



北京邮电大学
Beijing University of Posts and Telecommunications

程序设计与实践

领域特定语言的脚本解释器

实验报告

姓名： 张明昱 2019211590

班级： 2019211310

时间： 2021/12/20

目录

一、 功能需求与分析.....	3
二、 数据结构说明:	3
三、 模块划分.....	4
1、划分.....	4
2、示意图.....	4
四、 接口.....	5
1、 程序间接口:	5
2、 人机接口:	5
五、 功能.....	6
六、 测试.....	6
1、 理解:	6
2、 Parser 单元测试 unittest.....	6
3、 Parser 自动测试脚本.....	6
4、 Interpreter 自然语言分析测试桩.....	7
5、 Interpreter 媒体测试桩.....	7
6、 Robot 解释器自动测试脚本.....	7
七、 总结与分析.....	8
1、 在进行 Parser 测试时遇到的困难:	8
2、 测试倒逼函数耦合性的启示:	8
3、 silence 判断:	8
八、 附录: TSL 语言 Token 说明文档.....	8
TSL(Ticket Selling Language)售票语言语法描述.....	9

一、功能需求与分析

总目标：设计实现一个领域特定语言 DSL 和它的脚本解释器

1、确定领域：电话售票订票领域，可以实现功能：欢迎致电，查询在售剧目，买票，预定活动，退票，投诉，人工客服，告别语等。

2、初步定义脚本文本：根据功能需求设定一些语言的实例：①国家大剧院②北京人艺③蜂巢剧场。

3、分析 Parser：可以看出，parser 需要实现输入一个脚本语言文件，对文件中的语句进行分析拆解和存储，从而得到针对这个 DSL 文件的语法树。可以大致分为：①文件处理拆分部分，把需要的信息从文件传入内存并用适当的数据形式（list、class.....）进行存储；②分支处理部分，根据语法特点，按照每行的首个关键字对数据进行处理和存储。最终得到语法树。

4、分析 Interpreter：interpreter 解释器需要通过 parser 得到的语法树和用户的输入数据进行程序的实际执行。可以大致分为：①确定运行环境：用户的信息和语法树，均通过与用户的交互询问获得；②while 循环执行程序

5、接口设置：①程序间接口：Parser 和 Interpreter 之间需要的数据传递就是语法树和运行环境，即 Parser 需要 Interpreter 把用户信息和脚本文件传递给自己以构造出语法树，Interpreter 需要经过 Parser 得到语法树用于解释语言。也因此可以得到 Parser 函数的参数和返回值。②人机接口：通过 input 和 print 输出提示语实现交互。

6、测试分析：①对 Parser 的主要函数进行必要的单元测试，再对 Parser 创建自动测试脚本以便进行回归测试。②对 Interpreter 设置两个测试桩，一个自然语言分析测试桩：简单关键字匹配返回用户意愿，一个媒体服务器测试桩：播放内容输出到文件，从测试数据读入应答文本。③自动测试脚本：实现对一系列测试文件的一次性测试，无需输入直接显示是否通过测试，便于进行回归测试。

二、数据结构说明：

①Person 类：脚本执行环境的一部分

```
class Person:
    """记录每个用户的姓名，购票数和订购数"""
    name=""
    ticket=0
    order=0
```

②Step 类：对每个步骤的相关信息存储

```
class Step:
    """记录每个 step 中的信息"""
    stepId=""
```

```

output=list()#要说的一系列 string 数组
branchTable=dict()#key 是听取到的 answer, value 是 stepid
Silence='silenceProc'
Default='defaultProc'
isEnd=0#标识是否为终结步骤
isDeal=0#标记是否为处理步骤

```

③Scrip 类:在 Interpreter 里记录用户名称和当前 step

```

class Script:
    """脚本的运行环境"""
    person=Person()
    stepNow=Step("") #当前 step

```

④Tree 类: 用于储存语法树的类

```

class Tree:
    """存储语法树的结构"""
    idNow='',
    stepTable=dict()

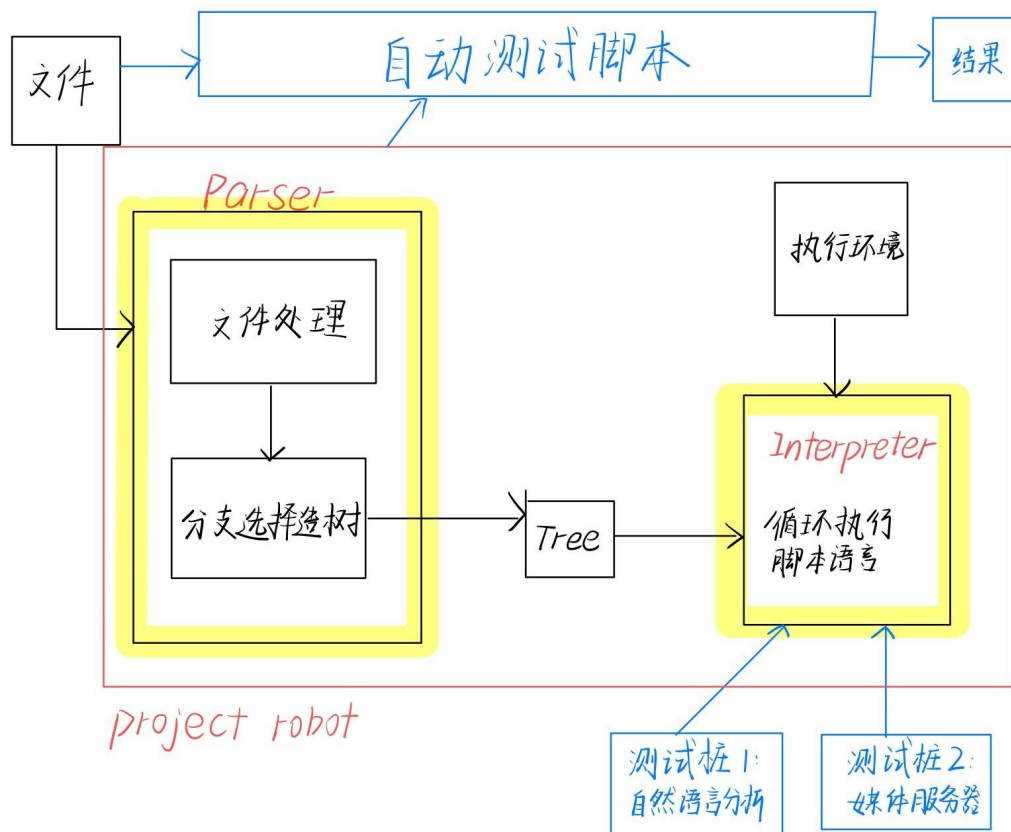
```

三、模块划分

1、划分

整个项目分为实现部分和测试部分。其中实现部分又由 Parser 和 Interpreter 两个部分组成

2、示意图



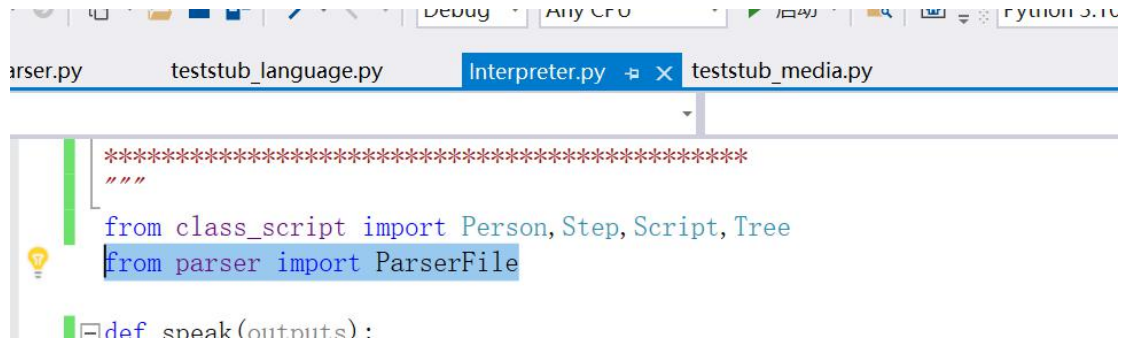
四、接口

1、程序间接口：

①Parser 和 Interpreter 之间需要的数据传递就是语法树和运行环境，即 Parser 需要 Interpreter 把用户信息和脚本文件传递给自己以构造出语法树，Interpreter 需要经过 Parser 得到语法树用于解释语言。也因此可以得到 Parser 函数的参数和返回值：

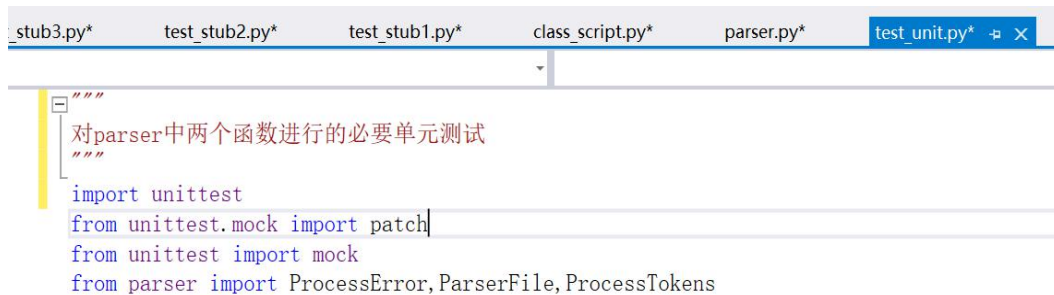
```
def interpreter(name, theatre):  
    tree=ParserFile(theatre, name)  
    script=Script()  
    script.person.name=name
```

②由于 Interpreter 和 Parser 分别放在两个项目中，按照 python 语言的性质，要在 Interpreter 中调用 Parser 就要 import Parser：



```
*****  
"""  
from class_script import Person, Step, Script, Tree  
from parser import ParserFile  
  
def speak(outputs):
```

③在测试文件中，要测试特定函数的功能，也要从相应模块里 import 函数，例如在 Parser 的单元测试内：



```
"""  
对parser中两个函数进行的必要单元测试  
"""  
import unittest  
from unittest.mock import patch  
from unittest import mock  
from parser import ProcessError, ParserFile, ProcessTokens
```

2、人机接口：

通过 input 和 print 函数进行输入输出以完成人机接口。例如 interpreter 中：

```
def main():  
    #进行初始化  
    print("\n*****欢迎使用语音售票服务系统***** \n")  
    name=input(' 请输入您的姓名: ')  
    theatre=input("请输入您要拨号的剧院：（国家大剧院/北京人艺/蜂巢剧场）")  
    theatre+=" ".txt  
    tree=ParserFile(theatre, name)#调用parser得到语法树  
  
    interpreter(name, theatre)
```

五、功能

- 1、parser 的功能：parser 需要实现输入一个脚本语言文件，对文件中的语句进行分析拆解和存储，从而得到针对这个 DSL 文件的语法树。可以大致分为：①文件处理拆分部分，把需要的信息从文件传入内存并用适当的数据形式（list、class.....）进行存储；②分支处理部分，根据语法特点，按照每行的首个关键字对数据进行处理和存储。最终得到语法树。
- 2、Interpreter 的功能：nterpreter 解释器需要通过 parser 得到的语法树和用户的输入数据进行程序的实际执行。可以大致分为：①确定运行环境：用户的信息和语法树，均通过与用户的交互询问获得；②while 循环执行程序
(此处功能为解释器功能，TSL 语言的功能详见附录 TSL 说明文档)

六、测试

1、理解：

程序由两个部分组成，一个是 Parser 一个是 Interpreter，所以要对两个部分分别进行测试，也要对整体进行测试。选用 python 的 unittest 模块和 mock 框架进行测试。

2、Parser 单元测试 unittest

在 Parser 中进行两个单元测试，一个测试 ParserFile 函数，用 mock 模拟 ParserLine 函数，通过判断调用次数与预期值是否相等来进行测试。另一个测试 PcocessToken，查看是否被正确调用。

```
@mock.patch('parser.ParserLine')
def test_ParserFile(self, mockpl):
    """测试其中的ParserLine函数是否被正确调用"""
    ParserFile("test.txt", "Mr. Foo") #test.txt中有9行数据，其中第一行为注释
    self.assertEqual(mockpl.called, True)
    self.assertEqual(mockpl.call_count, 8)

def test_ProcessTokens(self):
    """测试分支处理是否根据tokens0正确进行"""
    with mock.patch('parser.ProcessStep') as mockps:
        ProcessTokens(['step', 'welcome'], 'Mr. Foo')
        self.assertEqual(mockps.called, True)
    with mock.patch('parser.ProcessBranch') as mockpb:
        ProcessTokens(['branch', '购票', 'purchaseProc'], 'Mr. Foo')
        self.assertEqual(mockpb.called, True)
```

3、Parser 自动测试脚本

对 Parser 进行整体测试，将实际获得的 tree 与预先存入的正确 Tree 进行比较，assert 断言判断测试。

```

tree_stan.stepTable["goodbye"].output=['感谢您的来电，再见！']
tree_stan.stepTable["goodbye"].branchTable={}
tree_stan.stepTable["goodbye"].Silence='silenceProc'
tree_stan.stepTable["goodbye"].Default='defaultProc'
tree_stan.stepTable["goodbye"].isEnd=1
tree_stan.stepTable["goodbye"].isDeal=0

```

将得到的语法树与标准正确语法树进行对比

```

for stepname in stepnames:

    if(tree.idNow!=tree_stan.idNow):equal=0
    else:
        if(tree.stepTable[stepname].__dict__!=tree_stan.stepTable[stepname].__dict__):

            if(equal):print('Equal!')

```

4、Interpreter 自然语言分析测试桩

```

def test_haveNextStep1(self):
    mockStepNow=mock.Mock()
    mockStepNow.branchTable={'上映剧目':'showProc','订票':'purchaseProc','退票':'refundProc'}
    mockStepNow.Silence='silenceProc'
    mockStepNow.Default='defaultProc'
    self.assertEqual(haveNextStep('上映',mockStepNow),'showProc')
    self.assertEqual(haveNextStep('剧目',mockStepNow),'showProc')
    self.assertEqual(haveNextStep('退票',mockStepNow),'refundProc')
    self.assertEqual(haveNextStep(' ',mockStepNow),'silenceProc')
    self.assertEqual(haveNextStep('foo',mockStepNow),'defaultProc')

```

5、Interpreter 媒体测试桩

teststub_language.py Interpreter.py teststub_media.py

```

from unittest import mock
from unittest.mock import patch
from Interpreter import interpreter

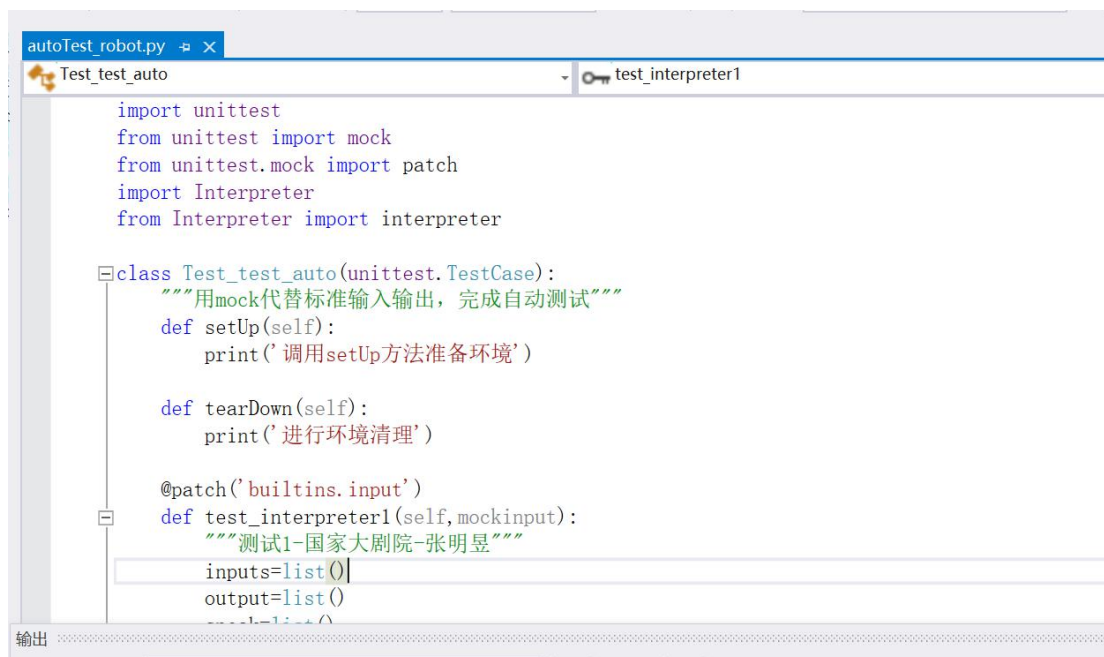
class Test_test_1(unittest.TestCase):
    """用mock代替标准输入输出"""
    def setUp(self):
        print('调用setUp方法准备环境')

    def tearDown(self):
        print('进行环境清理')

    @patch('builtins.input')
    def test_interpreter1(self, mockinput):
        """测试1"""
        inputs=list()
        output=list()
        speak=list()

```

6、Robot 解释器自动测试脚本



```
autoTest_robot.py x
Test_test_auto test_interpreter1

import unittest
from unittest import mock
from unittest.mock import patch
import Interpreter
from Interpreter import interpreter

class Test_test_auto(unittest.TestCase):
    """用mock代替标准输入输出，完成自动测试"""
    def setUp(self):
        print('调用setUp方法准备环境')

    def tearDown(self):
        print('进行环境清理')

    @patch('builtins.input')
    def test_interpreter1(self, mockinput):
        """测试1-国家大剧院-张明昱"""
        inputs=list()
        output=list()
        mock=list()
```

输出

七、总结与分析

1、在进行 Parser 测试时遇到的困难：

首先是要判断两个自定义的类结构语法树 tree 相等的问题，查阅后发现需要在类的定义里重定义 eq 方法。定义后发现仍然无法判断相等，于是逐个分析两个类的实例中的所有属性，属性一 string 经判断相等，另一个属性 dict 字典类型使用 print(__dict__) 后注意到输出的 key 顺序不同，怀疑可以是字典 key 的问题。循环所有 key 比较同一个 key 中的 value 值，发现仍然全部相同，于是确定是字典中 key 的排序问题。而由于字典是散列表无法排序，所以改为逐个 key 判断 value 相等，所有均相等则认为类 Tree 的两个实例相等。

2、测试倒逼函数耦合性的启示：

在 interpreter 中的 haveNextStep 函数中，要根据用户输入 input 确定下一步的 step。本来函数设置的三个参数是 input、语法结构树 tree、脚本 script，运行时也并无问题。在测试时，发现要 mock 的结太复杂，于是返回观察函数，发现其实只要传入 tree 的 stepTable 和 script 的 stepNow 就足以完成功能，于是进行更改。

3、silence 判断：

在 silence 判断时，由于使用了输入代替真正的语音交互，而且又用字符串匹配进行识别，所以要模拟 silence 沉默情况不能直接回车，而是要输入一个空格。

八、附录：TSL 语言 Token 说明文档

TSL(Ticket Selling Language)售票语言语法描述

一、简介：

TSL 语言的作用领域是剧院用于电话购票预定的语言，可以实现基础语音应答回复、买票、预定活动、退票、取消预定、投诉处理并把客户的相关信息进行存储等诸多功能的实用性语言。

二、保留字：

关键字	语义
step	完整表示一个步骤的所有行为
output	输出一段文字
listen	听取用户需求，调用自然语言分析进行处理
branch	对用户的意愿进行分支处理并进行 step 的跳转
default	用户输入没有匹配的情况
Silence	用户沉默的跳转
exit	结束对话

三、注释

TSL 语言支持以'#'符号开始进行注释。注释位置可以单独占据一行，或者在语句后直接进行注释。需要注意的是，在语句后的直接注释要用一个空格进行分隔。

四、基础语法

语言用关键词“step”定义类似函数性质的结构体，在每个 step 内完成相应操作，用 branch 实现函数间的跳转。**TSL 语言的默认开始 step 名称是 welcome。**

五、关键词语义和语法

(1) step: step 函数名

step 用于函数定义，在关键字 step 和函数名之间用空格分开。Step 分为两类，一是交互 step，二是数据处理 step。

①交互 step 的一般格式为：

```
step xxx
output xxx
listen
branch xxx
silence xxx
default xxx
exit
```

②数据处理 step 的一般格式为：

```
step xxx
output
$xxx
exit
```

(2) output: output xxx + xxx

output 实现输出，在 output 和输出语句之间用空格分开。变量和输出语句用“+”连接，需要注意的是，“+”前后也要有空格

(3) listen: listen

listen 实现用户输入，只能单独使用。

(4) branch: branch 关键字 函数名

branch 实现根据用户输入进行函数的跳转，用户输入内容是关键字的子集，函数名是需要用 step 定义的函数名称

(5) silence: silence 函数名

对用户沉默情况进行处理，当沉默时则进入沉默处理 step。需要注意的是，由于程序使用文字模拟语音输入，所以要模拟 silence 沉默情况不能直接回车，而是要输入一个空格。

(6) default: default 函数名

对用户不能识别的输入进行处理。

(7) exit: exit

单独出现，判断函数结束退出。

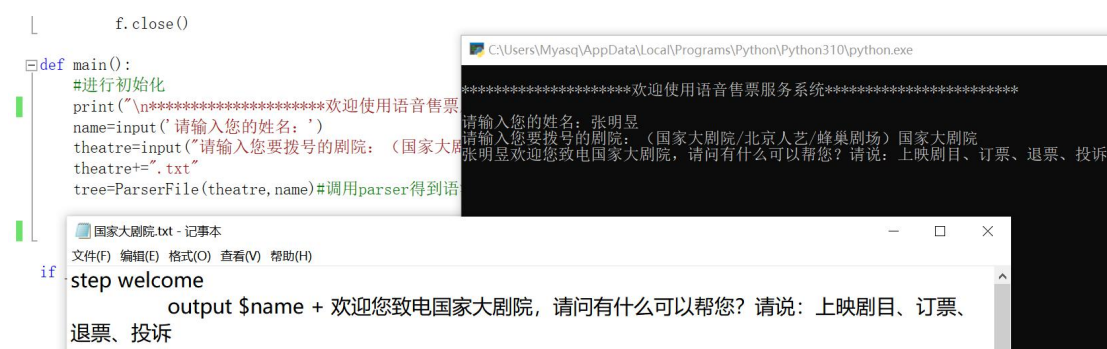
(8) \$ticket++/\$ticket--/\$order++/\$order--

数据处理语句，会根据\$后的内容进行相应的信息更改，并在程序运行结束后存入文件。

六、实现功能

1、欢迎功能：

获取到用户名称后，程序会根据用户名称和调用的脚本输出相应的欢迎语，一个示例：



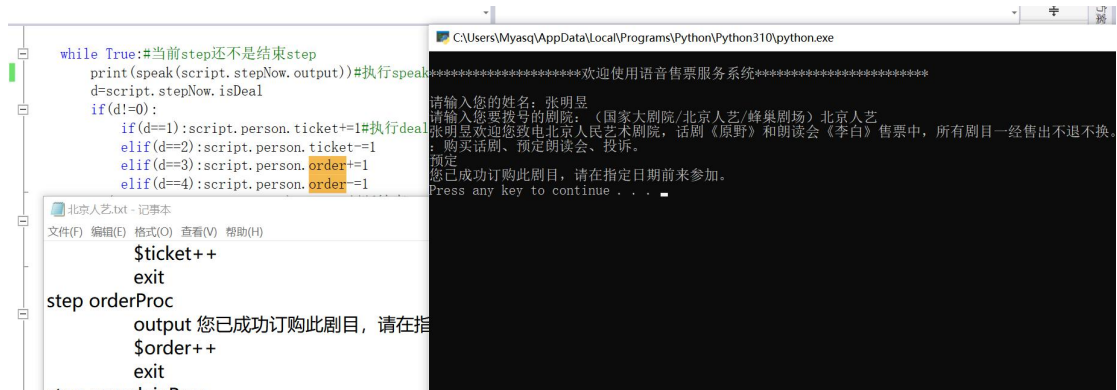
2、售票功能

在进入脚本并选择订票后，程序可以进行购票，输出成功语句并更改用户的属性，并在程序运行结束后存入文件。一个示例：



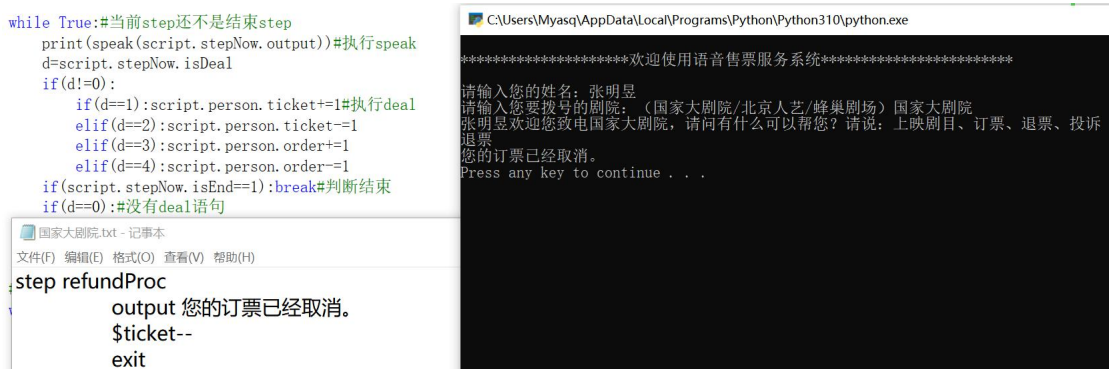
3、预定功能

在进入脚本选择预定活动后，程序可以预定活动，更改 i 用户 order 属性并存入文件。
一个示例：



4、退票功能

选择退票后，程序可以更改相应属性并存入文件



5、存储功能

程序完成后将把个人的订票信息存入文件 `informatio` 中，一个示例：

```
script.stepnow=tree.stepable[navenextStep(myinput, script.stepnow)]##获取下一步step

#将订票信息输出到information文件
with open("information.txt", "a") as f:
    f.write(script.person.name)
    f.write("\n")
    f.write(str(script.person.order))
    f.write("\n")
    f.write(str(script.person.ticket))
    f.write("\n")
    f.close()
```

information.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

张明昱
1
0

6、投诉功能

投诉处理，会要求补充信息，处理后直接退出，一个示例：

```
exit
step complainProc
    output 感谢您的建议，您的反馈我们已经收到
    listen
    silence silenceProc
    default goodbye
step silenceProc
    output 对不起我没有听到您的声音，可以请您重复一遍吗？
    branch 上映剧目 showProc
```

*****欢迎使用语音售票服务系统*****

请输入您的姓名：张明昱
请输入您要拨号的剧院：（国家大剧院/北京人艺/蜂巢剧场）国家大剧院
张明昱欢迎您致电国家大剧院，请问有什么可以帮您？请说：上映剧目、订票、退票、投诉、投诉
感谢您的建议，您的反馈我们已经收到，请问您还有什么要补充的吗？
感谢您的来电，再见！
Press any key to continue . . .

7、沉默处理功能

当用户沉默时（输入一个空格），会进入沉默处理，一个示例：

```
step silenceProc
    output 对不起我没有听到您的声音，可以请您重复一遍吗？
    branch 上映剧目 showProc
    branch 订票 purchaseProc
    branch 退票 refundProc
    branch 投诉 complainProc
    silence silenceProc
    default defaultProc
step defaultProc
    output 对不起我没有理解，您能换一种方式表达吗？请说：上映剧目、订票、退票、投诉、退出
```

选择C:\Users\Myasq\AppData\Local\Programs\Python\Python310\python.exe

*****欢迎使用语音售票服务系统*****

请输入您的姓名：张明昱
请输入您要拨号的剧院：（国家大剧院/北京人艺/蜂巢剧场）国家大剧院
张明昱欢迎您致电国家大剧院，请问有什么可以帮您？请说：上映剧目、订票、退票、投诉、投诉
对不起我没有听到您的声音，可以请您重复一遍吗？

8、错误处理功能

当用户输入无法被识别时，会进入 default 错误处理，要求用户重新输入

```
step defaultProc
    output 对不起我没有理解，您能换一种方式表达吗？请说：上映剧目、订票、退票、投诉、退出
退出
    branch 上映剧目 showProc
    branch 订票 purchaseProc
    branch 退票 refundProc
    branch 投诉 complainProc
    branch 退出 goodbye
    silence silenceProc
    default defaultProc
step goodbye
    output 感谢您的来电，再见！
```

C:\Users\Myasq\AppData\Local\Programs\Python\Python310\python.exe

*****欢迎使用语音售票服务系统*****

请输入您的姓名：张明昱
请输入您要拨号的剧院：（国家大剧院/北京人艺/蜂巢剧场）国家大剧院
张明昱欢迎您致电国家大剧院，请问有什么可以帮您？请说：上映剧目、订票、退票、投诉、退出
对不起我没有理解，您能换一种方式表达吗？请说：上映剧目、订票、退票、投诉、退出

七、示例

```
step welcome
    output $name + 欢迎您致电国家大剧院，请问有什么可以帮您？
    listen #test
    branch 上映剧目 showProc
    branch 订票 purchaseProc
    branch 退票 refundProc
```

```

    branch 投诉 complainProc
    silence silenceProc
    default defaultProc
step showProc
    output 正在上演的剧目是舞剧《冼星海》，请问您是否要进行购票？请说：是、否
    listen
    branch 是 purchaseProc
    branch 否 goodbye
    silence silenceProc
    default defaultProc
step purchaseProc
    output 您已成功购买此剧目，请在指定日期前来取票观看。
    $ticket++
    exit
step refundProc
    output 您的订票已经取消。
    $ticket--
    exit
step complainProc
    output 感谢您的建议，您的反馈我们已经收到，请问您还有什么要补充的吗？
    listen
    silence silenceProc
    default goodbye
step silenceProc
    output 对不起我没有听到您的声音，可以请您重复一遍吗？
    branch 上映剧目 showProc
    branch 订票 purchaseProc
    branch 退票 refundProc
    branch 投诉 complainProc
    silence silenceProc
    default defaultProc
step defaultProc
    output 对不起我没有理解，您能换一种方式表达吗？请说：上映剧目、订票、退票、
    投诉、退出
    branch 上映剧目 showProc
    branch 订票 purchaseProc
    branch 退票 refundProc
    branch 投诉 complainProc
    branch 退出 goodbye
    silence silenceProc
    default defaultProc
step goodbye
    output 感谢您的来电，再见！
    Exit

```
