**一、功能手动测试**

文件功能

1. 记录、保存文本文件
2. 打开文件
3. 重命名文件
4. 删除当前文件

编辑功能

1. 撤回
2. 剪切
3. 复制
4. 粘贴
5. 全选
6. 查找

运行功能

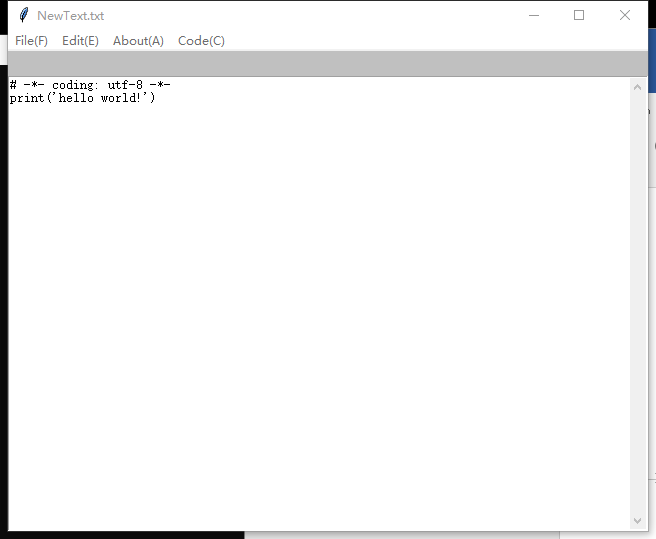
1. 运行.py文件
2. 运行.cpp文件

**测试步骤：**

打开产品软件，输入文本

# -\*- coding: utf-8 -\*-

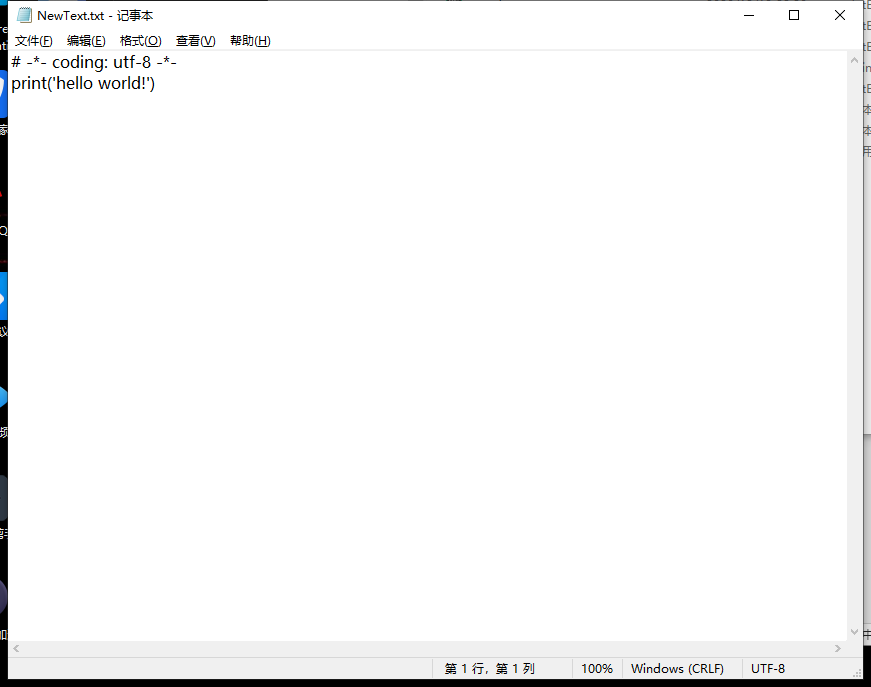
print('hello world!')



通过快捷键方式（如Ctrl+A）测试各种编辑功能，功能均可实现。

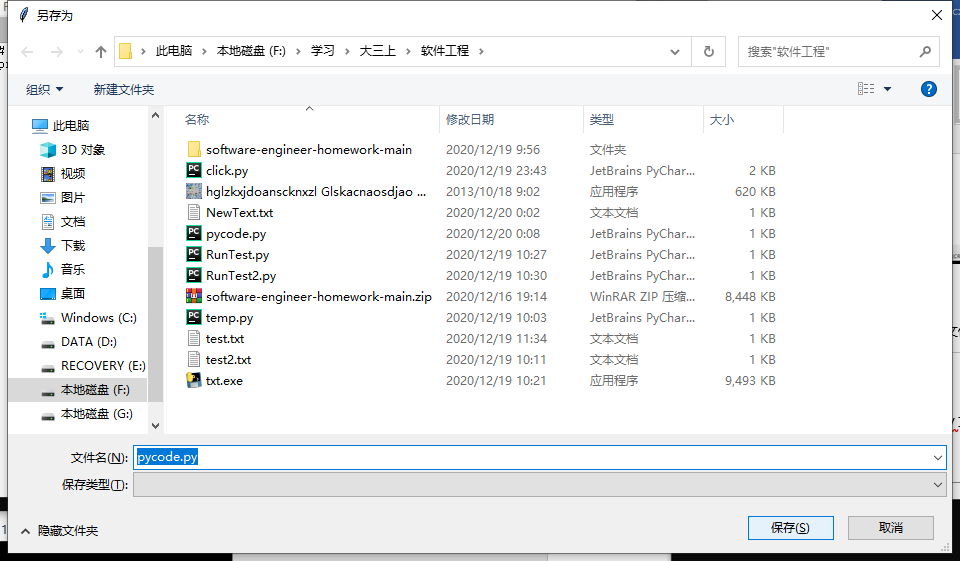
保存文本文件，查看文件所保存的目录，发现文件成功保存。

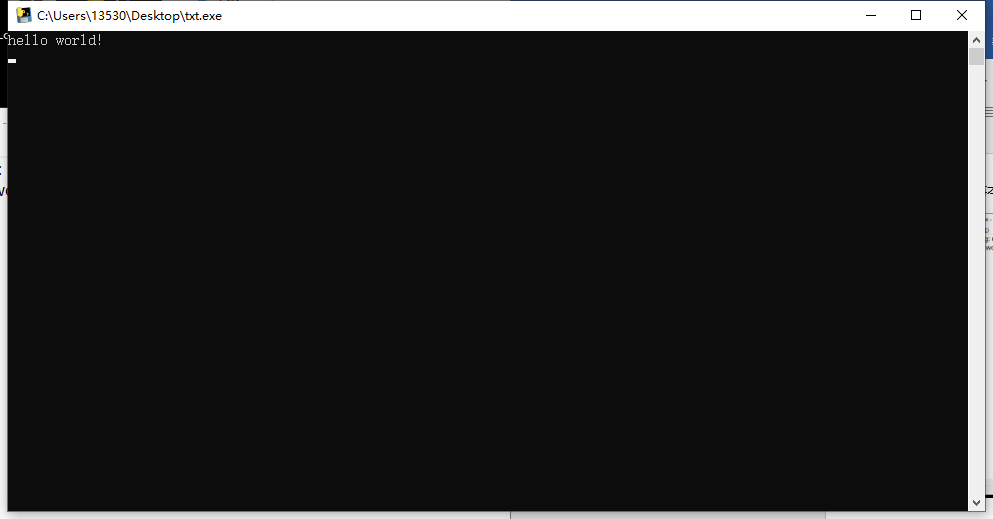
用记事本打开txt文件，文本内容与所编写内容一致，输入文本保存功能测试成功。



关闭产品软件后重新打开，选择刚保存的txt文件打开，文件成功打开。

选择“Code”-“Run Python”，保存生成的py文件，观察命令窗口，窗口成功输出“hello world!”，运行py文件功能测试成功。





**二、两个简陋的自动化测试**

**测试功能：**输入文本并保存

**测试步骤：**自动打开产品文件，在输入框内输入“Testing”，自动保存。

**实现代码：**

# -\*- coding: utf-8 -\*-

"""

Created on Sat Dec 19 09:50:51 2020

@author: 13530

"""

import pyautogui

#打开桌面的exe

pyautogui.moveTo(344, 251, duration=1)

pyautogui.doubleClick()

import time

time.sleep(2)

pyautogui.moveTo(600, 400, duration=1)

pyautogui.click()

pyautogui.typewrite(['T','e','s','t','i','n','g','enter',],'0.1')

pyautogui.typewrite('success!','0.2')

pyautogui.moveTo(530, 240, duration=1)#File

pyautogui.click()

pyautogui.moveTo(549, 323, duration=1)#Save

pyautogui.click()

pyautogui.moveTo(1296, 760, duration=1)#确定

pyautogui.click()

pyautogui.moveTo(1123, 215, duration=1)#关闭

pyautogui.click()

import os

os.system('F:/学习/大三上/软件工程/NewText.txt')

**测试功能：运行.py文件**

**测试步骤：自动打开产品软件，选择一个.py文件打开，运行。**

**实现代码：**

# -\*- coding: utf-8 -\*-

"""

Created on Sat Dec 19 21:48:20 2020

@author: 13530

"""

import pyautogui

import time

#运行py文件测试

pyautogui.moveTo(530, 240, duration=1)#file

pyautogui.click()

pyautogui.moveTo(590, 300, duration=1)#open

pyautogui.click()

pyautogui.moveTo(732, 426, duration=1)#select py

pyautogui.click()

pyautogui.moveTo(1296, 760, duration=1)#确定

pyautogui.click()

time.sleep(2)

pyautogui.moveTo(698, 240, duration=1)#Code

pyautogui.click()

pyautogui.moveTo(721, 277, duration=1)#run py

pyautogui.click()

pyautogui.moveTo(1280, 757, duration=1)#保存

pyautogui.click()

"""

pyautogui.moveTo(1020, 532, duration=1)#若有重复文件则替换原有文件

pyautogui.click()

"""

pyautogui.moveTo(383, 308, duration=1)#点击命令窗口观察py运行结果

pyautogui.click()

**测试心得**

第一次听到自动化测试这个概念时，我的想法就是通过一个脚本来测试产品的功能，因此有了用python来实现的念头。可我对python不熟悉，当开始将想法变成现实时，遇到了种种问题，以致到最后只做出了简陋的自动化测试脚本。

在制作测试脚本的过程中，我去查阅了相关资料，去了解一个项目的自动化测试是如何做到的，以下为我查阅到比较详细的测试分析与设计

* **测试分析：**

**1. 我们需要做什么？**

1. 把明确的需求点转换成测试项
2. 缺陷预防

**2. 怎么做？**

1. 整体模块分析
2. 逻辑分析【这一点主要是从产品实现的原理上去分析可能的影响】

* 怎么做？
* 开发的设计文档
* 补充和挖掘测试点

1. 全部服务的异常监控、服务重启
2. 各类存储对空间的占用、占满、是否需要做存储的接口测试
3. 所有类型的管理员、操作权限测试、支持的多少管理员并发操作
4. 对流程图的挖掘 -- 流程图全部流程测试、流程图重要的节点异常测试
5. 对状态的挖掘 -- 所有状态的相互转化需要覆盖全、状态转化是否合理、每一个状态下哪些操作可做哪些不可做，多个状态是否可以共存
6. 对关联项的挖掘 -- 流程进展到哪一步关机重启/服务重启、和备份配置的关联，和操作日志的关联等等
7. 任务的并发操作测试、是否可配置、是否会出现性能不足，是否符合用户场景
8. 异常处理机制测试，异常处理机制是否完善
9. 指标测试，开发的指标设计是否合理

* 修正不合理的需求
* 如何分析
* 逻辑原理：

1. 该模块是否涉及到一些全新的概念(比如我们的 bbc 全量包)，需要明确?
2. 该模块包括哪些服务?
3. 该模块涉及到哪些存储技术(如 mysql、dap、redis)?具体怎么存储的?占用大小如何?
4. 该模块的操作流程有哪些?是否有子流程图?
5. 该模块是否有多个状态的转化?是否有明确的状态转化图?
6. 该模块对多个管理员是否区分，管理员权限如何设计?
7. 该模块是否有一些特殊的操作限制?操作限制是否有明确的表格?
8. 该模块的任务是否有并发需求?并发的设计?
9. 该模块的所有指标如何?
10. 该模块是否有异常处理机制?在设备各种异常时，该模块的设计是否满足能稳健运行?

* 场景分析

1. 从用户的使用习惯和使用方法去分析影响
2. 检查当前案例是否覆盖到用户场景

* 关联测试分析：

1. 考虑你的模块所在整个系统的地位，分析上下游的影响
2. 对老功能的影响

* 经验补充分析

1. 版本分析
2. 模块分析

* 输出

1. 测试项
2. 补充测试地图

* **测试设计：**

1. **需要做什么？**

* 把测试项细化成测试点
* 缺陷预防

**2. 需要做什么？**

* 基本设计方法

1. 等价类划分法【将输入域和输出域划分为不同的等价类，等价类之内的操作结果相同】，使用范围：显示输入框输入
2. 边界值法【需要结合等价类划分法方法，在划分出来的等价类选取有代表性的值】
3. 正反对比【一般会放到同一个用例里覆盖】
4. 字符多样性【考虑不同字符的输入】
5. 测试类型

* 产品专项测试
* 正交组合设计【正交矩阵，覆盖各个参数间的组合情况】
* 业务逻辑设计【根据业务设计测试点】

**3. 输出：**

* 基本测试点