day09-函数

函数 概念

> 什么是函数？为什么要用函数？
> 函数：特定功能的代码块

函数特点

> 1.复用性 高
> 2.节省 开发时间，提高开发效率，减少代码量
> 3.维护（修改，升级 优化）方便
> 4.灵活 结构清晰 模块性加强   模块化开发

- 函数声明 以及 调用

> 声明：
>
> ```
> 语法规范
> def 函数名(参数):
>     #代码块
> ```
>
> 调用：
>
> ```
> 函数名(参数)
> ```
>
> 注意：函数不调用不执行

- 参数

表现形式分类：

- 形参：函数声明时，函数名后括号里的参数 变量。形式参数
- 实参：函数调用时，函数名后括号里的参数。实际参数

本质：实参 是 对 形参 的 赋值。

针对python 分类：

- 位置参数：

  形参与实参进行一 一对应。数量/顺序 都要对应。
- 关键字参数：

  调用函数时，以 变量=值 形式 传实参。

  优势：不区分顺序，不用担心位置混乱。

  混用 位置参数：注意 关键字参数必须在位置参数后面
- 默认参数：
  - 声明函数时，形参=值 形式
  - 优势：调用函数时，默认参数可以不传值，直接使用默认值
  - 注意：声明函数时，默认参数 与 位置参数 混用，默认参数必须在 位置参数后面。

- 可变长参数 收集参数
  - *args
    - args:变量名称。约定俗成 该名字，也可自定义。

- 传实参时，只需要位置参数形式来传递，传递的实参会被打包成 元组的形式。args接收到是元组
- **kwargs
  - kwargs：变量名称。约定俗成 也可自定义
  - 传实参时，需要使用 关键字参数 来传实参。实参被打包为 字典 形式。

总结：
约定俗成：声明函数时：位置参数 默认参数 收集参（*args,**kwargs）
调用函数时：位置参数必须在前面 关键字参数在后

混用：

```
def my_intro(name,age=18,*hobbys,**others):
    print(name,age)
    print(hobbys)
    print(others)
my_intro('蒋浩楠',18,"女",'男','打篮球','rap',other1='烫头',other2='抽烟',other3='喝酒')
```

- return 关键字

return: 返回
    - 存在于 函数内部 的一个关键字
    - 功能一：返回函数/ 结束函数结构 （类似break）

return：
    - 功能二： 后面可以跟内容 习惯称为 返回值，为其他处所用
        - 返回值数据类型 可以多种
    - 注意：
        如果函数中没有手动书写return，系统在执行完函数内部的代码程序之后，默认会执行 return None。

- 函数嵌套调用

在一个函数内部，去调用另一个函数。

```
def fn1():
    return '我是fn1'
def fn2():
    print('我是fn2')
    fn1()    # 函数fn2在fn1内进行调用。嵌套调用
```

- 函数嵌套调用 自己

递归：在一个函数内部，去调用 自身。

- 递归 有效应用，需要有一个结束条件，从而防止 无限递归。
- 循环 同样可以 解决 递归解决的问题，所以尽量选择循环
- 递归 效率 性能低，慎用

```
# for i in range(10,0,-1):
#     print(i)

def fn1(num):
    if num == 0:
        return
    print(num)
    num -= 1
    fn1(num)

fn1(10)
```

- 作用域

概念：变量/函数 起作用的区域/范围
分类：
    - 全局作用域：
        - 从定义开始，到整个脚本文件末尾。
        - 全局变量 ： 全局作用域下声明的变量。顶格书写
    - 局部作用域(函数作用域)：
        - 函数 内部 区域
        - 局部变量：函数内部声明的变量
注意：
    - 局部作用域下 访问某个变量，首先从本作用域开始查找，如果有，直接使用，如果没有，逐级往上一级作用域查找，直到全局作用域，如果某一级有，则直接使用，如果没有，则报错nameError。就近原则

函数内部 定义了一个函数

```
def fn3():
    # 局部变量
    a = 10
    print(a,'17行')  # 5
    def fn4():
        print(a,'19行')  # 5
        print('fn4')
    fn4()
fn3()
```

- global

global： 局部作用域下 操作 全局变量，需要使用 global

1.语法规范
    global 变量名称
注意点：
    - global 变量: 也可以把 该局部变量 设置为 全局变量

```
        def fn2():
            global b
            b = 20
            print(b)
        fn2()
        print(b)
```

    - global 只针对当前本层级作用域，如果下一级作用域仍然想操作全局变量，则还需要使用global。

- nonlocal

nonlocal 可以允许 下一级作用域 对 上一级局部变量进行操作
1.语法规范
    nonlocal 局部变量
 注意点：
    - nonlocal只在当前本层级作用域起作用，如果下一级仍像操作，还需要使用nonlocal.

- 内嵌函数

函数内部 定义了一个新的函数，形成的一种代码结构

```
def fn3():
    # 局部变量
    a = 10
    print(a,'17行')  # 5
    def fn4():
        print(a,'19行')  # 5
        print('fn4')
    fn4()
fn3()
```

- 函数作为参数
- 重载

函数 重载：在python中 没有重载现象。
在python中同一个文件中，注意 函数声明时，前后函数名称如果相同，则后者会覆盖掉前者。

```
    def fn1(name):
        print('我叫',name)


    # fn1('德兴')
```

```
    def fn1(name,age):
        print('我叫',name,'年龄为',age)

    fn1('德兴',18)
```

- 匿名函数

```
普通函数
匿名函数：没有名字的函数   lambda表达式

demo1：
    a = lambda x: x + 2
    print(a)
    print(a(10))  # 12
    print((lambda : 10+30)())
特点：
    - 结构简单
    - 功能单一
    - 使用范围比较限制，仅使用一次，表示单一功能时可以选用
```