

## 09. 前方高能-初识函数

本节内容:

1. 什么是函数
2. 函数定义, 函数名, 函数体以及函数的调用
3. 函数的返回值
4. 函数的参数

### 一. 什么是函数

1. 我们到目前为止, 已经可以完成一些软件的基础功能了. 那么我们来完成这样一个功能: 约x:

```
print("拿出手机")
print("打开陌陌")
print("找个漂亮的妹子")
print("问她, 约不约啊?")
print("oK. 走你!")
```

ok. so easy. 我们已经完成了对一个功能的描述. 那么问题来了. 我还想再约一次. 怎么办呢? 很简单. 再写一次就好了

```
# 约一次
print("拿出手机")
print("打开陌陌")
print("找个漂亮的妹子")
print("问她, 约不约啊?")
print("oK. 走你!")
# 再来一次
print("拿出手机")
print("打开陌陌")
print("找个漂亮的妹子")
print("问她, 约不约啊?")
print("oK. 走你!")
```

OK. 也很简单. 但是. 我现在还想约. 约个10次8次的. 怎么办呢? 也简单. 加个循环就好了

```
while 1:
    print("拿出手机")
    print("打开陌陌")
    print("找个漂亮的妹子")
    print("问她, 约不约啊?")
    print("oK. 走你!")
```

哇, 终于可以不停的约了. 但是呢, 想想. 这样写出来的程序. 是不是一直在约? 人啊. 要有节

制. 有需求了再约, 这样比较好. 所以呢. 这样写是不行的. 最好是我想什么时候约就什么时候约. 好了. 说到这. 我们可以这样做, 把约会这个事情啊, 先计划一下, 然后呢安排好流程. 在需要约的时候呢. 把这个约的流程拿出来执行一下就好了. 那么这里. 我们可以先去定义一个事情或者功能. 等到需要的时候直接去用就好了. 那么这里定义的东西就是一个函数.

## 函数: 对代码块和功能的封装和定义

### 二. 函数的定义, 函数名, 函数体以及函数的调用

我们使用def关键字来定义函数, 函数的定义语法:

```
def 函数名():  
    函数体
```

这里的函数名的命名规则和使用和变量基本一样. 自己回顾一下变量的命名规则.

函数体: 就是函数被执行之后要执行的代码

让我们来定义一个约x功能:

```
def yue():  
    print("拿出手机")  
    print("打开陌陌")  
    print("找个漂亮的妹子")  
    print("问她, 约不约啊?")  
    print("oK. 走你!")
```

哦了定义完了. 但是这个时候去执行. 会发现什么都没有发生. 因为我只定义了一个函数. 但是还没有执行过这个函数.

函数的调用: 使用函数名可以调用函数, 写法: 函数名(), 这个时候函数的函数体会被执行

```
# 调用yue()函数  
yue()
```

结果:  
拿出手机  
打开陌陌  
找个漂亮的妹子  
问她, 约不约啊?  
oK. 走你!

看一下执行过程:

<pre> 1 def yue(): 2     print("拿出手机") 3     print("打开陌陌") 4     print("找个漂亮的妹子") 5     print("问她, 约不约啊?") 6     print("ok. 走你!") 7 8 # 调用yue()函数 9 yue() </pre>	<ol style="list-style-type: none"> <li>1. 定义函数yue()</li> <li>2. 调用函数yue()</li> <li>3. 准备开始执行函数yue()</li> <li>4. 打印 拿出手机</li> <li>5. 打印 打开陌陌</li> <li>6. 打印 找个漂亮的妹子</li> <li>7. 打印 问她, 约不约</li> <li>8. 打印 ok, 走你! 此时 yue()的函数体执行完毕</li> <li>9. 函数执行完毕. 本次调用完毕, yue()这次调用完毕</li> </ol>
--	--

终于可以约了. 如果我还想约呢? 多次调用就可以了. 很方便.

```

# 调用yue()函数
yue()
yue()
yue()
yue()
yue()

```

继续分析. 约完了之后你需要得到一个结果吧. 比如. 约完了得到了一个萝莉, 少妇, 大妈. 总得有个结果. 那么这个结果怎么来描述和获得呢? 这个就涉及到函数的返回值的问题.

### 三. 函数的返回

执行完函数之后. 我们可以使用return来返回结果.  
函数中return的使用:

1. 函数中遇到return, 此函数结束, 不再继续执行.

```

def yue():
    print("约你")
    print("约我")
    print("约他")
    return
    print("约谁呀")    # 这句话不会被执行

yue()

```

2. 给函数的调用者一个访问结果

```

def yue():
    print("约你")
    print("约我")
    print("约他")
    return "美女一枚"

girl = yue()
print(girl)    # 美女一枚

```

函数的返回值可以有多个结果

```
def yue():  
    print("约你")  
    print("约我")  
    print("约他")  
    return "美女一枚", "萝莉一枚"  
  
girl = yue()  
print(type(girl))    # tuple
```

总结一下:

1. 遇到return. 此函数结束, 函数后面的东西将不会再执行
2. return 返回值

关于返回值:

如果return什么都不写 或者 干脆不写return .那么返回的就是None

如果return后面写了一个值. 则调用者可以接收一个结果

如果return后面写了多个结果, 则调用者可以接收一个tuple, 调用者可以直接解构成多个变量

OK. 完美. 可以得到结果了. 但是我们的约的方式是不是有点儿问题呢?, 陌陌现在还能约到么? 约不到了吧. 该换探探了. 那过两天探探也死掉了呢? 是不是还会有一个替代品. 那你想. 有一个替代的. 你就需要去改一下代码. 是不是不太合适了. 最好的方式是不是想用什么约就用什么约? ok. 我们可以通过函数的参数来给函数传递一些信息.

#### 四. 函数的参数

参数, 函数在调用的时候指定具体的一个变量的值. 就是参数. 语法:

```
def 函数名(参数列表):  
    函数体
```

首先我们先把代码该一下. 能够实现上面的需求.

```
def yue(chat):  
    print("拿出手机")  
    print("打开"+chat)  
    print("找个漂亮的妹子")  
    print("约不约")  
  
yue("陌陌")  
yue("微信")  
yue("探探")
```

结果：

```
拿出手机
打开陌陌
找个漂亮的妹子
约不约
拿出手机
打开微信
找个漂亮的妹子
约不约
拿出手机
打开探探
找个漂亮的妹子
约不约
```

perfect. 搞定了. 我们在调用yue的时候给chat一个值. 然后再执行函数体.

关于参数:

### 1. 形参

写在函数声明的位置的变量叫形参. 形式上的一个完整. 表示这个函数需要xxx

### 2. 实参

在函数调用的时候给函数传递的值. 叫实参, 实际执行的时候给函数传递的信息. 表示给函数xxx

### 3. 传参

给函数传递信息的时候将实际参数交给形式参数的过程被称为传参.

```
def yue(chat):    # chat 形参
    print("拿出手机")
    print("打开"+chat)
    print("找个漂亮的妹子")
    print("约不约")

yue("陌陌")      # 实参

len("字符串")    # "字符串"在这里就是实参
print("麻花藤")  # "麻花藤"就是实参
```

参数的分类:

首先我们先看实参:

#### 4.1.1 位置参数

约到这里了. 有没有想过这个问题. 啥样的都约么? 哪里的都约么? 不一定吧. 比如, 我在

北京, 我很寂寞, 我喜欢小姐姐. 强哥, 在哈尔滨, 很寂寞, 大妈就行了. 需求是不一样的. 而我们的函数没有这些功能. 那怎么办呢? 很简单, 多来几个参数就好了

```
def yue(chat, address, age):    # 形参
    print("拿出手机")
    print("打开"+chat)
    print("找个"+address+"附近漂亮的"+str(age)+"岁妹子")
    print("约不约")
```

```
yue("微信", "北京", 18)    # 实参
```

结果:

拿出手机

打开微信

找个北京附近漂亮的18岁妹子

约不约

分析: 在访问yue()的时候, 我们按照位置的顺序分别把"微信", "北京", 18 赋值给 chat, address, age. 在传参过程中. 系统会默认按照位置把实参赋值到形参.

练习: 编写函数, 给函数传递两个参数a, b. 比较a, b的大小, 返回 a, b中最大的那个数

答案:

```
def my_max(a, b):
    if a > b:
        return a
    else:
        return b

# 有点儿麻烦, 我们在这里学一个三元运算符.
def my_max(a, b):
    c = a if a > b else b    # 当a>b成立返回a, 否则返回b
    return c
```

#### 4.1.2 关键字参数

位置参数好不好呢? 如果是少量的参数还算OK, 没有问题. 但是如果函数在定义的时候参数非常多怎么办? 程序员必须记住, 我有哪些参数, 而且还有记住每个参数的位置, 否则函数就不能正常调用了. 那怎么办呢? python提出了一种叫做关键字参数. 我们不需要记住每个参数的位置. 只要记住每个参数的名字就可以了

```
def yue(chat, address, age):
    print("拿出手机")
    print("打开"+chat)
    print("找个"+address+"附近漂亮的"+str(age)+"岁妹子")
    print("约不约")

yue(chat="微信", age=18, address="北京")    # 关键字参数.
```

结果：  
拿出手机  
打开微信  
找个北京附近漂亮的18岁妹子  
约不约

搞定, 这样就不需要记住繁琐的参数位置了.

#### 4.1.3 混合参数

可以把上面两种参数混合着使用. 也就是说在调用函数的时候可以给出位置参数, 也可以指定关键字参数.

```
# 混合参数
yue("微信", age=18, address="上海")    # 正确. 第一个位置赋值给chat, 后面的参数开始指定关键字.

yue(age="18", "微信", address="广州")    # 错误, 最开始使用了关键字参数, 那么后面的微信的位置就串了, 容易出现混乱
```

**注意: 在使用混合参数的时候, 关键字参数必须在位置参数后面**

综上: 在实参的角度来看. 分为三种:

1. 位置参数
2. 关键字参数
3. 混合参数, 位置参数必须在关键字参数前面

#### 4.2 在形参角度看. 一共分为三种. 今天我们学习两种

4.2.1 位置参数. 按照位置来赋值, 到目前为止, 我们编写的函数都是这种

```
def yue(chat, address, age):
    print("拿出手机")
    print("打开"+chat)
    print("找个"+address+"附近漂亮的"+str(age)+"岁妹子")
    print("约不约")
```

4.2.2 默认值参数. 在函数声明的时候, 就可以给出函数参数的默认值. 在调用的时候可以给出具体的值, 也可以不给值, 使用默认值.

比如, 我们录入咱们班学生的基本信息. 通过调查发现. 我们班大部分学生都是男生. 这个时候就可以给出一个sex='男'的默认值.

```
def stu_info(name, age, sex='男'):
    print("录入学生信息")
```

```
print(name, age, sex)
print("录入完毕")

stu_info("张强强", 18)
```

注意, 必须先声明位置参数, 才能声明默认值参数.

综上: 在形参的角度来看,

1. 位置参数
2. 默认值参数(大多数传进来的参数都是一样的, 一般用默认参数)