day04 课堂笔记

课程之前

复习和反馈

作业

今日内容

```
容器
函数
定义和调用函数，特点
```

容器

in 操作符

```
in 是 Python 中的关键字.
数据 in 容器    可以用来判断 容器中是否包含这个数据，如果包含返回 True,如果不包含返回 False
对于字典来说,判断的是 字典中是否包含这个键
```

集合[了解]

集合 set，{数据，数据，...}
1，集合中的数据是不能重复的，即没有重复数据
2，应用，对列表进行去重操作 就是类型转换 ，可以将 列表转换为 集合，然后再将集合转换为列表

```
my_list = [1, 2, 1, 2, 5, 2, 2, 4, 13]

# 方式一
list1 = list(set(my_list))
print(list1)

# 方式二
new_list = []   # 定义新列表 ,保存去重后的数据

# # 遍历原列表
# for i in my_list:
#     # 判断数据是否存在新列表
#     if i in new_list:
#         # 存在 什么都不操作
#         pass
#     else:
#         # 不存在，添加到新列表
```

```
#         new_list.append(i)

# 遍历原列表
for i in my_list:
    # 判断数据是否存在新列表
    if i not in new_list:
        # 不存在，添加到新列表
        new_list.append(i)

print(new_list)
```

函数

```
print() 在控制台输出
input() 获取控制台输入的内容
type()  获取变量的数据类型
len()   获取容器的长度（元素的个数）
range() 生成一个序列[0, n)
```

函数可以实现一个特定的功能

我们学习自己如何定义函数，实现特定的功能

函数：将多行代码(可以实现一个特定的功能)放在一块,并给它起一个名字．在需要使用多行代码的时候，可以使用名字代替．

定义函数的好处：减少代码冗余(重复的代码不需要多次书写),提高编程效率

函数必须 先定义 后 调用

定义和调用

定义

1，函数定义,就是给多行代码起名字的过程

2，函数的定义需要使用 关键字 def，单词 define

```
def 函数名():
    函数中的代码
    函数中的代码


# 1，处在 def 缩进中的代码 都属于这个函数
# 2，函数名要满足标识符的规则，见名知意
# 3，def 这行代码的最后需要一个 冒号
# 4，函数定义不会执行函数中的代码,想要执行,需要调用
```

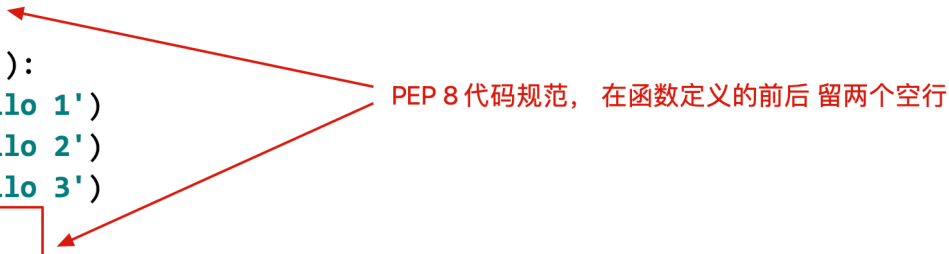
调用

```
1，函数调用,就是使用 多行代码的过程
2，语法  函数名()
```

定义函数的小技巧

```
1，先不使用函数,将多行代码写完
2，在多行代码的上方使用 def 起名字
3，使用 tab 键，将多行代码进行缩进
```

```python
"""
1. 编写一个打招呼 say_hello 的函数，封装三行打招呼的代码
2. 在函数下方调用打招呼的代码
"""


def say_hello():
    print('hello 1')
    print('hello 2')
    print('hello 3')


# 函数调用
say_hello()
say_hello()
```



函数的文档注释[了解]

1， 函数的文档注释,本质就是注释， 只不过作用和书写位置有特定的要求

2， 作用： 是对函数的作用和使用方法进行说明， 比如 有哪些参数， 返回值是什么

3， 书写位置： 在def 的下方,使用 三对双引号来书写

查看

4.1 在函数名上,使用快捷键 Ctrl q 查看

4.2 在函数名上,使用 快捷键 Ctrl B 跳转到函数定义的地方查看

4.3 在函数名上， 按住 Ctrl 键,点击函数名,跳转到函数定义的地方查看

参数

参数：在函数定义的时候,在括号中写入变量,这个变量就称为是函数的参数. 形式参数(形参)
在函数调用的时候,可以给定义时候的形参传递具体的数据值,供其使用. 实际参数(实参)

即：在函数调用的时候,会将函数的实参值传递给形参.

好处：可以让函数更加的通用，函数中的数据值不是固定的,是调用的时候,你传递的.

使用场景：判断 函数中 数据值是不是固定不变的，如果是变化的,就可以使用参数传递

注意点：目前书写的函数，如果存在形参,必须传递相同个数的实参.

```
def sun_2_num(a, b):  # a, b  形参
    c = a + b
    print(c)


sun_2_num(10, 20)   # 10, 20 实参 10 给 a,  20 给 b
sun_2_num(1, 2)
sun_2_num(20, 39)
```

函数的嵌套调用

在一个函数中调用另一个函数.

1，代码从上到下执行的
2，函数定义不会执行函数中的代码
3，函数调用会进入函数中执行函数中的代码
4，函数中的代码执行结束,会回到调用的地方继续向下执行

```
# 1. 定义名为test01的函数，打印当前函数的名称
def test01():
    print(1)
    print('func01')
    print(2)


# 2. 定义名为test02的函数，打印当前函数的名称，并 调用test01函数
def test02():
    print(3)
    print('func2')
    test01()
    print(4)
```

```
print(5)
test02()
print(6)

# 5 3 1 2 4 6
#
```

函数的返回值

返回值: 函数执行的结果

print() ---> None

input() ---> 键盘输入的内容, 类型 字符串

type()   ---> 变量的数据类型

len()   ---> 容器长度

1, 在一个函数中,想要返回一个数据(想要有返回值), 需要使用 return 关键字

2, 为什么返回值? 在函数中可能通过各种代码,得到的数据结果,想要在函数外部使用,就需要使用返回值

3, 如果函数有返回值, 一般在调用的时候 会使用变量来接收(保存) 返回值, 以便后续使用

4, return 关键字的作用,

- 将一个数据值返回到调用的地方
- 函数遇到 return 会结束函数的执行

5, return 关键字只能用在函数中

6，如果一个函数 没有写 return,可以认为 返回值是 None

```
# 设计一个函数用于获取两个数中的较大数，数据来自于函数的参数
def get_max(a, b):
    if a > b:
        return a
    else:
        return b
    print('我会执行吗，不会执行')


# 调用
num = get_max(10, 20)
print(num)
# 1 会  2 不会
```

案例

1. 定义名为 input_username 的函数，获取用户输入的用户名
2. 定义名为 input_password 的函数，获取用户输入的密码
3. 定义名为 login 的函数，判断获取的用户名和密码信息
4. 要求当获取的用户名为:admin 并且密码为: 123456 时，输出“登录成功!”，否则提示“用户名或 密码错误!”

```
# 1. 定义名为 input_username 的函数，获取用户输入的用户名
```

```python
def input_username():
    """输入用户名"""
    return input('请输入用户名:')


# 2. 定义名为 input_password 的函数，获取用户输入的密码
def input_password():
    """输入密码"""
    return input('请输入密码:')


# 3. 定义名为 login 的函数，判断获取的用户名和密码信息
# 4. 要求当获取的用户名为:admin 并且密码为: 123456 时，输出“登录成功!”，否则提示“用户名或 密码错误!”
def login():
    """登录函数"""
    if input_username() == 'admin' and input_password() == '123456':
        print('登录成功')
    else:
        print('用户名或密码错误')


login()
```

模块和包

模块

1，在Python 中，每个代码文件 都可以称为是一个模块
2，在模块中 别人书写好的功能(变量，函数，类),我们可以拿来直接使用
3，我们自己写的代码文件，想要作为模块让别人使用，你的代码文件名(模块名) 满足标识符的规则
4，想要使用 别人模块中写好的功能,需要先导入别人写好的功能
5，as 关键字 ，可以给模块或者功能起别名

模块导入的方法

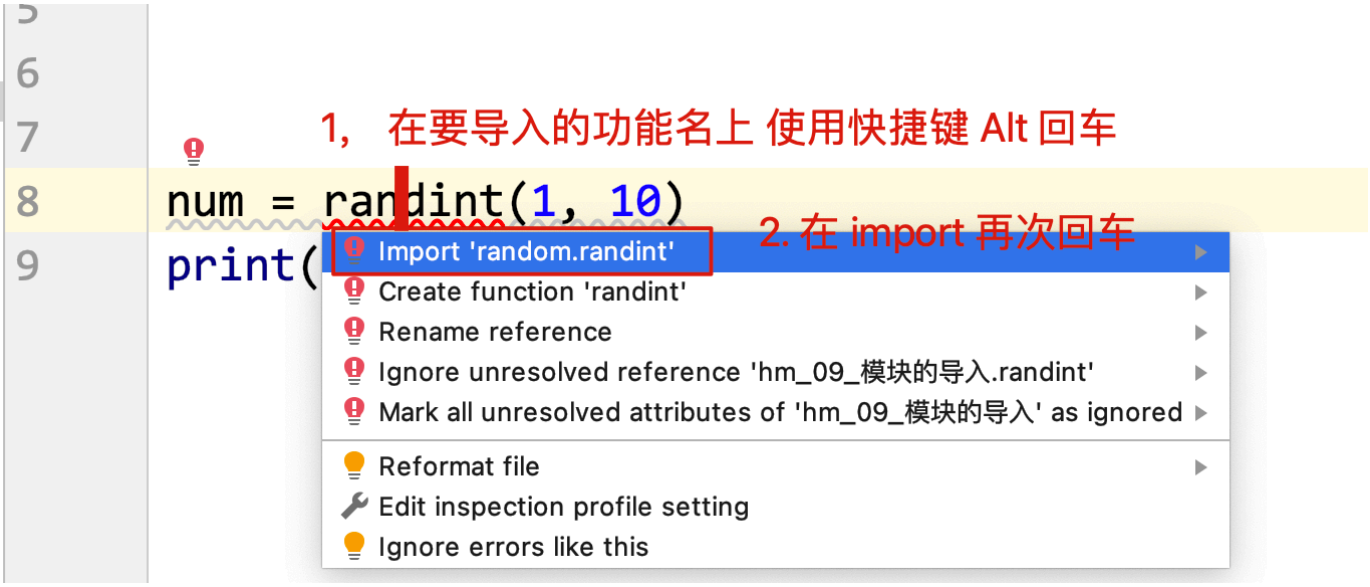
方式一

```
import 模块名    # 模块名 就是代码文件名 ,不要 .py

# 使用其中的功能
模块名.功能名   # 功能可以是变量，函数 和类

# 多用于 导入系统中的常用的模块和功能
```

```python
import random

num = random.randint(1, 10)
print(num)
```

方式二

```python
from 模块名 import 功能名  # 导入指定的功能

# 使用
功能名()

# 方式二  多用于导入自己书写的,或者是第三方的模块
# 可以使用快捷键 Alt 回车
```



```python
from random import randint

num = randint(1, 10)
print(num)
```

练习

```
1. 定义一个模块 tools.py , 在模块中定义一个函数 sum_2_num(), 可以对两个数字求和
2. 新定义一个代码文件, 调用 tools.py 文件中的 sum_2_num() 函数, 对 10 和 20 求和
```

- `tools.py`

```python
def sum_2_num(a, b):
    """求和函数"""
    return a + b
```

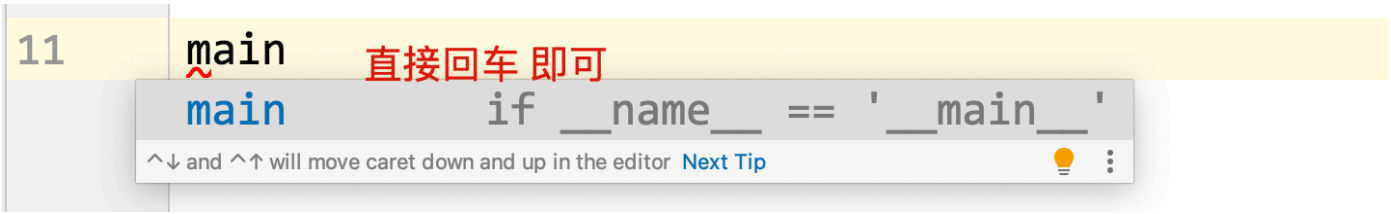
- `xx.py`

```
# import tools
#
# print(tools.sum_2_num(10, 20))


from tools import sum_2_num

print(sum_2_num(10, 20))
```

__name__ 变量

1，导入模块的时候，会执行模块中的代码
2，作用：如果在导入模块的时候，模块中的部分代码不想被执行，可以使用 __name__ 来解决
3，__name__ 变量,是 Python 解释器内置的变量(变量的值是自动维护的)，每个代码文件中,都有这个变量
3.1 在模块中 直接右键运行代码文件，__name__ 变量的值是 '__main__'
3.2 如果是被导入运行代码文件，__name__ 变量的值 是 模块名(文件名)

```
if __name__ == '__main__':
    # 在这个 if 的缩进中书写的代码,导入的时候不会被执行
```



模块的导入顺序

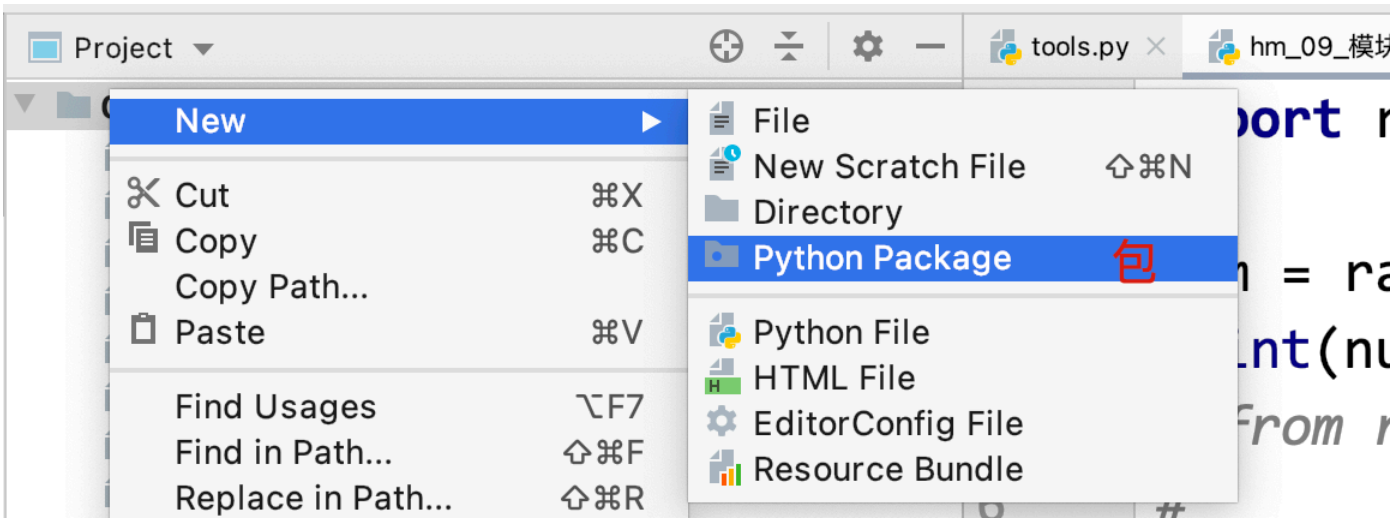
1，在导入模块的时候，会先从代码所在的目录进行导入
2，如果没有找到,回去 Python 系统的目录查找导入
3，如果没有找到，报错

注意点：我们自己定义的代码文件名字 不要和你导入的系统的模块文件名一样

包(package)

包：将多个模块放在一个目录中集中管理，并在这个目录中创建一个 __init__.py 文件(可以什么都不写)，就是一个包

包的创建



包的导入

```
# 方式一
import 包名.模块名
# 使用
包名.模块名.工具名

# 方式二
from 包名 import 模块名
# 使用
模块名.工具名
# 方式三，使用快捷键导包
from 包名.模块名 import 工具名
# 直接使用工具名
工具名
```

案例

案例 模拟开发登录功能并测试

需求：

函数 login

参数

1. 模拟开发人员实现登录功能，定义一个函数能够接收用户输入的用户名、密码、验证码，根据不同的测试数据返回不同的验证结果
2. 已知正确的登录信息（用户名：admin、密码：123456、验证码：8888）

if

login 函数写在这个文件中

3. 要求1：创建login_service.py模块开发登录功能代码，并存放到service包中
4. 要求2：创建test_login.py模块开发测试代码，并存放到script包中
5. 要求3：至少测试3种测试场景

调用 login 函数