# python2-02-函数进阶

## 函数基础

1. 创建函数
2. 调用函数
3. 匿名函数

- 创建函数

def 语句创建函数

```
def function_name(arguments):
    function_documentation_string
    function_body_suit
```

前向引用：函数不允许函数未声明之前对其进行引用或者调用
内嵌函数：在函数体内创建另外一个函数

- **调用**函数
  利用一对圆括号来调用函数，如果不加圆括号，只是对函数的引用，不是调用函数。函数所有的参数都必须要放入到圆括号中。

关键字参数：通过调用参数的名字来调用参数，可以允许参数缺失或者不按顺序调用参数

```
def set_age(name, age):
    print('%s is %s years old' % (name, age)

set_age('bob', 23)
set_age()  #参数不够，错误
set_age('bob', 23, 100)  err, 参数太多
set_age(23, 'bob')  语法没有错，语义不正确
set_age(age = 23, name = 'bob')
set_age(age = 23, 'bob') 语法错误，kev=val样式必须在后
set_age
```

参数组：允许程序员执行一个没有显式定义的函数，方法是通过元组或字典作为参数传递给函数

```
def myfunc(*args):    *表示args是个元组
    print(args)

def myfunc2(**kwargs):  **表示args是个字典
    print(kwargs)

if __name__ == '__main__':
    myfunc()                          #args=()
    myfunc('hello')                   #参数将会放到元组中args=('hello',)
    myfunc('hello', 123)              #args=('hello', 123)
    myfunc2()
    myfunc2(name='bob', age=23)       #args={'name': 'bob','age': 23}
```

```
def add(x, y)
    print(x + y)

if __name__ == '__main__':
    add(10, 20)
    nums = [10, 20]
    add(nums[0], nums[1])
    add(*nums)                #*nums 表示把nums拆开
```

- 匿名函数
  python3 使用lambda 关键字在创建匿名函数，意思即不再使用def语句这样标准的形式定义一个函数

语法：

```
lambda 参数： expression
```

普通函数：

```
def func(x, y):
return x+y
```

---

# 函数的高级应用

1. 变量作用域
2. 函数式编程
3. 内部函数