一 课程回顾

```
# 1.定义一个函数，可以用来求任意三个数的乘积
def fn(a,b,c):
    print(a*b*c)
fn(1,2,3)

# 2.定义一个函数，可以根据不同的用户名显示欢迎信息
def welcome(username):
    print('欢迎',username,'光临')
welcome('左手')
welcome('一大口')

# 3.编写一个函数，功能是输出2个数中较大的一个数
def max(a,b):
    # if a > b:
    #     print(a)
    # else:
    #     print(b)
    # 三目运算
    print(a if a > b else b)
max(5,10)


# 编写一个函数，判断用户传入的列表长度是否大于2，如果大于2，就保留前两个 并输出出来
def fn1(a):
    if len(a) > 2:
        print(a[:2])
    else:
        pass
b = [1,2,3,4,5,6,7]
fn1(b)


# 编写一个函数，接收一个列表(列表元素个数不限)，求列表数字的和
def fn2(c):  # c = [1,2,3,4,5,745]
    result = 0  # 存储变量
    for i in c:
        result += i  # result = 0 + 1
    print(result)
fn2([1,2,3,4,5,745])


# 写一个函数，判断用户传入的对象（字符串、列表、元组）的元素是否为空
def fn3(d):
    if len(d) == 0:
        print('对象为空')
    else:
        print('对象不为空')
fn3((1,2,3))
fn3([])
```

```
# 输入颜色（RGBA），打印描述信息，否则提示颜色不存在(这道题不是一定得用函数写)
#     "R" -->"红色"
#     "G" -->"绿色"
#     "B" -->"蓝色"
#     "A" -->"透明度"
dict1 = {'R':'红色','G':'绿色','B':'蓝色','A':'透明度'}
e = input('请输入颜色(RGBA):')
if e in dict1:
    print(dict1[e])  # dict1[e] = 值
else:
    print('您输入颜色的不存在')
```

二 函数基础下

2.1 可变参数 （不定长参数）

定义一个函数可以求任意个数的和

我们在定义函数时写形参的时候可以再形参前面加一个*号，保存到一个元祖中

```
def sum(a,b,c):
    print(a+b+c)
sum(1,2,3)

def fn(*args):
    print(args)
    print(type(args))
print(fn)

def fn(*args): # *a  *b  *c
    print(args)
    print(type(args))
fn(1,2,3,4,5,6,7,8,9,10,11,12,15)


def sum(*args):  # def 定义函数
    result = 0  # 初始化数据  存储变量
    for i in args:
        result += i
    print(result)
sum(1,2,3,4,5,6)
sum(1,2,3,4,5,6,54,413,53153,153)
```

带*的形参只能有一个，带*的参数写多了 就没有办法分配实参
带*的参数可以和其他的参数配合使用，但是得写在最后面

```
def fn(a,b,*c):
    print('a=',a)
    print('b=',b)
    print('c=',c)
fn(1,2,3,4,5,6,78,8,9,10)

def fn(*c,a,b,d):
```

```
    print('a=',a)
    print('b=',b)
    print('c=',c)
    print('d=',d)
fn(1,2,3,4,5,6,78,d=8,a=9,b=10)
```

使用带*的参数一般取名 *args

*args弊端就是只能接收未知参数

**kwargs

```
def fn(**args):
    print('args=',args)
    print(type(args))
fn(a=1,b=2,c=3)
```

总结：
**形参可以接受任意的关键字参数，会将这些参数统一保存到字典当中，字典的键是实参 数据的变量名，字典的值就是实参 变量的数据，所以 现在 键是a 值是1
注意： **形参也是只能有一个，并且必须写在所有参数最后

2.2 同时使用不定长参数

当要同时使用*args 和 * *kwargs 的时候， *args 必须写在 * *kwargs的前面 这是语法规定

```
def fn(*args,**kwargs):  # **args  **kwargs
    # print(a)
    # print(b)
    print(args)
    print(kwargs)
fn(1,2,3,4,5,31,53,352,5,3515,3,k=1,c=5)
```

2.3 参数的解包

```
def fn(a,b,c):
    print('a=',a)
    print('b=',b)
    print('c=',c)
t = (1,2,3)
fn(t[0],t[1],t[2])
fn(*t)
```

传递实参时，也可以在序列类型的参数前面添加*号，这样会自动将序列中的元素依次作为参数传入
注意：序列中的元素个数必须和我们函数形参的个数一样 # 序列类型：str list tuple

```
def fn(a,b,c):
    print('a=',a)
    print('b=',b)
    print('c=',c)
t = (1,2,3,4)
fn(t[0],t[1],t[2])
```

```
fn(*t)  # 报错  元素个数与形参个数不一致
```

```
def fn(a,b,c):
    print('a=',a)
    print('b=',b)
    print('c=',c)
d = {'a':10,'b':20,'c':30}
fn(**d)
```

要注意字典的key必须和形参一致

总结：
1.*args适用于接收多个未命名的位置参数，**kwargs用于接收关键字参数，其中args是一个元祖类型，而kwargs是一个字典类型
2.解包：*args也就是把元祖中的数据拆成单个的数据
**kwargs把字典中的数据拆成单个的键值对

2.4 函数的返回值

返回值就是函数执行以后返回的结果

f(x) = 2x + 1

x = 1 == > 3 f(1) = 2*1 +1

x = 2 == > 5 f(2) = 2 *2 + 1

return ==> 指定函数的返回值

```
def fn():
    return 1
fn()
# 获取返回值方法是 把 函数调用当做一个值来输出

def fn():
    return {'a':1}

x = fn()
print(x)
print(fn())
```

return 后面是什么， 返回的结果就是什么
return后面可以是任意的值，包括函数
无论定义的是返回什么类型，return只能返回单个值，但是值可以有多个元素

```
def fn():
    return 1,3,5

x = fn()
print(x)
print(type(fn()))
```

```
def fn():
    def fn2(): # 经过fn2函数  没有用到fn2  因为没有调用fn2
        print('hello')
    print('f')
fn()


def fn():
    def fn2(): # 经过fn2函数  没有用到fn2  因为没有调用fn2
        print('hello')
    return fn2
fn()


def fn():
    def fn2(): # 经过fn2函数  没有用到fn2  因为没有调用fn2
        print('hello')
    return fn2

fn()  # fn() = fn2
a = fn()
print(a)  # fn2 内存地址
a()
```

None是一个特别空值，用来代表空的

```
def fn():
    pass
print(fn())
print(type(None))

如果不给函数返回值，那么他的返回值就是None
只要函数不带return 返回的就是None

def fn():
    return None
print(fn())

总结：如果仅仅写一个return 或者不写 return 就等价于return None
```

```
def fn():
    print('1')
    print('2')
    print('3')
    return
    print('4')

fn()

在函数中，return代表函数执行结束
```

```python
def fn():
    for i in range(1,101):
        if i == 66:
            print('提前结束')
            # return
            break
        print(f'抄{i}遍')
fn()
```

break: 退出当前循环
continue: 跳过本次循环，但是对于后面没有影响
return : 用来结束函数

```python
def sum(*args):  # def 定义函数
    result = 0  # 初始化数据  存储变量
    for i in args:
        result += i
    return result
r = sum(1,2,3,4,5,6)
print(r+10)


a = len('hello')
print(a)  #len方法的返回值 就是他的长度
```

fn和fn()的区别
fn==》函数对象，打印他其实就是在打印函数对象
fn() ==> 调用函数，打印fn()其实就是在打印fn()的返回值

总结:
    return语句的作用
结束函数调用，返回值
函数体中return语句有指定的返回值时返回的是其值
函数体中没有return语句，函数运行结束会隐含返回一个None作为返回值，类型是NoneType 与return ，return None等效 都是返回None

print和return
print仅仅是打印在控制台，而return则是将return后面的部分作为返回值，作为函数的输出，可以用变量接受 ，继续使用这个返回值做其他事

作业

求两个浮点数的差 并返回他们的值

判断一个整数n能否同时被3 和 5 整除 ，如果能同时被3和5整除返回yes，否则返回No

定义一个函数，函数传一个列表[1,2,3, -1, -2,2],统计列表中负数的个数并返回结果

定义一个函数 ， 传入一个有重复元素列表 a=[1,2,3,4,4,5,6,2,1] ， 让函数返回的是一个无重复元素的列表