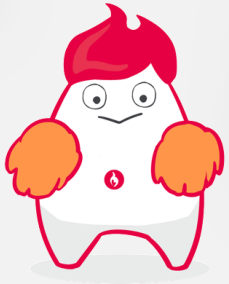
函数的定义



本节课内容

- 函数的定义
- 函数的分类
- 函数的创建方法
- 函数的返回 return

函数的定义

- ◆将一件事情的步骤封装在一起并得到最终结果
- ◆函数名代表了这个函数要做的事情
- ◆函数体是实现函数功能的流程
- ◆方法或功能

把大象装进冰箱

1：把冰箱门打开

2：把大象装进去

3：把冰箱门关上

- ◆函数可以帮助我们重复使用，通过函数名我们可以知道函数的作用

函数的分类

- 内置函数
- 自定义函数

print
id
int
str
max,
min
range

通过关键字 def 的功能

实现python中函数的创建

通过关键字 def 定义函数

def name (args...) :

todo something..

返回值

通过关键字 def 函数执行

```
In [1]: def say_hello():
   ...:     print('hello xiaomu')
   ...:

In [2]: say_hello()    # 函数名+小括号执行函数
hello xiaomu
```

函数结果的返回--return

- 将函数结果返回的关键字
- return只能在函数体内使用
- return支持返回所有的python类型
- 有返回值的函数可以直接赋值给一个变量

函数结果的返回--return

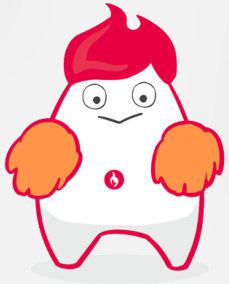
举例：

```
def add(a, b):
    c = a + b
    return c
result = add(a=1, b=1)  # 参数按顺序传递
print(result)
>> 2
```

return与print的区别

- print只是单纯的将对象打印，不支持赋值语句
- return是对函数执行结果的返回，也支持赋值语句

函数的传参



本节课内容

- 必传参数
- 默认参数
- 不确定参数
- 参数规则

必传参数

- 函数中定义的参数没有默认值，在调用函数时如果不传入则报错
- 在定义函数的时候，参数后边没有等号与默认值
- 错误：def add（a=1, b=1） x

```
In [3]: def add(a, b):
   ...:     return a + b
   ...:

In [4]: result = add()
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-9a8e1c8be307> in <module>
----> 1 result = add()

TypeError: add() missing 2 required positional arguments: 'a' and 'b'
```

```
In [5]: result = add(1, 2)

In [6]: result
Out[6]: 3
```

在定义函数的时候，没有默认值且必须在函数执行的时候传递进去的参数就是必传参数

默认参数

- 在定义函数的时候，定义的参数含有默认值，就说他是一个默认参数

```
In [7]: def add(a, b=1):
   ...:     return a + b
   ...:

In [8]: result = add(1)

In [9]: result
Out[9]: 2
```

def add(a, b=1)

参数名，赋值等号，默认值

不确定参数—可变参数

◆没有固定参数的函数，需要传递不确定参数（不知道要传的参数名具体是什么）

```
def add(*args, **kwargs):
    ...
add(1, 2, 3, name=’dewei’, age=33)
```

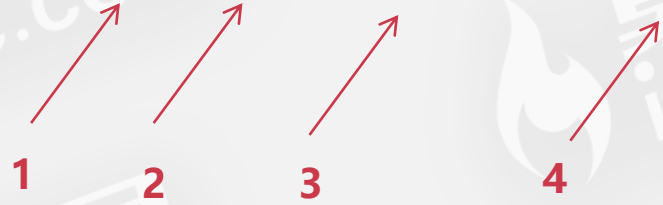
对应*args　　对应**kwargs

◆*args 代表： 将无参数的变量合并成元组

◆**kwargs 代表将有参数与默认值的赋值语句合并成字典

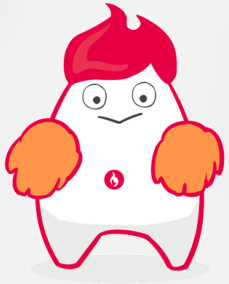
参数规则

```
def add(a, b=1, *args, **kwargs)
```



- 参数的定义从左到右依次是 必传参数，默认参数，可变元组参数，可变字典参数
- 函数的参数传递非常灵活
- 必传参数与默认参数的传参多样化

函数的参数类型定义



本节课内容

参数定义类型的方法

参数定义类型的方法

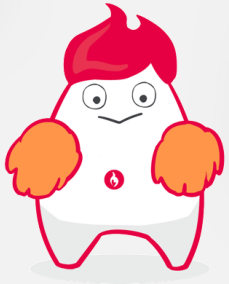
参数名+冒号+类型函数

参数名+冒号+类型
函数+等号+默认值

```
def person(name:str, age:int=33):
    print(name, age)
```

- 函数定义在python3.7之后可用
- 函数不会对参数类型进行验证

局部变量与全局变量



本节课内容

- 全局变量
- 局部变量
- global

全局变量

```
# coding:utf-8

name = 'dewei'

def test():
    print(name)
```

在python脚本最上层代码块的变量

全局变量可以在函数内被读取使用

局部变量

```
# coding:utf-8

def test():
    name = 'dewei'
    print(name)

print(name)
```

在函数体内定义的变量

X 局部变量无法在自身函数以外使用

global

将全局变量可以在函数体内进行修改

global

```
# coding:utf-8

name =

def test():
    name = 'dewei'
    print(name)

print(name)
```

在函数体内定义的变量

X 局部变量无法在自身函数以外使用

global

```
# coding:utf-8

name = 'dewei'

def test():
    global name
    name =  '小慕'

print(name)
```

定义一个全局变量

定义函数

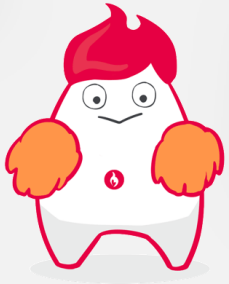
global + 全局变量名

函数体内给全局变量重新赋值

小慕

工作中，不建议使用global对全局变量进行修改

函数的递归



本节课内容

- 什么是递归函数
- 递归的定义方法
- 递归函数的说明

递归函数

只要我不累：
我可以一直跑步

一个函数不停的将自己反复执行

递归函数的定义方法

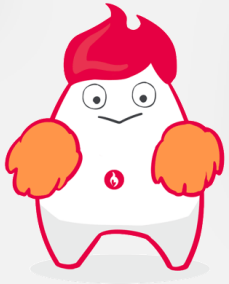
```
# coding:utf-8

def test(a):
    print(a)
    return test(a)
```

通过返回值 直接执行自身函数

递归函数的说明

- 内存溢出
- 避免滥用递归

匿名函数lambda



本节课内容

- lambda 功能
- lambda 用法

lambda功能

- 定义一个轻量化的函数
- 即用即删除，很适合需要完成一项功能，但是此功能只在此一处使用

匿名函数的定义方法

无参数

```
f =  lambda : value
f()
```

有参数

```
f =  lambda x,y: x * y
f(3, 4)
```