实验报告

一、 实验目的

编写程序实现交互小游戏hangman。

二、 实验过程

注：为适应排版，此处代码与源代码有所出入

1. 编写程序提纲

- 辅助函数

1.

```
#返回猜测的字母能否组成电脑选定的单词
def is_word_guessed(secret_word, letters_guessed):
```

2.

```
#根据猜测的字母组合出部分完成的单词
def get_guessed_word(secret_word, letters_guessed):
```

3.

```
#给出目前还没有猜过的单词
def get_available_letters(letters_guessed):
```

- 主要函数

```
#根据电脑选定的单词进行游戏
def hangman(secret_word):
```

2. 具体函数实现

1. is_word_guessed

```
def is_word_guessed(secret_word, letters_guessed):
    #如果在secret_word中有字母没有出现在letters_guessed中，则该单词还没有被猜中
    for i in range(len(secret_word)):
```

```
        if not secret_word[i] in letters_guessed:
            return False
    return True
```

2. get_guessed_word

```
def get_guessed_word(secret_word, letters_guessed):
    temp=''#temp初始为空
    for i in range(len(secret_word)):
      if secret_word[i] in letters_guessed:
        #当该字母出现在letters_guessed中时，将其添加到temp末尾
        temp=temp+secret_word[i]
      else:
        #当该字母没有出现在letters_guessed中时，在temo末尾添加 “_ ”
        temp=temp+'_ '
    return temp
```

3. get_available_letters

```
def get_available_letters(letters_guessed):
    #全集U为26个小写字母组成的集合，集合A为letters_guessed中的元素组成的集合
    #现求A的补集
    res=''
    alpha=string.ascii_lowercase
    for letter in alpha:
      if not letter in letters_guessed:
        res=res+letter
    return res
```

4. hangman

```
def hangman(secret_word):
    print('\nI am thinking of a word that is ',end='')
    print('{} letters long'.format(len(secret_word)))
    print('You have 3 warnings left.\n')
    print('-------------')
    guesses_left=6#剩余的猜的次数
    warnings_left=3#剩余的警告数
    letters_guessed=[]#当前已经猜过的字母
    output=''#当前单词的完成情况
    right_letters=[]#猜中的字母

    while guesses_left>0:#当猜的次数用完是结束循环

        #输出提示信息
        print('You have {} guesses left.'.format(guesses_left))
        print('Available letters:',get_available_letters(letters_guessed))
```

```
        letter=input('Please guess a letter:')#获取用户输入

        guessed_before=False
        if letter.lower() in letters_guessed:
            guessed_before=True
        else:
            guessed_before=False

        #如果输入只有一个字母且之前没有猜这个字母
        if len(letter)==1 and letter.isalpha() and not guessed_before:
            letter=letter.lower()#将其转为小写字母
            letters_guessed.append(letter)#将其加入到猜过的字母集合中

            if letter in secret_word:#如果secret_word中有该字母

                #output为当前单词的完成情况
                output=get_guessed_word(secret_word,letters_guessed)

                right_letters.append(letter)#将该字母加入到猜中的字母集合中
                print('Good guess: ',output)

                if not '_' in output:#secret_word中所有字母均被猜出
                print('\n-----------\n')
                break

            else:#secret_word中没有该字母
                guesses_left-=1#可以猜的次数减一
                if letter in ['a','e','i','o','u']:#如果是元音字母，再减一
                guesses_left-=1
                print('Oops! That letter is not in my word: ',output)

        else:
            if letter in letters_guessed:#该字母之前已经猜过
                print("Oops! You've already guessed that letter.",end='')
            else:#不合法的字符
                print('Oops! That is not a valid letter.',end='')

            if warnings_left>0:#警告次数还有剩余时，扣一次警告次数
                warnings_left-=1
                print('You have '+str(warnings_left),end='')
                print(+'  warning(s) left: ',output)

            else:#警告次数没有剩余时，直接扣一次猜的次数
                guesses_left-=1
                print('You have no warnings left, so you lose one guess: ',output)

        print('\n-----------\n')

    if guesses_left>0:#如果可以猜的次数尚有剩余
        print('Congratulations, you won!')

        #输出得分
        score=guesses_left*len(right_letters)
```

```
            print('Your total score for this game is: ',score)
        else:
            #输出失败信息
            print('Sorry, you ran out of guesses.',end='')
            print(' The word was {}.'.format(secret_word))
```

3. 增加提示功能

1. match_with_gaps(my_word, other_word)验证当前部分完成的单词能否与完整单词匹配

```
def match_with_gaps(my_word, other_word):
    i=0#my_word字符串中的下标
    j=0#other_word中的下标
    tmp=[]#存放my_word不应该存在的字母
    while i<len(my_word) and j<len(other_word):
        if my_word[i]=='_':#如果i指向的字符为'_'
            i+=2#则i需要加2，因为'_'后面还有一个空格
            tmp.append(other_word[j])#将j指向的字母加入到tmp中
            j+=1#j加1
            continue

        #如果i和j指向的字母不等，则直接返回False
        elif not my_word[i]==other_word[j]:
            return False
        i+=1
        j+=1
    for letter in tmp:
        if letter in my_word:#如果my_word中出现了不该出现的字母，则返回False
            return False

    #i和j没有同时走到字符串末尾，返回False
    if not i==len(my_word) or not j==len(other_word):
        return False
    return True
    #所谓不该出现的字母是指在my_word中出现了
    #但是该字母并没有出现在my_word中所有该出现的地方。
    #例如match_with_gaps('ap_ le','apple')
    #在my_word中，p既然出现了，根据游戏规则，两个p应该同时出现
    #而不是只出现一个。在这种情况下，p就是my_word中不应该出现的字母
```

2. show_possible_matches(my_word):展示所有可能的结果

```
#如果单词中所有字母均被找到
#则直接输出该单词并结束函数
if not '_ ' in my_word:
        print(my_word)
        return
```

```
    #统计符合条件的单词的个数
    cnt=0
    # for i in range(len(my_word)):
    #   if my_word[i]==' ':
    #     cnt+=1
    #pattern='^'+my_word.replace('_ ','.')+'$'
    for word in wordlist:#一一匹配单词表中的单词
      #if not re.match(pattern,word)==None:
        if match_with_gaps(my_word,word): #如果某个单词与当前部分完成的单词匹配成功
            print(word,end=' ')#输出该单词
            cnt+=1
    if cnt==0:#没找到合适的单词
        print('No matches found')
    print('')
```

3. hangman_with_hints(secret_word)

   该函数与之前的hangman(secret_word)大致相同，在此不再赘述。

三、实验思考

1. 在函数show_possible_matches中，需要把当前部分完成的单词与单词表中所有单词一一比较，效率较低。因为单词表中的单词是按序排列的，在单词首字母确定的情况下，可以尝试二分法定位到以首字母开头的单词，从而缩小范围，提高效率。
2. 在函数match_with_gaps中，可以尝试引入正则表达式来进行匹配。