

CS001 编程零基础 Python语言入门

第五讲

本讲内容

函数定义和调用

函数参数

函数返回值

变量的作用域

函数——积木

- 当程序越来越大，越来越复杂，有必要分成较小部分来组织。

函数

对象

模块：例如math、time、random

函数

内置函数

- input
- print
- int
- round

模块里的函数

- math.sqrt
- random.randint

自定义函数

Def(定义)

`def` 函数名(参数表) :
语句块

试着在函数定义里再加一句话，输出
I'm a genius!

定义（创建）函数

调用（使用）函数

```
#Function1.py  
def introduce():  
    print("I'm Sam.")  
    print("I'm 10 years old.")
```

```
introduce()  
print('Done.')
```

试着把函数名字改为praise

执行步骤

1. 执行主程序
2. 主程序调用函数时，跳到函数的第一行代码
3. 执行函数中的代码
4. 完成后，从离开主程序的那个位置继续执行

2
3

1
4

```
#Function1.py
def introduce():
    print("I'm Sam.")
    print("I'm 10 years old.")

introduce()
print('Done.')
```

为什么要使用函数？

- 一旦定义了函数，就可以通过调用反复使用。

introduce()

introduce()

introduce()

introduce()

introduce()

为什么不用循环？

- 在程序的不同位置打印，而不是全部都一次完成，循环就不行了。
- 使用函数还有个好处，可以每次调用有不同的表现。这通过函数参数来实现。

向函数传递参数

```
#Function2.py
def introduce(name):
    print("I'm", name+'.')
    print("I'm 10 years old.")

introduce('Tom')
```

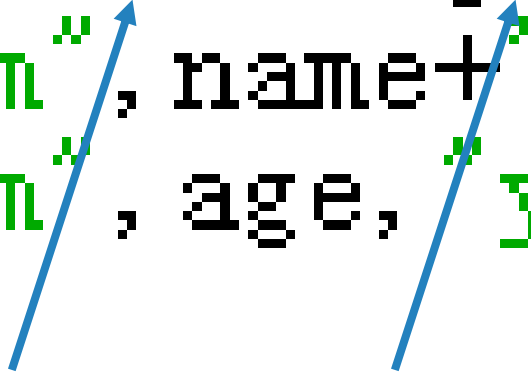
形式参数，
简称形参

实际参数，
简称实参

多个参数

```
#Function3.py
def introduce(name, age):
    print("I'm", name + ".")
    print("I'm", age, "years old.")

introduce('Alice', 11)
```

A diagram with two blue arrows. One arrow starts from the string 'Alice' in the function call 'introduce('Alice', 11)' and points to the 'name' parameter in the function definition 'def introduce(name, age)'. The other arrow starts from the integer '11' in the function call and points to the 'age' parameter in the function definition.

形参和实参

形参一定是变量。

实参可以是字面量，可以是变量，只要和形参一致。

三个一致：个数、顺序、类型

```
Name=input('Please input the name:')  
Age=int(input('Please input the age:'))  
introduce(Name, Age)
```

time模块

time.py

```
#time.py  
import time  
print(time.localtime())
```

#导入time模块
#打印本地当前时间

```
time.struct_time(tm_year=2017, tm_mon=7, tm_mday=9,  
tm_hour=9, tm_min=36, tm_sec=8, tm_wday=6, tm_yday=1  
90, tm_isdst=0)
```

年、月、日、时、分、秒、星期几（0~6表示周一到周日）、本年第几天、是否夏时制

函数的返回

Age.py

- 可以向函数发送参数
- 函数也可以向调用者发回信息，称之为结果或者返回值
- 使用关键字**return**来返回

实参是**year**变量，
形参是**birth_year**，
不同名也是可以的。

age函数的返回值
被赋给Age

```
#Age.py
import time
def age(birth_year):
    this_year=time.localtime().tm_year
    return this_year-birth_year

year=int(input('Which year were you born?'))
Age=age(year)
print('You are',Age,'years old.')
```

函数的返回

Age2.py

- 函数的返回值可以用在表达式里。
- 实际上可以用在很多地方。

```
#Age2.py
import time
def age(birth_year):
    this_year=time.localtime().tm_year
    return this_year-birth_year

year=int(input('Which year were you born?'))
print('You are', age(year), 'years old.')
```


函数实例

Temperature.py

- 摄氏转华氏温度

```
#Temperature.py
def c2f(c):
    return 1.8*c+32

c=float(input("What is the Celsius degree? "))
print('The Fahrenheit degree is', c2f(c))
```


变量作用域

- 函数里定义的变量，只能在函数里使用。称为局部变量。
- 函数的形式参数类似局部变量

this_year是函数里面定义的局部变量。
birth_year是函数的形参。
他们都不能在函数定义之外使用。

```
Which year were you born?2005
You are 12 years old.
>>> year
2005
>>> this_year
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    this_year
NameError: name 'this_year' is not defined
>>> birth_year
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    birth_year
NameError: name 'birth_year' is not defined
>>>
```

复习阅读

- 课本第5章的5.1、5.2、5.3、5.5。

