day22

今日内容

- 讲作业
  - 栈
  - 顺序查找
- 可迭代对象
- 约束 + 异常
- 反射

内容详细

1.作业

1.1 代码从上到下执行

```
print('你好')

def func():
    pass

func()
```

```
class Foo:
    X = 1

    def func(self):
        pass
Foo.X
```

```
class Foo:
    print('你好')
    def func(self):
        pass
```

```python
class Foo:
    x = 1
    def func(sef):
        pass

    class Meta:
        y = 123
        def show(self):
            pass
```

```python
class Foo:
    print('你')
    x = 1
    def func(sef):
        pass

    class Meta:
        print('好')
        y = 123
        def show(self):
            pass
```

1.2 栈

```python
class Stack(object):
    """
    后进先出
    """
    def __init__(self):
        self.data_list = []

    def push(self, val):
        """
        向栈中压入一个数据（入栈）
        :param val:
        :return:
        """
        self.data_list.append(val)

    def pop(self):
        """
        从栈中拿走一个数据（出栈）
        :return:
        """
        return self.data_list.pop()
```

2.可迭代对象

表象：可以被for循环对象就可以称为是可迭代对象： "x" [11,2] {}...

```
class Foo:
    pass

obj = Foo()
```

如何让一个对象变成可迭代对象?

在类中实现`__iter__`方法且返回一个迭代器（生成器）

```
class Foo:
    def __iter__(self):
        return iter([1,2,3,4])

obj = Foo()


class Foo:
    def __iter__(self):
        yield 1
        yield 2
        yield 3

obj = Foo()
```

记住：只要能被for循环就是去看他内部的iter方法。

3.约束

```
# 约束字类中必须写send方法，如果不写，则调用时候就报抛出 NotImplementedError
class Interface(object):
    def send(self):
        raise NotImplementedError()

class Message(Interface):
    def send(self):
        print('发送短信')z

class Email(Interface):
    def send(self):
        print('发送邮件')
```

```
class Message(object):

    def msg(self):
        print('发短信')

    def email(self):
        print('邮件')
```

```
    def wechat(self):
        print('微信')

obj = Message()
obj.msg()
obj.email()
obj.wechat()
```

```
class BaseMessage(object):
    def send(self,a1):
        raise NotImplementedError('字类中必须有send方法')

class Msg(BaseMessage):
    def send(self):
        pass

class Email(BaseMessage):
    def send(self):
        pass

class Wechat(BaseMessage):
    def send(self):
        pass

class DingDing(BaseMessage):
    def send(self):
        print('钉钉')

obj = Email()
obj.send()
```

4.反射

根据字符粗的形式去某个对象中 操作 他的成员。

- getattr(对象,"字符串") 根据字符粗的形式去某个对象中 获取 对象的成员。

  ```
  class Foo(object):
      def __init__(self,name):
          self.name = name
  obj = Foo('alex')

  # 获取变量
  v1 = getattr(obj,'name')
  # 获取方法
  method_name = getattr(obj,'login')
  method_name()
  ```

- hasattr(对象,'字符串') 根据字符粗的形式去某个对象中判断是否有该成员。

  ```
  #!/usr/bin/env python
  ```

```
# -*- coding:utf-8 -*-
from wsgiref.simple_server import make_server

class View(object):
    def login(self):
        return '登陆'

    def logout(self):
        return '等处'

    def index(self):
        return '首页'


def func(environ,start_response):
    start_response("200 OK", [('Content-Type', 'text/plain; charset=utf-8')])
    #
    obj = View()
    # 获取用户输入的URL
    method_name = environ.get('PATH_INFO').strip('/')
    if not hasattr(obj,method_name):
        return ["sdf".encode("utf-8"),]
    response = getattr(obj,method_name)()
    return [response.encode("utf-8")  ]

# 作用：写一个网站，用户只要来方法，就自动找到第三个参数并执行。
server = make_server('192.168.12.87', 8000, func)
server.serve_forever()
```

- setattr(对象,'变量','值') 根据字符粗的形式去某个对象中设置成员。

```
class Foo:
    pass


obj = Foo()
obj.k1 = 999
setattr(obj,'k1',123) # obj.k1 = 123

print(obj.k1)
```

- delattr(对象,'变量') 根据字符粗的形式去某个对象中删除成员。

```
class Foo:
    pass

obj = Foo()
obj.k1 = 999
delattr(obj,'k1')
print(obj.k1)
```

python一切皆对象

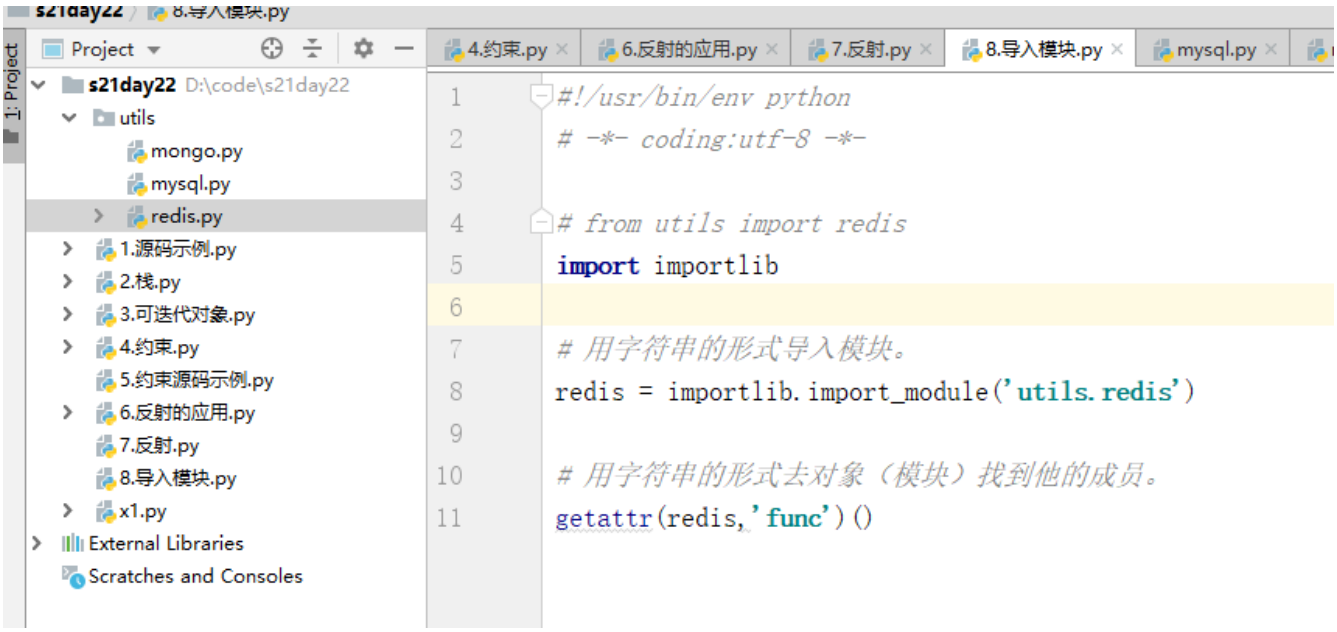
- py文件
- 包
- 类
- 对象

python一切皆对象，所以以后想要通过字符串的形式操作其内部成员都可以通过反射的机制实现。

5. 模块：importlib

根据字符串的形式导入模块。

```
模块 = importlib.import_module('utils.redis')
```



作业

1. 学习笔记
2. 思维导图