一、练习题

简答题

1.日志的基本作用有哪些?

```
1.调试程序
2.了解系统程序运行的情况，是否正常
3.程序运行故障分析和问题定位
4.用户行为分析和数据统计
```

2.日志级别有几个?对应的值是多少?

级别	作用	
DEBUG	**调试级别--详细日志--代码调试**	10
INFO	**信息级别--运行过程**	20
WARNING	**警告级别--潜在错误**	**30**
ERROR	**错误级别--异常错误**	40
CRITICAL	**严重级别--严重错误**	50

3.使用basicConfig设置root日志器的输出级别,输出位置,和输入格式?

```
# 设置日志输出级别
logging.basicConfig(level=logging.DEBUG)

# 设置日志输出位置--文件
logging.basicConfig(filename="a.log")

# 设置日志输出格式
logging.basicConfig(format="%(asctime)s %(levelname)s [%(name)s] [%(filename)s(%
(funcName)s:%(lineno)d)] - %(message)s")
```

4.logging四大组件包含哪些内容?作用分别是什么?

组件	类	作用
日志器	Logger	日志入口
处理器	Handler	输出处理
格式化器	Formatter	日志格式
过滤器	Filter	过滤日志

5.logging的日志器/处理器/格式化器的简单使用?

```
# 要求创建3个处理器：输出到文件、控制台、按时间切割日志文件

import logging.handlers

# 获取日志器
logger = logging.getLogger("test")
# 设置日志输入级别
logger.setLevel(logging.DEBUG)

# 获取格式化器
fmtr = logging.Formatter(
    fmt="%(asctime)s %(levelname)s [%(name)s] [%(filename)s(%(funcName)s:%
(lineno)d)] - %(message)s")

# 获取处理器 终端--文件--时间划分文件
shdl = logging.StreamHandler()
# 文件 -- filename -- encoding
fhdl = logging.FileHandler(filename="./log/test.log", encoding="utf-8")
# 时间划分文件 -- filename -- when -- interval -- backupCount -- encoding
tfhdl = logging.handlers.TimedRotatingFileHandler(filename='./log/testtime.log',
when='M', interval=1, backupCount=2, encoding='utf8')

# 处理器添加格式化器
shdl.setFormatter(fmtr)
fhdl.setFormatter(fmtr)
tfhdl.setFormatter(fmtr)

# 日志器添加处理器
logger.addHandler(shdl)
logger.addHandler(fhdl)
logger.addHandler(tfhdl)

# 设置日志输出级别
logger.setLevel(logging.DEBUG)

# 输出日志
logger.debug("debugtest")
logger.info("infotest")
logger.warning("warningtest")
logger.error("errortest")
logger.critical("criticaltest")
```

代码题

1.按下面的要求对day07的代码添加日志文件：

```
"""
1.要求按时间切割日志文件，并且同时输出到控制台；
2.要求使用日志打印出登陆的参数数据；
3.使用日志标记登陆执行开始位置；
"""
```

```
"""
1.新增config.py
```

```
"""
import logging.handlers

def log_config():
    # 获取日志器
    logger = logging.getLogger()
    # 设置日志输入级别
    logger.setLevel(logging.DEBUG)
    # 输出到控制台
    console = logging.StreamHandler()
    # 输出到文件中，全部日志信息
    all_file = logging.handlers.TimedRotatingFileHandler("all.log", when="H",
interval=1, backupCount=3, encoding="UTF-8")

    # 格式化器
    fmt = '%(asctime)s %(levelname)s [%(name)s] [%(filename)s(%(funcName)s:%
(lineno)d)] - %(message)s'
    formatter = logging.Formatter(fmt)

    # 把格式化器添加到处理器中
    console.setFormatter(formatter)
    all_file.setFormatter(formatter)

    # 把处理器添加到日志器中
    logger.addHandler(console)
    logger.addHandler(all_file)
```

```
"""修改测试用例文件"""
# ~/case/test_login.py
import unittest
from parameterized import parameterized
from page.login_page import LoginProxy
from utils import DriverUtils
from config import log_config

log_config()

# 定义测试类
class TestLogin(unittest.TestCase):

    # 类级别的初始化fixture:用来在第一个测试方法前打开浏览器，最大化隐式等待
    @classmethod
    def setUpClass(cls):
        # 获取浏览器驱动对象后为什么要用cls.driver类属性来承接
        # 原因：因为在测试方法中还需要用到大量浏览器驱动对象所提供的方法：元素定位等等
        cls.driver = DriverUtils.get_driver()
        # 实例login_page.py业务层对象
        cls.login_proxy = LoginProxy()

    # 类级别的销毁fixture:执行完所有的测试方法之后关闭浏览器
    @classmethod
    def tearDownClass(cls):
        # 调用工具类中关闭浏览器驱动的方法
        DriverUtils.quit_driver()

    # 方法级别fixture:每个测试方法都需要从首页开始
    def setUp(self):
```

```
        self.driver.get('http://localhost/Admin/Admin/login')

    # 定义测试方法
    @parameterized.expand(DriverUtils.get_json_data())
    def test_login(self, username, password, code, expect):
        logging.info("username={} password={} code={} expect={}
".format(username, password, code, expect))
        # 调用登陆PO文件业务层登陆的方法
        logging.info("--->开始执行登陆操作！！！")
        self.login_proxy.test_login(username, password, code)
        # 断言

 self.assertIn(expect,self.driver.find_element_by_css_selector(".error").text)
```