4-函数

4.1-函数的说明文档

作用：查看函数的作用

声明：

```
def 函数名(参数)
    """说明文档的位置"""
    代码
    ...
```

查看函数的说明文档

```
help(函数名)
```

4-2 函数的嵌套调用

```
def test1():
    print("---开始执行test1---")
    print("执行test1")
    test2()
    print("---结束执行text1---")

def test2():
    print("---开始执行test2---")
    print("执行test2")
     print("---结束执行text2---")

test1()

# result:
# ---开始执行test1---
# 执行test1
# ---开始执行test2---
# 执行test2
# ---结束执行text2---
# ---结束执行text1---
```

4-3 函数的参数

1、位置参数：根据函数定义的参数位置来传递参数

2、关键字参数：通过“键=值”的形式对参数加以指定

```
def user_info(name,age,gender):
    print(f'您的名字是{name},{age}岁,性别{gender}')

#位置参数
user_info("yulong",21,"男")
#关键字参数
user_info("Yanlei",gender="女",age=21)

# result:
# 您的名字是yulong,21岁,性别男
# 您的名字是Yanlei,21岁,性别女
```

3、缺省参数：默认参数，定义函数时为其提供默认值

```
def user_info(name,age,gender="男"):
```

```
    print(f'您的名字是{name},{age}岁,性别{gender}')

#缺省参数
user_info("Yulong",21)
user_info("Yanlei",gender="女",age=21)

# result:
# 您的名字是Yulong,21岁,性别男
 # 您的名字是Yanlei,21岁,性别女
```

4、不定长参数：使用packing位置参数传递关键字

```
#包裹位置传递（组合成元组）
def user_info(*args):
    print(args)
user_info("yulong",21)
#result:('yulong', 21)

#包裹关键字传递（组合成字典）
def user_info_key(**kwargs):
    print(kwargs)
 user_info_key(name="yulong",age=21)
 #result:{'name': 'yulong', 'age': 21}
```

4-4 拆包

拆元组：

```
def outCheck(tuple):
    a,b,c = tuple
    return a,b,c
tuple1 = (100,200,300)
first,second,third = outCheck(tuple1)
print(first,second,third)
#result:100 200 300
```

拆字典：

```
def outCheck(dict):
    a,b=dict
    return a,b

dict1 = {"name":"yulong","age":21}
first,second=outCheck(dict1)
print(first,second)
print(dict1[first],dict1[second])

 #result:
 # name age
 # yulong 21
```

4-5 变量值交换

```
a=10
b=20
print(a,b)
a,b=b,a
print(a,b)

```

```
#result:
# 10 20
# 20 10
```

4-6 递归

应用：n的累乘

```
def function(n):
    if n!=1:
        result = function(n-1) * n
    else:
        result = 1
    return result
print(function(10))
# result:3628800
```

4-7 lambda表达式

用法：一个函数有一个返回值且只有一句代码时用lambda简化

```
def fn1():
    return 100
print(fn1)
print(fn1())
# result:
# <function fn1 at 0x7fc7e68d3040>
# 100

fn2 = lambda : 200
print(fn2)
print(fn2())
# result:
# <function <lambda> at 0x7fc7e6ac59d0>
# 200
```

> 注意：
> 1、lambda表达式参数可有可无
> 2、lambda表达式能接收任何数量的参数但只能返回一个表达式的值
> 3、直接打印lambda表达式，输出的是lambda的内存地址

应用：

```
#lambda的应用
print((lambda a,b:a+b)(1,2))                    #3

#无参形式
print((lambda :100)())                          #100
#一个参数
print((lambda a:a)("hello world"))              #hello world
#默认参数
print((lambda a,b,c=100:a+b+c)(10,20))          #130
#可变参数：*args
print((lambda *args:args)(10,20,30))             #(10, 20, 30)
#可变参数：**kwargs
print((lambda **kwargs:kwargs)(name="yulong",age=21))   #{'name': 'yulong', 'age': 21}
#带判断的lambda
print((lambda a,b:a if a>b else b)(1000,500))   #1000
```

lambda结合字典

```
students=[
    {"name":"Yulong","age":21},
    {"name":"Ze","age":22},
    {"name":"Jiameng","age":24}
]
#按age值升序排列
students.sort(key=lambda x:x['age'])
print(students)
#按age值降序排列
 students.sort(key=lambda x:x['age'],reverse=True)
 print(students)

#result:
# [{'name': 'Yulong', 'age': 21}, {'name': 'Ze', 'age': 22}, {'name': 'Jiameng', 'age': 24}]
# [{'name': 'Jiameng', 'age': 24}, {'name': 'Ze', 'age': 22}, {'name': 'Yulong', 'age': 21}]
```

4-8 高阶函数

1、abs

```
abs(-2)             #2
```

2、map

作用：将传入函数变量func作用到list变量的每个元素中，结果组成新的列表

```
list1=[2,3,5,7,11]
def func(x):
    return x**2
result = map(func,list1)
print(result)               #<map object at 0x7f824a8c9550>
print(list(result))         #[4, 9, 25, 49, 121]
```

3、reduce

作用：每次func计算的结果继续和序列的下一个元素做累加和计算

> 注意：reduce()传入的参数func必须接受2个参数

```
from functools import reduce
list1=[2,3,5,7,11]
def func(a,b):
    return a+b
result = reduce(func,list1)
print(result)               #28
```

4、filter

作用：过滤序列。过滤掉不符合条件的元素，返回一个filter对象。（用list函数转换）

```
list1=[2,3,5,7,11]
def func(x):
    return x % 2==0
result = filter(func,list1)
print(result)
print(list(result))
#result:2
```