一,函数定义的弊端:

1）Python是动态语言,变量随时可以被赋值,且能赋值为不同的类型。

2）Python不是静态编译型语言，变量类型是运行器决定的

3）动态语言很灵活，但这也是其弊端：

def add(x + y):

return x + y

print(add(4,5))

print(add('hello','world'))

add(4,'hello') #这句显然是错的

首先，难发现，由于不做任何类型检查，直到运行期问题菜显现出来，或线上运行时才发现问题；

其次，难使用，函数使用者看到函数的时候并不知道你的函数的设计，并不知道应该传入什么类型的数据

二，如何解决这种弊端？

1）增加文档Documentation String:

这只是一个惯例,并不是强制标准,不能要求程序员一定为函数提供说明文档,并且函数定义更新了,文档未必同步更

def add(x,y):

'''

:param x:int

:param y:int

:return: int

'''

return x + y

print(help(add))

2）函数注解

def add(x:int,y:int)->int:

'''

:param x:int

:param y:int

:return: int

'''

return x + y

print(help(add))

print(add(4,5))

print(add('func','tion'))

定义：

Python 3.5引入；对函数的参数进行类型注解；对函数的返回值进行类型注解；只对函数参数做一个辅助的说明，并不对函数参数进行类型检查；提供给第三方工具，做代码分析，发现隐藏BUG；函数的注解信息保存在\_\_annotations\_\_属性中;

————————————————

版权声明：本文为CSDN博主「奔跑的小狼狗」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/weixin\_41869526/article/details/80021061