一、阶段考试

单选题(共15题，每题2分，共30分)

1、web自动化测试按是否查看代码划分，其所属测试分类是?

- [x] A.黑盒测试;
- [ ] B.灰盒测试;
- [ ] C.白盒测试;
- [ ] D.单元测试;

2、自动化测试说法正确的是?

- [ ] A.自动化测试完全替代手工测试;
- [ ] B.自动化测试一定比手工测试厉害;
- [ ] C.自动化可以发掘更多的BUG;
- [x] D.自动化测试可以提高部分项目回归测试测试效率;

3、对于在线安装selenium包说法错误的是?

- [ ] A.pip install selenium默认下载最新的selenium版本;
- [ ] B.pip install selenium=="selenium 版本号"可以下载指定selenium版本;
- [ ] C.pip.exe是python解释器中自带一个可以在线下载、查看、卸载第三方包的工具;
- [x] D.使用pip下载第三方工具时不需要连接外网;

4、以下实例化浏览器驱动对象错误的是?

- [ ] A.driver = webdriver.Chrome()
- [ ] B.driver = webdriver.Fixfox()
- [x] C.driver = webdriver.chrome
- [ ] D.driver = webdriver.Ie()

5、查看下面元素，找出定位方式错误选项?

```
<input id="testA" name="testA" class="testA demoA">demo</input>
```

- [ ] A.driver.find_element_by_id('testA')
- [ ] B.driver.find_element(By.NAME,'testA')
- [x] C.driver.find_element_by_class_name('testA demoA')
- [ ] D.driver.find_element_by_class_name('demoA')

6、需要通过元素定位点击下面的超链接，找出错误选项?

```
<a id="CZBK" name="CZBK" class="CZBK TESTER" href="http://www.itcast.cn/">传智播客</a>
```

- [ ] A.driver.find_element_by_link_text('传智播客').click()

- [ ] B.driver.find_element_by_partail_link_text('智播').click()
- [ ] C.driver.find_element_by_xpath('//*[text()='传智播客']').click()
- [x] D.driver.find_element_by_link_text('播客').click()

7、下面对元素定位说法错误的是：

- [ ] A.当元素定位匹配到多个元素时都是默认返回第一个元素对象；
- [ ] B.在使用partail_link_text、class_name定位时都尽量挑选唯一性较强的局部信息或类名；
- [x] C.元素有id属性时必须使用id定位；
- [ ] D.在实施web自动化元素定位时尽量使用能唯一定位到指定元素方法，保障元素对象的唯一性；

8、定位下面div元素xpath表达式写法有误的是：

```
<div id="CZBK" name="CZBK" class="CZBK TESTER">xpath表达式</div>
```

- [ ] A.//div[@id='CZBK']
- [ ] B.//*[@name='CZBK']
- [ ] C.//*[@name='CZBK' and @id='CZBK']
- [x] D.//*[@text()='xpath表达式']

9、定位下面div元素CSS表达式写法有误的是：

```
<input id="CZBK" name="CZBK" class="CZBK TESTER" tester="this is demo">CSS表达式
</div>
```

- [ ] A. #CZBK
- [ ] B. .TESTER
- [ ] C.[tester*='demo']
- [x] D.[class='CZBK' and tester='this']

10、WebDriver API清除文本内容说法错误的选项是：

- [ ] A.element.clear() 在输入框输入文本时，先执行清除操作，防止输入框默认值影响输入数据；
- [x] B.driver.clear() 清除文本是浏览器驱动对象下的方法；
- [ ] C. 针对元素对象可以执行多次清除文本操作；
- [ ] D.在输入文本前先执行清除操作，可以提高自动化脚本的代码健壮性；

11、以下哪个方法为浏览器的刷新方法：

- [x] A.refresh()
- [ ] B.forward()
- [ ] C.back()
- [ ] D.close()

12、以下哪个方法为浏览器的最大化操作方法：

- [x] A.maximize_window()
- [ ] B.set_window_size(100,100)
- [ ] C.set_window_position(300,200)
- [ ] D.quit()

13、以下哪个方法为获取元素属性的选项是？：

- [ ] A.text
- [ ] B.title
- [ ] C.get_attr()
- [x] D.get_attribute()

14、以下哪个为鼠标悬停方法?

- [ ] A.drop_and_drag()
- [ ] B.drag_and_drop()
- [x] C.move_to_element()
- [ ] D.perform()

15、以下方法对应说明错误的是?

- [ ] A.driver.switch_to.alert 获取js弹出框对象
- [ ] B.driver.switch_to.frame("标签元素对象") 切换frame
- [ ] C.driver.switch_to.window(driver.window_handlers[-1]) 切换最新窗口
- [x] D.driver. switch_to.default_content() 切换到子frame框架

多选题(共10题，每题2分，共20分)

1、下面哪些场景需要或者可以通过自动化进行测试?

- [x] A.某web网站新上线抢购功能，在抢购开始后，出现系统经常崩溃的情况；
- [x] B.某web网站需要保障chrome、Firefox、ie8/9/10、360、Safari等多达10种浏览器能正常使用；
- [ ] C.某web网站首页广告图经常更换，且需要验证图片显示内容；
- [ ] D.某web网站界面交互、风格主题调整版本上线。

2.通常情况下，挑选web自动化测试工具的依据有哪些?

- [x] A.为了节省成本，优先挑选开源测试工具；
- [x] B.基于公司平台所需要支持浏览器种类，尽量挑选支持浏览器多的测试工具；
- [x] C.挑选市场上使用比较多的且功能比较强大的测试工具；
- [ ] D.在无基础的情况下挑选一款学习成本较大且不稳定的新测试工具；

3、请选择出下面哪些项目不适合做web自动化?

- [x] A.项目-1：某公司为了做广告宣传，发布了一个静态广告宣传页；
- [x] B.项目-2：某外包公司承接一个档案管理系统，项目整体周期为期1个月；
- [x] C.项目-3：某电商公司平台还未上线，在基础版本迭代过程中；
- [ ] D.项目-4：某公司平台已上线，主体业务流程已稳定，后续会陆续上线一些独立新功能；

4.[多选]请选择定位下面的input元素的方式效果一样的选项：

```
<div class="navitems2 p" id="navitems5">
    <input id="testA" name="testA" value="all" class="selected">请输入</input>
</div>
```

- [x] A.driver.find_element_by_id("testA") 和 driver.find_element_by_class_name("selected")
- [x] B.driver.find_element(By.XPATH,"//input") 和 driver.find_element_by_css_selector("div input")
- [ ] C.driver.find_elements(By.CSS_SELECTOR,"#testA") 和 driver.find_element_by_name("testA")
- [x] D.driver.find_element_by_xpath("//div[@class='p']/input") 和 driver.find_element_by_xpath("//input")

5.[多选]以下关于driver.close()和driver.quit()区别说法正确的是:

- [x] A.driver.quit()会关闭所相关的窗口，退出驱动;
- [x] B.driver.close()关闭当前窗口，如果当前打开的是最后一个窗口，则退出浏览器;
- [x] C.driver.close()当前打开的是最后一个窗口，则退出浏览器，但是浏览器驱动进程不会关闭;
- [ ] D.同一个浏览器驱动对象下面可以存再多个当前页面;

6.[多选]关于显示等待下面说法错误的是:

- [x] A.显示等待对该设置后所有元素定位都生效，都会执行等待过程;
- [ ] B.显示等待只针对其WebDriverWait(driver, 10, 1).until(lambda x: x.find_element_by_xxx("value")方法中所定位指定元素生效;
- [x] C.显示等待在超过最大时长后如果未找到对应元素，则会抛出异常：NoSuchElementException;
- [ ] D.执行显示等待的方法如能定位到元素，最终返回的是元素对象;
- [ ] E.显示等待需要导包WebDriverWait;

7.[多选]针对下面的元素请选择下面frame切换正确做法:


```
<body>
    <iframe id="login_frame" name="login_frame" height="100%" scrolling="no"
width="100%" frameborder="0"></iframe>
        <html>
            <head>...</head>
            <body>
                <div class="bottom hide" id="bottom_qlogin" style="display:
block;">
                    <a class="link" id="switcher_plogin"
href="javascript:void(0);" tabindex="8">帐号密码登录</a>
                    <span class="dotted" id="docs_dotted">|</span>
                    <a href="https://qzs.qq.com/qzone/v6/reg/index.html"
class="link" target="_blank">注册新帐号</a>
                    <span class="dotted">|</span>
                    <a class="link" id="feedback_qlogin"
href="https://support.qq.com/products/14800" target="_blank">意见反馈</a>
```

```
                </div>
            </body>
        </html>
</body>
```

- [x]
```
# 点击账户密码登陆
driver.switch_to.frame(driver.find_element_by_id('login_frame'))
driver.find_element_by_id("switcher_plogin").click
```

- [ ]
```
# 点击注册新账户
driver.find_element_by_xpath("//*[text()='注册新账号']")
```

- [x]
```
# 点击账户密码登陆
driver.switch_to.frame(driver.find_element(By.NAME,'login_frame'))
driver.find_element_by_css_selector("#switcher_plogin").click
```

- [ ]
```
# 点击意见反馈
driver.switch_to_frame(driver.find_element_by_css_selector("#login_frame"))
driver.find_element_by_id("feedback_qlogin").click
```

8.[多选]以下对于fixture理解正确选项是:

- [x] A.对测试用例环境的初始化和销毁表示一个fixture;
- [x] B.fixture通过不同的控制级别来决定初始化和销毁动作的作用范围;
- [x] C.unittest提供了3种不同的fixutre控制级别:模块级别、类级别、方法级别;
- [x] D.通过fixture可以更好的组织测试用例执行，减少部分代码冗余;

9.[多选]以下对于fixture理解正确选项是:

- [x] A.对测试用例环境的初始化和销毁表示一个fixture;
- [x] B.fixture通过不同的控制级别来决定初始化和销毁动作的作用范围;
- [x] C.unittest提供了3种不同的fixutre控制级别:模块级别、类级别、方法级别;
- [x] D.通过fixture可以更好的组织测试用例执行，减少部分代码冗余;

10.[多选]下面对于多窗口描述错误的是:

- [x] A.使用自动化脚本打开浏览器某页面后，对页面某业务功能按钮执行了点击事件，触发了新的页面窗口打开，可以直接对新页面窗口的元素进行操作;
- [ ] B.selenium通过句柄来实现driver传递;
- [ ] C.句柄:handler，是浏览器窗口的唯一标识码；每个句柄可以对应一个页面对象。
- [x] D.使用自动化脚本打开浏览器某页面后，对页面某业务功能按钮执行了点击事件，触发了页面刷新，也需要做窗口切换。

填空题(共10题，每题2分，共20分)

---

1. **PO模式分为:**[对象库层] [操作层] [业务层];
2. **Unittest核心组件:**[TestCase]、[TestSuite]、[TestLoader]、[TextTestRunner]、[Fixture];
3. **使用Unittest组织测试用例，测试类必须继承** [unittest.TestCase]，**测试方法名必须以** [test] **开头**;

4. **driver.add_cookie() 添加cookie的方法中传递参数为字典对象，必须包含** [value和name];
5. **selenium3.0包含3大核心组件为：**[IDE]、[Grid]、[webdriver];
6. **json文件中的布尔值为：**[ture]、[false];
7. **快捷导包快捷键：**[alt+enter]，**快捷查看方法内部实现代码快捷键：**[ctrl+鼠标左键];
8. **下拉框对象实例化方法：**[select=Select("下拉框标签元素对象")]
9. **定义类方法的装饰器：**[@classmethod];
10. **Fixture控制级别包含：**[类级别、方法级别、模块级别]**三种**;

代码题(30分)

1.针对下面登陆窗口以及html代码编写测试用例，要求如下：

```
1.使用unittest组织测试用例；
2.测试三种登陆异常情况，自选场景，需要使用数据驱动技术；
3.引入fixture只打开一次浏览器，3个测试用例运行完之后关闭浏览器；
4.增加断言；
5.使用日志模块打印参数化数据；
6.使用PO封装界面；
"""
注：验证码默认为123456
"""
```

```
<!--核心元素如下-->
<html>
    <body>
        <iframe id="testA"/>
            <..这里中间元素忽略...>
            <input id='username' name='username' value='q'>请输入用户名</input>
            <input id='password' name='password' value='c'>请输入密码</input>
            <input id='verify_code' name='code' value='e'>请输入验证码</input>
            <button id='submit' name='button' value='c'>登陆</button>
    </body>
</html>
```

```
"""
1.出现异常会弹出自定义弹出框，元素信息为：
<div id='msg' name='msg' value='d'>动态错误提示信息</input>
动态的错误提示信息含：
  请输入账户信息！
  请输入密码！
  账户信息不存在！
  密码错误，请重新输入！
  请输入验证码！
  验证码错误！
  ......
2.有效账户信息为：15800000001   密码123456
3.地址信息为：http://www.test_login.com
"""
```

```
# login_page.py PO文件
class LoginPage:
    def __init__:
        self.driver = DriverUtil.get_driver()
        self.username = (By.ID,"username")
        self.password = (By.ID,"password")
        self.code = (By.ID,"verify_code")
        self.submit_btn = (By.ID,"submit")

    def find_username(self):
        return self.driver.find_element(*self.username)
    def find_password(self):
        return self.driver.find_element(*self.password)
    def find_code(self):
        return self.driver.find_element(*self.code)
    def find_submit_btn(self):
        return self.driver.find_element(*self.submit_btn)

class LoginHandler:
    def __init__(self):
        self.login_page = LoginPage()
    def input_username(self,username):
        self.login_page.find_username().send_keys(username)
    def input_password(self,pwd):
        self.login_page.find_password().send_keys(pwd)
    def input_code(self,code):
        self.login_page.find_code().send_keys(code)
    def click_submit_btn(self):
        self.login_page.find_submit_btn().click()
```

```
class LoginProxy:
    def __init__(self):
        self.login_handler = LoginHandler()
    def test_login(self,username,pwd,code):
        self.login_handler.input_username(username)
        self.login_handler.input_password(pwd)
        self.login_handler.input_code(code)
        self.login_handler.click_submit_btn()
```

```
# test_login.py 用例文件
import unittest
from parameterized import parameterized
from selenium import webdriver
from selenium.webdriver.common.by import By
import json
from login_page import LoginProxy
from utils import DriverUtils
import logging
from config import log_cof

# 执行日志配置方法
log_cof()

# 获取json文件数据
def get_json_data(json_file_path):
    with open(json_file_path, encoding='utf-8') as f:
        login_data = []
        dict_str = json.load(f)
        # 第一次遍历字典，取键值
        for case_data in dict_str.values():
            login_data.append(list(case_data.values()))
        # print(login_data)
    return login_data
# 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):
        cls.driver = DriverUtils.get_driver()
        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
        cls.driver.quit()

    # 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://www.test_login.com')

    @parameterized.expand(get_json_data("data.json"))
    def test_login(self, username, password, code,expect):
        loggin.info("username ={} password={} code={} expect=
{}".format(username, password, code,expect))
```

```
        self.login_proxy.test_login(username, password, code)
        self.assertEqual(expect,self.driver.find_element(By.ID,"msg").text)
```

```
# utils.py 工具类文件
import time
from selenium import webdriver

class DriverUtils:
    # 定义私有变量，用来存储浏览器驱动对象
    __driver = None
     # 获取浏览器驱动对象并且初始化
    # 1.为了方便调用，设置类级别的方法
    # 2.为了保障get_driver在多次调用的时候浏览器驱动对象的唯一性，需要添加判断
    @classmethod
    def get_driver(cls):
        # 如果浏览驱动对象的私有变量__driver=None
        if cls.__driver is None:
            # 实例化浏览器驱动
            cls.__driver = webdriver.Chrome()
            # 窗口最大化
            cls.__driver.maximize_window()
            # 隐式等待
            cls.__driver.implicitly_wait(10)
        # 返回浏览器驱动对象
        return cls.__driver
    # 关闭浏览器驱动对象
    @classmethod
    def quit_driver(cls):
        # 通过获取浏览器驱动对象的方法拿到浏览器驱动对象
        # 为了保障代码的健壮性，添加__driver不为空的判断
        if cls.__driver is not None:
            cls.get_driver().quit()
            # 浏览器驱动对象cls.__driver 在调用quit()之后只是关闭只管所看到的效果，实际里面还有一些缓存信息
            cls.__driver = None
```

```
# data.json 数据文件
{
  "login_no_account": {
    "username": "",
    "password": "123456",
    "code": "123456",
    "expect": "请输入账户信息"
  },
    "login_success": {
    "username": "15800001234",
    "password": "123456",
    "code": "123456",
    "expect": "账户信息不存在"
  },
    "login_success": {
    "username": "15800000001",
    "password": "error",
    "code": "123456",
    "expect": "密码错误"
  }
```

```
}
```

```
# config.py 日志配置文件
# 日志配置方法
import logging.handlers

def log_cof():
    # 创建日志器,配置日志打印级别
    logger = logging.getLogger()
    logger.setLevel(level=logging.INFO)
    # 创建处理器
    lt = logging.handlers.TimedRotatingFileHandler('../log/web_atuo_test.log',
when='midnight', interval=1,
                                                   backupCount=2)
    # 创建输出到控制台的处理器
    ls = logging.StreamHandler()
    # 创建格式化器
    formatter = logging.Formatter(
        fmt='%(asctime)s %(levelname)s [%(name)s] [%(filename)s(%(funcName)s:%
(lineno)d)] - %(message)s')

    # 添加格式化器到处理器
    lt.setFormatter(formatter)
    ls.setFormatter(formatter)
    # 将处理器添加到日志器
    logger.addHandler(lt)
    logger.addHandler(ls)
```