

day23

今日内容

- 上节作业
- 单例模式

```
class Foo:
    pass

obj1 = Foo() # 实例, 对象
obj2 = Foo() # 实例, 对象
```

- 日志模块 (logging)
- 程序的目录结构

内容回顾 & 作业

1.字符串格式化

```
msg = "我是%s,年龄%s" %('alex',19,)
print(msg)

msg = "我是%(n1)s,年龄%(n2)s" % {'n1': 'alex', 'n2': 123, }
print(msg)
```

```
# v1 = "我是{0},年龄{1}".format('alex',19)
v1 = "我是{0},年龄{1}".format(*('alex',19,))
print(v1)

# v2 = "我是{name},年龄{age}".format(name='alex',age=18)
v2 = "我是{name},年龄{age}".format(**{'name':'alex','age':18})
print(v2)
```

2.有序字典

```
from collections import OrderedDict

info = OrderedDict()
info['k1'] = 123
info['k2'] = 456

print(info.keys())
print(info.values())
print(info.items())
```

3.作业

3.1 栈和队列

```
class Stack(object):
    pass

class Queue(object):
    pass
```

3.2 反射

```
class Cloud(object):

    def upload(self):
        pass

    def download(self):
        pass

    def run(self):
        # up|C:/xxx/xxx.zip
        # down|xxxx.py
        value = input('请用户输入要干什么? ')
        action = value.split('|')[0]
        # 最low的形式
        if action == 'up':
            self.upload()
        elif action == 'down':
            self.download()
        else:
            print('输入错误')

        # 构造字典 (*)
        method_dict = {'up':self.upload, 'down':self.download}
        method = method_dict.get(action)
        method()

        # 反射 (*)
        method = getattr(self,action) # upload # self.upload
        method()
```

补充:

```
class Foo(object):
    def get(self):
        pass

obj = Foo()
# if hasattr(obj, 'post'):
#     getattr(obj, 'post')

v1 = getattr(obj, 'get', None) # 推荐
print(v1)
```

3.3 循环过程中删除元素: pass

内容详细

1.单例模式（23种设计模式）

无论实例化多少次，永远用的都是第一次实例化出的对象。

```
class Foo:
    pass

# 多例，每实例化一次就创建一个新的对象。
obj1 = Foo() # 实例，对象
obj2 = Foo() # 实例，对象
# 单例，无论实例化多少次，都用第一次创建的那个对象。
obj1 = Foo()
obj2 = Foo()
```

单例模式标准

```
class Singleton(object):
    instance = None
    def __new__(cls, *args, **kwargs):
        if not cls.instance:
            cls.instance = object.__new__(cls)
        return cls.instance

obj1 = Singleton()
obj2 = Singleton()

# 不是最终，加锁。
```

文件的连接池

```
class FileHelper(object):
    instance = None
    def __init__(self, path):
        self.file_object = open(path, mode='r', encoding='utf-8')

    def __new__(cls, *args, **kwargs):
        if not cls.instance:
            cls.instance = object.__new__(cls)
        return cls.instance

obj1 = FileHelper('x')
obj2 = FileHelper('x')
```

2. 模块导入

- 崔楷文事件
- 多次导入重新加载

```
import jd # 第一次加载：会加载一遍jd中所有的内容。
import jd # 由已经加载过，就不在加载。
print(456)
```

```
import importlib
import jd
importlib.reload(jd)
print(456)
```

通过模块导入的特性也可以实现单例模式：

```
# jd.py
class Foo(object):
    pass

obj = Foo()
```

```
# app.py
import jd # 加载jd.py, 加载最后会实例化一个Foo对象并赋值给obj
print(jd.obj)
```

3. 日志（模块 logging）

- 基本应用
- 日志处理本质：Logger/FileHandler/Formatter
- 推荐处理日志方式

```
import logging

file_handler = logging.FileHandler(filename='x1.log', mode='a', encoding='utf-8',)
logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelname)s -%(module)s: %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S %p',
    handlers=[file_handler,],
    level=logging.ERROR
)

logging.error('你好')
```

- 推荐处理日志方式 + 日志分割

```
import time
import logging
from logging import handlers
# file_handler = logging.FileHandler(filename='x1.log', mode='a', encoding='utf-8',)
file_handler = handlers.TimedRotatingFileHandler(filename='x3.log', when='s',
interval=5, encoding='utf-8')
logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelname)s -%(module)s: %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S %p',
    handlers=[file_handler,],
    level=logging.ERROR
)

for i in range(1,100000):
    time.sleep(1)
    logging.error(str(i))
```

注意事项:

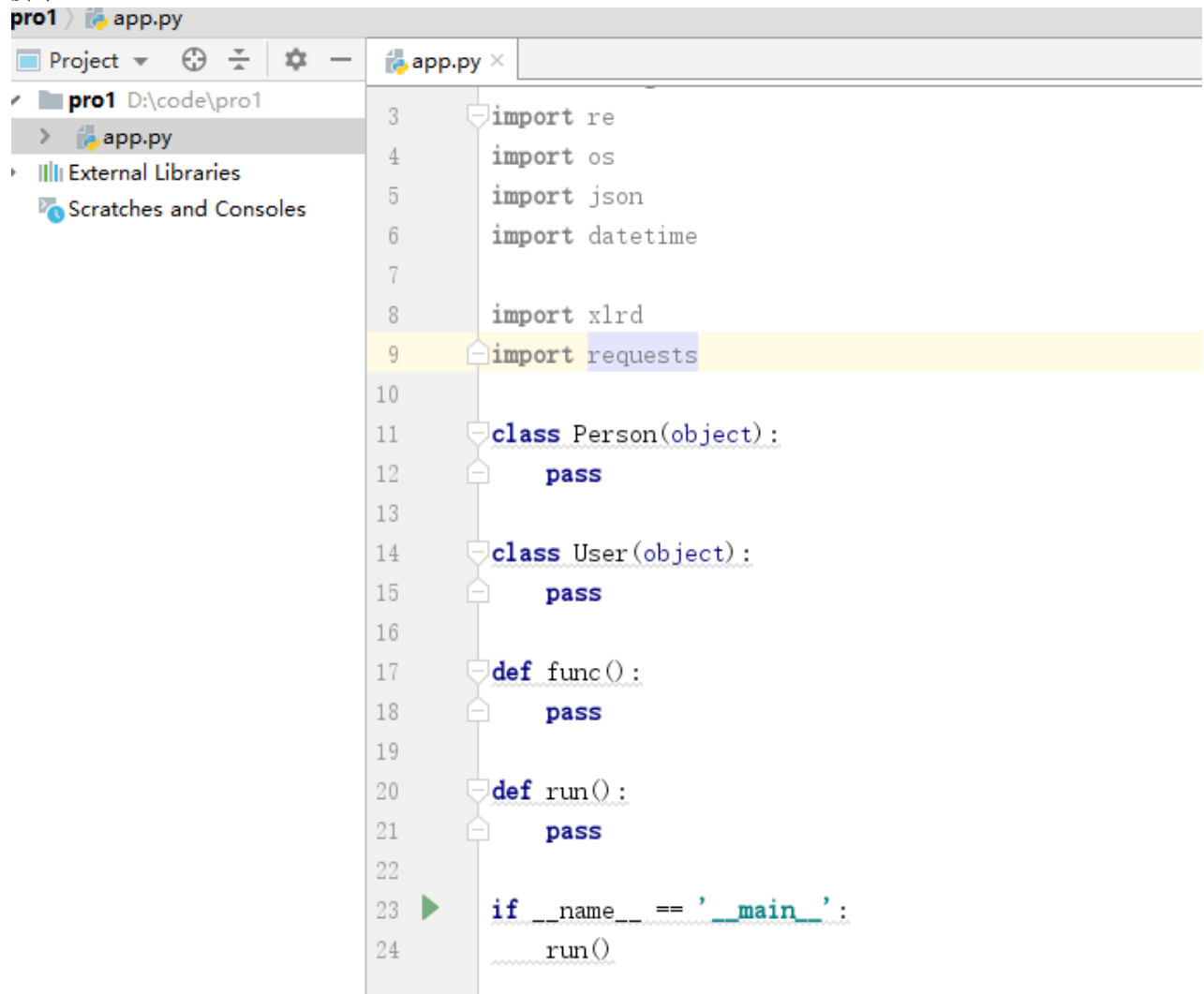
```
# 在应用日志时, 如果想要保留异常的堆栈信息。
import logging
import requests

logging.basicConfig(
    filename='wf.log',
    format='%(asctime)s - %(name)s - %(levelname)s -%(module)s: %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S %p',
    level=logging.ERROR
)

try:
    requests.get('http://www.xxx.com')
except Exception as e:
    msg = str(e) # 调用e.__str__方法
    logging.error(msg,exc_info=True)
```

4.项目结构目录

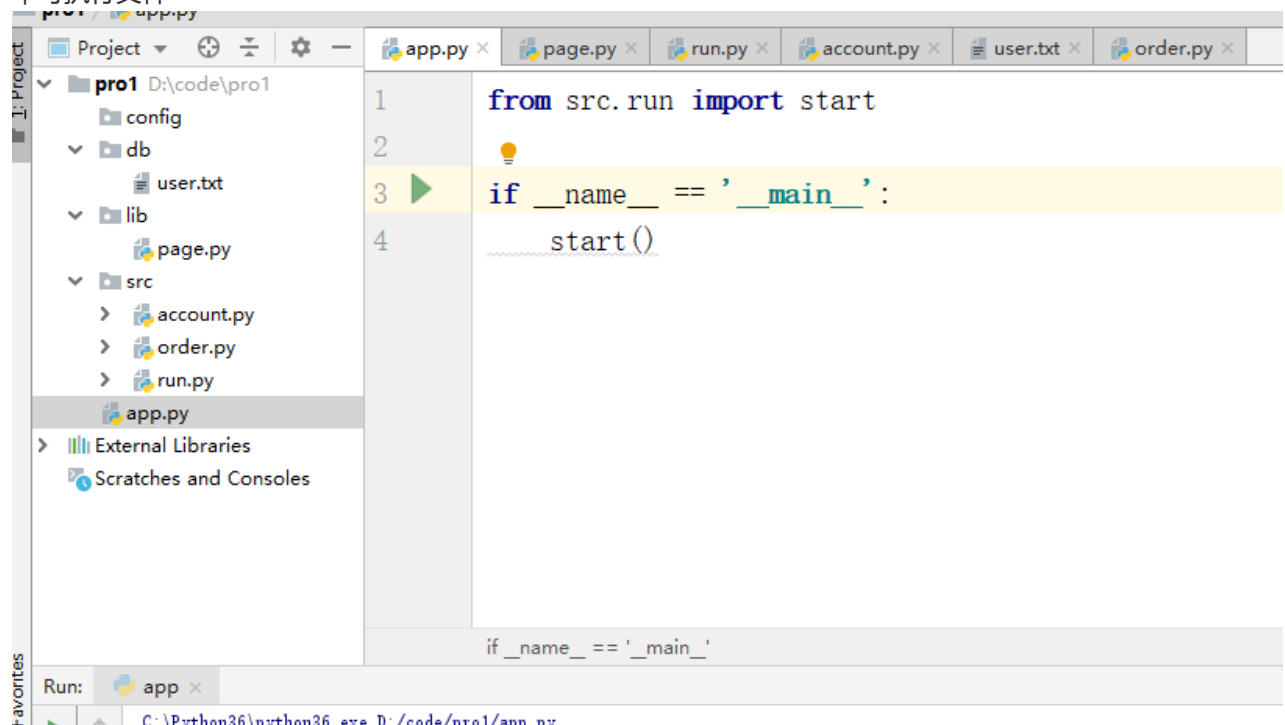
- 脚本



```

3  import re
4  import os
5  import json
6  import datetime
7
8  import xlrd
9  import requests
10
11 class Person(object):
12     pass
13
14 class User(object):
15     pass
16
17 def func():
18     pass
19
20 def run():
21     pass
22
23 if __name__ == '__main__':
24     run()
  
```

- 单可执行文件



```

1  from src.run import start
2
3  if __name__ == '__main__':
4      start()
  
```

if __name__ == '__main__'

Run: app x

C:\Python36\python36.exe D:/code/pro1/app.py

- 多可执行文件

