>



Trước tiên, về hình thức, em có thể thấy các lệnh trên đều có các dấu mở đóng ngoặc đi sau tên lệnh. Tiếp đó, khi viết trong chương trình, bên trong các dấu ngoặc, nói chung, em có thể cần ghi thêm các tham số là các đại lượng, các biến hoặc thậm chí, trong một số trường hợp, cả các biểu thức. Ví dụ:

- Lệnh `print("Thời khoá biểu")` thực hiện việc in xâu kí tự "Thời khoá biểu" trong dấu ngoặc ra màn hình.
- Lệnh `x = int("52")` chuyển xâu "52" thành số nguyên 52.
- Lệnh `type(y)` trả lại kiểu dữ liệu của biến y.

Trong một số trường hợp bên trong dấu ngoặc có thể bỏ trống. Ví dụ, lệnh `x = input()` thực hiện yêu cầu nhập vào một xâu kí tự bất kì và gán cho biến x.

Các lệnh trong Bảng 26.1 chính là các chương trình con được thiết kế sẵn của Python, cho phép người dùng tùy ý sử dụng trong các chương trình của riêng mình.

Trong các ví dụ trên, dấu ngoặc nhọn bên trong ngoặc của các hàm `int()` và `print()` là **tham số của hàm**. Cú pháp câu lệnh gọi hàm trong Python có dạng chung như sau:

```
<tên hàm>(<danh sách tham số hàm>)
```

Python cung cấp sẵn nhiều hàm thực hiện những công việc khác nhau cho phép người dùng được tùy ý sử dụng khi viết chương trình bằng các câu lệnh gọi hàm tương ứng.



Mô tả tham số và giá trị trả lại của mỗi hàm sau: `float()`, `str()`, `len()`, `list()`.

## 2. THIẾT LẬP CÁC HÀM TỰ ĐỊNH NGHĨA



Ngoài các hàm thiết kế sẵn, Python còn cho phép người dùng tự thiết lập các hàm của riêng mình (còn gọi là các hàm tự định nghĩa).

### Hoạt động 2 Cách thiết lập hàm trong Python

Quan sát các ví dụ sau để biết cách viết hàm.



**Ví dụ 1.** Cách viết hàm có trả lại giá trị.

```
>>> def inc(n):  
    return n+1
```

```
>>> inc(3)
```

```
4
```

Tên hàm: `inc`.  
Tham số hàm: số `n`.  
Giá trị trả lại: số `n + 1`.

**Ví dụ 2.** Cách viết hàm không trả lại giá trị.

```
>>> def thong_bao(msg):  
    print("Xin chào bạn", msg)  
    return
```

```
>>> thong_bao("Trần Quang Minh")
```

```
Xin chào bạn Trần Quang Minh
```

Tên hàm: `thong_bao`.  
Tham số hàm: dấu ngoặc ngoặc `msg`.  
Giá trị trả lại: không có.

Hàm trong Python được định nghĩa bằng từ khóa `def`, theo sau là tên hàm (tên hàm sẽ theo quy tắc đặt tên định danh). Hàm có thể có hoặc không có tham số. Khối lệnh mô tả hàm được viết sau dấu `:` và viết lùi vào, thẳng hàng. Hàm có thể có hoặc không có giá trị trả lại sau từ khóa `return`.

Cú pháp thiết lập hàm có trả lại giá trị.

```
def <tên hàm>(<tham số>):  
    <khối lệnh>  
    return <giá trị>
```

Cần có lệnh **return** <giá trị>.  
Hàm sẽ kết thúc khi gặp lệnh **return**  
và trả lại <giá trị>.

Cú pháp thiết lập hàm không trả lại giá trị.

```
def <tên hàm>(<tham số>):  
    <khối lệnh>  
    return
```

Lệnh **return** không có giá trị trả lại.  
Hàm kết thúc khi gặp lệnh **return**.  
Nếu hàm không trả lại giá trị thì có  
thể không cần lệnh **return**.

Để thiết lập hàm trả lại giá trị, câu lệnh **return** trong khai báo hàm cần có <giá trị> đi kèm. Để thiết lập hàm không trả lại giá trị có thể dùng lệnh **return** không có <giá trị> hoặc không cần có **return**.



Quan sát các hàm sau, giải thích cách thiết lập và chức năng của mỗi hàm.

a)

```
def Nhap_xau():  
    msg = input("Nhập một xâu: ")  
    return msg
```

b)

```
def Inday(n):  
    for k in range(n):  
        print(k, end = " ")
```



## THỰC HÀNH

Thiết lập hàm trong Python.

**Nhiệm vụ 1.** Viết hàm yêu cầu người dùng nhập họ tên rồi đưa lời chào ra màn hình.

**Hướng dẫn.** Chương trình có thể như sau:

```
def meeting():  
    ten = input("Nhập họ tên của em: ")  
    print("Xin chào ",ten)  
meeting()
```

**Nhiệm vụ 2.** Viết hàm **prime(n)** với tham số là số tự nhiên  $n$  và trả lại **True** nếu  $n$  là số nguyên tố, trả lại **False** nếu  $n$  không phải là số nguyên tố.

**Hướng dẫn.** Số nguyên tố là số tự nhiên lớn hơn 1, không có ước nào ngoài 1 và chính nó. Để thiết lập hàm **prime(n)** chúng ta cần tính số ước thực sự của  $n$  (từ 1 đến  $n - 1$ ). Biến  $C$  dùng để đếm số các ước thực sự của  $n$ . Khi đó,  $n$  sẽ là số nguyên tố khi và chỉ khi  $C = 1$ .

Hàm **prime(n)** và chương trình có thể được thiết lập như sau:

```
def prime(n):  
    C = 0  
    k = 1
```

Ban đầu đặt  $k = 1$ . Vòng lặp sẽ tăng  $k$  lên 1 đơn vị cho đến khi  $k = n$  thì dừng. Với mỗi  $k$ , kiểm tra nếu  $k$  là ước của  $n$  thì tăng  $C$  lên 1.

```

while k < n:
    if n%k == 0:
        C = C + 1
    k = k + 1
if C == 1:
    return True
else:
    return False

```

---



## LUYỆN TẬP

- Viết hàm với tham số là số tự nhiên  $n$  in ra các số là ước nguyên tố của  $n$ .  
Gợi ý: Sử dụng hàm `prime()` trong phần thực hành.
- Viết hàm `numbers(s)` đếm số các chữ số có trong chuỗi  $s$ .  
Ví dụ `numbers("0101abc") = 4`.



## VẬN DỤNG

- Trong khi viết hàm có thể có nhiều lệnh **return**. Quan sát hàm sau và giải thích ý nghĩa của những lệnh **return**. Hàm này có điểm gì khác so với hàm `prime()` đã được mô tả trong phần thực hành.

```

def prime(n):
    if n < 2:
        return False
    C = 0
    k = 2
    while k < n:
        if n%k == 0:
            return False
        k = k + 1
    return True

```

- Viết chương trình yêu cầu nhập từ bàn phím một chuỗi ký tự, sau đó thông báo:
  - Tổng số các ký tự là chữ số của chuỗi.
  - Tổng số các ký tự là chữ cái tiếng Anh trong chuỗi.
 Viết hàm cho mỗi yêu cầu trên.