实验报告 ps2

【题意理解】

Problem1-3:

一：实现一个小游戏 Hangman （刽子手），游戏过程大致理解如下：

1. 计算机从 word.text 中随机抽取一个单词，secret_word 表示，并告知玩家字符串长度。

2. 用户一个字符一个字符去猜，手动输入猜测结果，总共有 6 次猜测次数，每次猜测完之后，计算机告知玩家已经猜测的结果和剩余可以猜测的字母范围，这个范围用函数 get_available_letters(letters_guessed) 求得。

3. 玩家猜测过程是有一定的规则和违规的相应惩罚措施的：

（1）：不能猜测已经猜测过的字符，每重复猜测一次，计算机发出警告一次，总共有 3 次警告机会；不能猜测除了大小写字母之外的字符，如‘#’ 等一些特殊字符，每猜测一次，警告次数减 1；当警告次数没有了，玩家如果继续犯规（猜测重复，或者猜测除大小写字母之外的字符）则猜测次数减去 1；

（2）：玩家猜测的字母不在目标串 secret_letters 中，如果猜测的字母是元音字母，则猜测次数减 2 作为惩罚，如果是辅音字母，则猜测次数减 1；

（3）：玩家猜测的字母在目标串 secret_letters 中，则猜测次数不变（不减），继续猜测；

4. 游戏的两种结果：一是玩家成功，即玩家在 猜测次数耗尽之前猜对目标字符串中所有字母，此时计算机显示祝贺信息，并给出玩家游戏分数，游戏计算公式为：

当前还剩猜测次数*目标串中不重复字母个数；

二是游戏结束，玩家失败，即玩家在耗尽了猜测次数仍然没有猜对目标单词的全部字母。

Problem 4:

实现一个 hangman 游戏的变体，计算机可以给出相关提示，具体游戏规则如下：

（1）：总体规则跟 problem1-3 中的 hangman 游戏一样，即计算机与玩家的交互，相关输入输出、以及猜测错误、猜测不符合要求的惩罚措施等是一样的

（2）：不同的是，当用户猜测字符 ‘*’ 时，计算机应输出所有 wordlist 中与当前已猜测结果相匹配的单词，而这些单词，通过方法 show_possible_matches(my_word)获取。

【解题思路】

首先在看完维基百科中 hangman 游戏的规则介绍后，脑子里大致想出了几个需要的函数，也画了一下相关草图，但还是头绪有点混乱，于是回到 ps2 的文档，一步一步跟着要求去实现了 hangman 的相关函数，发现给出的项目文档正是在一步一步引导着我们去解决整个问题，向我们灌输着模块化编程的思想，而自己所想的思路跟文档的引导思路还是有一定的差距。

因此思路就是跟着文档的步伐在走，一步步实现，微调完善的过程。

在实现每个函数之后，我都会单独对这个函数的功能进行测试，以确保其在被调用的时候能够正确发挥作用。相关测试的结果将在下面关键代码块中给出。

耗时最久的就是 hangman(secret_word) 函数的实现，因为需要运用到较多的选择结构（if,elif,else 结构），一开始跟着文档走，逐步添加玩家的猜测规则，虽然给出的案例测试结果都符合要求。但是当我测试用自己想的特殊案例时，一个问题出现了并引发出我的思考：

连续输入两次 $ ，$时（这种情况是文档中没有的，虽然应该没有玩家这么笨，但是还是符合输入规则的）第 2 次输入 $ 后，计算机提示“That is not a valid letter”并且警告次数减了 2，原因当然是因为：第 2 次输入的 $ 既满足重复输入，也满足不是大小写字母，因此警告次数减了 2。显然这种结果是不正确的，玩家有点冤，警告次数应该减 1 。

我发现我的设计模式是不合理的，于是我决定重新捋一下思路和设计框架：当有多方面的条件，意味着多方面的选择时，首先得确定什么条件才是“大前提”，然后是“次前提“小

前提”即条件选择的层次问题，最终给出了下面的框架（伪代码）

```
while 猜测次数没用完:
    输出相关提示话语
    输入 x
    if 玩家已经猜过这个字母
        if warning_left>0: 警告次数没用完
            warning_left-=1
        else:    警告次数为 0 了
            times_left-=1 #
    else:玩家尚未猜测过这个字母
        list_guessed.append(x) 先存储玩家猜测结果
        if 玩家输入不是字母:
            if warning_left>0:
                warning_left-=1
            else:
                times_left-=1
        if 玩家输入是字母
            if 玩家猜测字母在目标中：
                相关功能实现和输出提示
            else 玩家猜测字母不在目标中：
                if x 是元音字母
                    times_left-=2
                else:x 是辅音字母
                    times_left-=1
```

即由外到内判断顺序为：是否还有机会猜测——>猜测是否重复——>猜测字符是否是大小写字母——>猜测字母是否在目标种——>猜测字母是否为元音字母。

然后在正确的地方加上相关提示语句和惩罚措施语句。

Problem4 ： 先实现 match_with_gaps(my_word, other_word)， 函数，然后通过调用这个函数可以很容易实现函数 show_possible_matches(my_word)， 而 def hangman_with_hints(secret_word)与 之前的 hangman(secret_word)相比，就是需要在用户输入 * 号时输出所有 wordlist 中与已猜测结果匹配的单词。因此只需要在呀 u 俺来 hangman 的基础上，添加条件判断语句，如果条件成立，则调用函数

show_possible_matches(letters_guessed)函数输出所有符合条件的单词作为提示。

【关键代码块】

各个函数的实现代码及测试结果如下：因为文件 hangman 里面太多注释符号，看着比较乱，故有些函数实现好了并确保功能正确后才粘贴到 hangman 中，这些函数的实现过程是写在 testtemp.py 中，作为测试功能使用.

Problem1-3

```
#判断玩家猜测字符是否正确
def is_word_guessed(secret_word, letters_guessed):
    list_secret=list(secret_word)
    #只要目标中出现一个字母不在玩家猜测结果中，返回 False,都在的话，返回 True
    for i in list_secret:
        if i not in letters_guessed:
            return False
    return True
```

```
>>> from testtemp import is_word_guessed
>>> secret_word='apple'
>>> letters_guessed=['e','i','k','p','r','s']
>>> print(is_word_guessed(secret_word, letters_guessed))
False
>>> letters_guessed=['e','p','p','l','e']
>>> print(is_word_guessed(secret_word, letters_guessed))
False
>>> letters_guessed=['a','p','p','l','e']
>>> print(is_word_guessed(secret_word, letters_guessed))
True
```

```
#获取玩家已猜对字母
def get_guessed_word(secret_word, letters_guessed):
    length=len(secret_word)
    list_secret=['_ ']*length #列表元素先全部初始化为'_'
    for i in letters_guessed:
        for j in range(length):
            if i==secret_word[j]: #用猜对的字母替换掉对应位置的'_'
                list_secret[j]=secret_word[j]
    string="".join(map(lambda x:str(x),list_secret)) #列表转字符串
    return string
```

```
>>> import testtemp
>>> secret_word='apple'
>>> letters_guessed=['e','i','k','p','r','s']
>>> print(testtemp.get_guessed_word(secret_word,letters_guessed))
_ pp_ e
>>> letters_guessed=[]
>>> print(testtemp.get_guessed_word(secret_word,letters_guessed))
_ _ _ _ _
>>> letters_guessed=['p']
>>> secret_word='typedef'
>>> print(testtemp.get_guessed_word(secret_word,letters_guessed))
_ _ p_ _ _ _
>>>
```

```
#获取剩余可猜测字母范围
def get_available_letters(letters_guessed):
    #初始化可猜字母为全部小写字母
    letters_all="abcdefghijklmnopqrstuvwxyz"
    for i in letters_all:
        if i in letters_guessed: #如果玩家已经猜过 i 则将其替换为'_ '
            letters_all=letters_all.replace(i,'')
    return letters_all
```

```
>>> import testtemp
>>> letters_guessed=['e','i','k','p','r','s']
>>> print(testtemp.get_available_letters(letters_guessed))
abcdfghjlmnoqtuvwxyz
>>> letters_guessed=['e','b','c','d','a','b']
>>> print(testtemp.get_available_letters(letters_guessed))
fghijklmnopqrstuvwxyz
```

hangman 的猜测成功效果展示：secret_word='tact' 为了展示程序实现的功能，故意输入了一些非字母，不在目标串中的元音字母等情况

```
>>> import hangman
Loading word list from file...
   55900 words loaded.
>>> hangman.hangman('tact')
Welcome to the game hangman!
I'm thinking of a word that is 4 letters long!
You have 3 warnings left.
-------------
You have 6 guesses left.
Available letters: abcdefghijklmnopqrstuvwxyz
Please guess a letter:a
Good guess: _ a_ _
-------------
You have 6 guesses left.
Available letters: bcdefghijklmnopqrstuvwxyz
Please guess a letter:b
Oops! That letter is not in my word. _ a_ _
-------------
You have 5 guesses left.
Available letters: cdefghijklmnopqrstuvwxyz
Please guess a letter:s
Oops! That letter is not in my word. _ a_ _
-------------
You have 4 guesses left.
Available letters: cdefghijklmnopqrtuvwxyz
Please guess a letter:$
Oops!That is not a valid letter.You have 2 warnings left: _ a_ _
-------------
You have 4 guesses left.
Available letters: cdefghijklmnopqrtuvwxyz
Please guess a letter:6
Oops!That is not a valid letter.You have 1 warnings left: _ a_ _
-------------
You have 4 guesses left.
Available letters: cdefghijklmnopqrtuvwxyz
Please guess a letter:$
Oops!  You've already guessed that letter.You have 0 warnings left: _ a_ _

-------------
You have 4 guesses left.
Available letters: cdefghijklmnopqrtuvwxyz
Please guess a letter:u
Oops! That letter is not in my word. _ a_ _
-------------
You have 2 guesses left.
Available letters: cdefghijklmnopqrtvwxyz
Please guess a letter:c
Good guess: _ ac_
-------------
You have 2 guesses left.
Available letters: defghijklmnopqrtvwxyz
Please guess a letter:t
Good guess: tact
-------------
Congratulations,  you  won!
Your total score for this game is: 6
```

hangman 的猜测失败效果展示，secret_word='apple' 跟上面相比，主要是展示了上面那个例子中没有体现的，当警告次数为 0 后继续猜测重复或者不合规字母时，猜测次数减 1 的效果

```
>>> import hangman
Loading word list from file...
   55900 words loaded.
>>> secret_word='apple'
>>> hangman.hangman(secret_word)
Welcome to the game hangman!
I'm thinking of a word that is 5 letters long!
You have 3 warnings left.
-------------
You have 6 guesses left.
Available letters: abcdefghijklmnopqrstuvwxyz
Please guess a letter:i
Oops! That letter is not in my word. _ _ _ _ _
-------------
You have 4 guesses left.
Available letters: abcdefghjklmnopqrstuvwxyz
Please guess a letter:a
Good guess: a_ _ _ _
-------------
You have 4 guesses left.
Available letters: bcdefghjklmnopqrstuvwxyz
Please guess a letter:#
Oops!That is not a valid letter.You have 2 warnings left: a_ _ _ _
-------------
You have 4 guesses left.
Available letters: bcdefghjklmnopqrstuvwxyz
Please guess a letter:#
Oops!  You've already guessed that letter.You have 1 warnings left: a_ _ _ _
-------------
You have 4 guesses left.
Available letters: bcdefghjklmnopqrstuvwxyz
Please guess a letter:7
Oops!That is not a valid letter.You have 0 warnings left: a
-------------
You have 4 guesses left.
Available letters: bcdefghjklmnopqrstuvwxyz
Please guess a letter:7
Oops! You've already guessed that letter.You have no warnings left,so you lose one guess: a_ _ _ _
-------------
You have 3 guesses left.
Available letters: bcdefghjklmnopqrstuvwxyz
Please guess a letter:p
Good guess: app_ _
-------------
You have 3 guesses left.
Available letters: bcdefghjklmnoqrstuvwxyz
Please guess a letter:r
Oops! That letter is not in my word. app_ _
-------------
You have 2 guesses left.
Available letters: bcdefghjklmnoqstuvwxyz
Please guess a letter:o
Oops! That letter is not in my word. app_ _
-------------
Sorry, you ran out of guesses.The word was apple
```

由计算机随机抽取 word.text 中的一个单词作为目标串，这个比较难猜，即直接运行脚本文件

```
   55900 words loaded.
Welcome to the game hangman!
I'm thinking of a word that is 8 letters long!
You have 3 warnings left.
-------------
You have 6 guesses left.
Available letters: abcdefghijklmnopqrstuvwxyz
Please guess a letter:g
Oops! That letter is not in my word. _ _ _ _ _ _ _ _
-------------
You have 5 guesses left.
Available letters: abcdefhijklmnopqrstuvwxyz
Please guess a letter:h
Oops! That letter is not in my word. _ _ _ _ _ _ _ _
-------------
You have 4 guesses left.
Available letters: abcdefijklmnopqrstuvwxyz
Please guess a letter:i
Good guess: _ _ i_ _ _ _ _
-------------
You have 4 guesses left.
Available letters: abcdefjklmnopqrstuvwxyz
Please guess a letter:n
Good guess: _ _ in_ _ _ _
-------------
You have 4 guesses left.
Available letters: abcdefjklmopqrstuvwxyz
Please guess a letter:c
Oops! That letter is not in my word. _ _ in_ _ _ _
-------------
You have 3 guesses left.
Available letters: abdefjklmopqrstuvwxyz
Please guess a letter:l
Oops! That letter is not in my word. _ _ in_ _ _ _
-------------
You have 2 guesses left.
Available letters: abdefjkmopqrstuvwxyz
Please guess a letter:r
Oops! That letter is not in my word. _ _ in_ _ _ _
-------------
You have 1 guesses left.
Available letters: abdefjkmopqstuvwxyz
Please guess a letter:y
Oops! That letter is not in my word. _ _ in_ _ _ _
-------------
Sorry, you ran out of guesses.The word was stinkpot
```

Problem 4 各个函数的实现代码及测试结果如下：

```
def del_space(string): #将字符串去除空格后转化成列表
    lst=[]
    for i in string:
        if i!=' ':
            lst.append(i)
    return lst

#检验两个单词是否按规则匹配
def match_with_gaps(my_word, other_word):
    #先将字符串转换成列表，方便操作
    list_my_word=del_space(my_word)
    list_other_word=list(other_word)
    if len(list_my_word)!=len(list_other_word): #长度不一致
        return False
    else:
        length=len(list_my_word)
        for i in range(length): #对应位置均是字母且不相等
            if list_my_word[i]!='_' and list_my_word[i]!=list_other_word[i]:
                return False
        #list_my_word[i]=='_'时
        for i in range(length):
            j=i+1
            for j in range(length):
                if list_other_word[i]==list_other_word[j] and list_my_word[i]!=list_my_word[j]:
                    return False
    return True
```

```
>>> import testtemp
>>> print(testtemp.match_with_gaps("te_ t","tact"))
False
>>> print(testtemp.match_with_gaps("a_ _ le","banana"))
False
>>> print(testtemp.match_with_gaps("a_ _ le","apple"))
True
>>> print(testtemp.match_with_gaps("a_ ple","apple"))
False
>>> print(testtemp.match_with_gaps("a_ _ le","acple"))
True
>>>
```

```
#输出匹配呢的单词
def show_possible_matches(my_word):
    flag=0 #用于标记是否存在可能匹配的单词
    possible_word=[] #存储可能匹配的单词
    for i in wordlist:
        if match_with_gaps(my_word,i):
            flag=1
            possible_word.append(i)
    if flag==0:#不存在可能匹配的单词
        print("No matches found.")
    else:
        print("Possible word matches are:")
        for i in possible_word:
            print(i,end=' ')
    print("")
```

```
>>> show_possible_matches("t_ _ t")
Possible word matches are:
tact tart taut teat tent test text that tilt tint toot tort tout trot tuft twit
>>> show_possible_matches("abbbb_")
No matches found.

>>> show_possible_matches("a_ pl_ ")
Possible word matches are:
ample amply
>>>
```

```
>>> import hangman
Loading word list from file...
   55900 words loaded.
>>> hangman.hangman_with_hints("apple")
Welcome to the game hangman!
I'm thinking of a word that is 5 letters long!
You have 3 warnings left.
-------------
You have 6 guesses left.
Available letters: abcdefghijklmnopqrstuvwxyz
Please guess a letter:a
Good guess: a_ _ _ _
-------------
You have 6 guesses left.
Available letters: bcdefghijklmnopqrstuvwxyz
Please guess a letter:l
Good guess: a_ _ l_
-------------
You have 6 guesses left.
Available letters: bcdefghijkmnopqrstuvwxyz
Please guess a letter:*
Possible  word  matches  are:
addle adult agile aisle amble ample amply amyls angle ankle apple apply aptly arils atilt
-------------
You have 6 guesses left.
Available letters: bcdefghijkmnopqrstuvwxyz
Please guess a letter:e
Good guess: a_ _ le
-------------
You have 6 guesses left.
Available letters: bcdfghijkmnopqrstuvwxyz
Please guess a letter:o
Oops! That letter is not in my word. a_ _ le

-------------
You have 4 guesses left.
Available letters: bcdfghijkmnpqrstuvwxyz
Please guess a letter:u
Oops! That letter is not in my word. a_ _ le
-------------
You have 2 guesses left.
Available letters: bcdfghijkmnpqrstvwxyz
Please guess a letter:f
Oops! That letter is not in my word. a_ _ le
-------------
You have 1 guesses left.
Available letters: bcdghijkmnpqrstvwxyz
Please guess a letter:p
Good guess: apple
-------------
Congratulations,you won!
Your total score for this game is: 4
>>>
```

【项目源码】

大部分代码都已经展示在上面了，整个项目代码详情请见 hangman.py

【项目心得】

相比上两次实验，这次项目还是稍微费时间一点的。总体来说，难度不大，思路都已经给出来了，并且环环相扣，一步步引导。最重要的收获应该是这个项目的模块化编程思想，函数式思想。刚了解完游戏规则的时候，自己大概构思了一下怎么做，但是还是比较一头雾水。文件中化整为零，由总体到部分，再从部分回归整体的思路，从中受益匪浅。复杂的项目的实现往往就是一连串功能函数的灵活调用，而实现一个功能单一的函数比实现一整个项目自然是容易的多。