函数与抽象

Python Reference

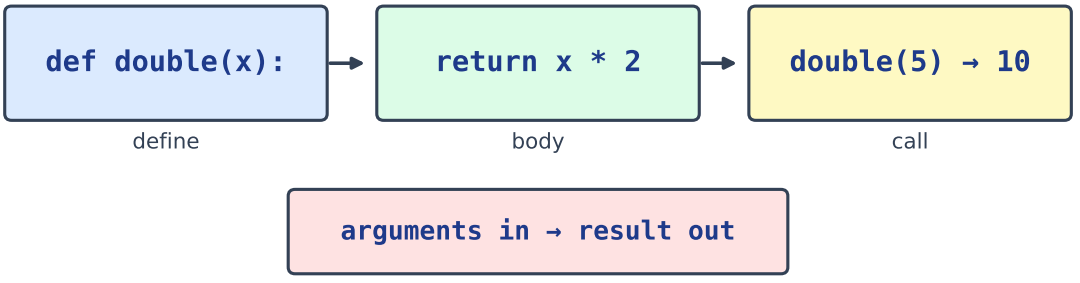
定义与调用函数

函数 (function) 是一段有名字、可以重复使用的代码。用 `def` 定义 (define) 它，再用名字调用 (call) 它。

```
def greet():
    print("Hello!")

greet()         # Hello!
greet()         # Hello!
```

- 里面的代码只有在你调用函数时才运行。



Exam tip: def creates the function; call runs it; return sends a value back to the caller

def defines; call runs; return sends a value back

返回值

函数可以用 `return` 返回 (return) 一个值。调用处就代表那个值。

```
def square(n):
    return n * n

print(square(5))          # 25
print(square(3) + 1)      # 10
```

- `return` 会立刻结束函数。没有 `return` 的函数返回 `None`。

参数、实参与作用域

形参 (parameter) 是 `def` 里的名字。实参 (argument) 是你传进去的值。

```
def power(base, exp):          # base, exp are parameters
    return base ** exp

print(power(2, 3))             # 8  (2 and 3 are arguments)
```

在函数内部建立的变量是局部 (local) 的 —— 它只在那里存在。这个范围就是它的作用域 (scope)。

```
def f():
    x = 10             # local to f
    return x

print(f())             # 10
# print(x) here would be an error: x is not defined outside f
```

形参可以有默认值 (default value), 调用者不传的时候就用它:

```
def greet(name, greeting="Hello"):
    return greeting + ", " + name

print(greet("Mei"))               # Hello, Mei
print(greet("Sam", "Welcome"))    # Welcome, Sam
```

过程抽象

过程抽象 (procedural abstraction) 的意思是把细节藏在一个名字后面。你按函数的名字和它做的事来使用它, 而不是按它怎么实现来使用。

```
def area_of_rectangle(w, h):
    return w * h

print(area_of_rectangle(4, 5))   # 20
```

- 好的函数只做一件事, 有清楚的名字, 并避免重复代码。

模块与导入

模块 (module) 是一个装着现成函数的文件。用 `import` 导入 (import) 它。

```
import random
random.seed(0)                # makes the result repeatable
print(random.randint(1, 6))   # a dice roll

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

常见错误

- 没有写 `return`, 函数就返回 `None`。打印 (print) 和返回 (return) 不是一回事。
- 不要用可变的默认值如 `def f(x=[])`—— 同一个列表会在所有调用之间共享。
- 函数内部创建的变量是局部的, 在外面看不到。
- 用 `f()` 才会运行函数; 单写 `f` 只是引用它。