实验报告

一、 实验目的

编写程序实现小游戏 6.0001 word game

二、 实验过程

注：为适应排版，此处代码与源代码有所出入

1. 确定编程思路

根据作业要求，按照模块化的思路编写程序

2. 编写程序提纲

- 辅助函数

1. 计算一个单词的得分

```
def get_word_score(word, n):
```

2. 返回一轮游戏的字母集合

```
def deal_hand(n):
```

3. 根据用户输入的单词更新字母集合

```
def update_hand(hand, word):
```

4. 检验用户输入的单词是否合法

```
def is_valid_word(word, hand, word_list):
```

5. 计算hand中的字母数

```
def calculate_handlen(hand):
```

6. 替换hand中的一个指定字母

```
def substitute_hand(hand, letter):
```

- 主要函数

1. 进行一轮游戏

```
def play_hand(hand, word_list):
```

2. 控制进行多轮游戏

```
def play_game(word_list):
```

3. 具体函数实现

- def get_word_score(word, n)

```
def get_word_score(word, n):
    word=word.lower()#将word所有字母转换为小写
    length=len(word)
    first=0
    for letter in word:
        first+=SCRABBLE_LETTER_VALUES[letter]#乘积的第一部分
    second=1
    second=max(second,7*length-3*(n-length))#乘积的第二部分
    return first*second
```

- def deal_hand(n):

```
def deal_hand(n):
    hand={}
    num_vowels = int(math.ceil(n / 3))#元音字母占1/3，其中包括一个通配符
    hand['*']=1#通配符

    for i in range(1,num_vowels):#元音字母
        x = random.choice(VOWELS)
        hand[x] = hand.get(x, 0) + 1

    for i in range(num_vowels, n):#其他字母
        x = random.choice(CONSONANTS)
        hand[x] = hand.get(x, 0) + 1

    return hand
```

- def update_hand(hand, word):

```
def update_hand(hand, word):
    hand_copy=hand.copy()
    word=word.lower()#将word所有字母转换为小写

    for letter in word:
        temp=hand_copy.get(letter,0)
        if temp>=1:#如果字典中有该键，且其值大于等于1
            hand_copy[letter]-=1#将其值减1
            if hand_copy[letter]==0:#若减1后值为0，将其从字典中去除
                hand_copy.pop(letter,0)

    return hand_copy
```

- def is_valid_word(word, hand, word_list):

```
def is_valid_word(word, hand, word_list):
    hand_copy=hand.copy()
    word=word.lower()#将word所有字母转换为小写
    for letter in word:
        temp=hand_copy.get(letter,0)
        if temp==0:#如果word中存在字典中没有的字母，直接返回False
            return False
        hand_copy[letter]-=1#如果这个字母在字典中存在，将其对应的值减1

    word=word.replace("*","[aeoiu]")#用[aeiou]替换*以进行正则表达式的匹配
    pattern=re.compile(word)#编译正则表达式

    for w in word_list:#对word_list中的每一个单词进行正则表达式的匹配
        if pattern.fullmatch(w):#如果可以完全匹配，则返回True
            return True
    return False#在word_list中没有找到与之匹配的单词，返回False
```

- def calculate_handlen(hand):

```
def calculate_handlen(hand):
    sum=0
    for k,v in hand.items():#遍历键值对，对值求和
        sum+=v
    return sum
```

- def substitute_hand(hand, letter):

```
def substitute_hand(hand, letter):
hand_copy=hand.copy()#对hand的副本进行操作
```

```
    if 0==hand_copy.get(letter,0):#如果hand中没有字母，直接返回即可
        return hand_copy

    alphabet=string.ascii_lowercase#26个小写字母
    # alphabet='a'

    unused=''#保存hand中用过的字母
    for alpha in alphabet:
        if hand_copy.get(alpha,0)==0:#如果字母在hand中出现过，将其加入到unused中
            unused+=alpha

    char=random.choice(unused)#从未用过的字母中随机挑选一个
    hand_copy[char]=1#加入到hand_copy中
    hand_copy[letter]-=1#将原来的字母对应的值减1
    if hand_copy[letter]==0:#如果原来的字母对应的值降为0，将其从hand_copy中去除
        hand_copy.pop(letter,0)

    return hand_copy
```

- def play_hand(hand, word_list):

```
def play_hand(hand, word_list):
    total_score=0
    #hand=deal_hand(HAND_SIZE)
    while calculate_handlen(hand)>0:#当还有可用字母时继续循环
        print("Current hand: ",end='')
        display_hand(hand)
        #提示用户输入答案
        ans=input('Enter word, or "!!" to indicate that you are finished: ')

        if '!!'==ans:
            break

        if is_valid_word(ans,hand,word_list):#如果答案合法
            score=get_word_score(ans,calculate_handlen(hand))#计算得分
            total_score+=score#更新总分
            print('"{}" earned {} points. '.format(ans,score),end='')
            print('Total: {} points\n'.format(total_score))

        else:
            print("That is not a valid word. Please choose another word.\n")

        hand=update_hand(hand,ans)#无论用户答案合法与否，均扣除用户输入的字母

    if calculate_handlen(hand)==0:
        print("\nRan out of letters")
    #print("Total score: {} points".format(total_score))
    return total_score
```

- def play_game(word_list):

```
def play_game(word_list):
hand_number=0#可以进行的游戏轮数
    substitute_chance=1#可替换一个字母的剩余次数
    replay_chance=1#可以重玩一轮的剩余次数
    replay=False#表示该轮是否是重玩的
    scores=[]#表示每一轮的最高得分
    alphabet=string.ascii_lowercase
    while True:
        inp=input("Enter total number of hands: ")#提示用户输入游戏轮数
        try:#简单的错误处理
            hand_number=int(inp)
            break
        except:
            print("Bad input, please try again!")

    hands=[]#每一轮游戏的字母集
    for i in range(hand_number):#随机获取每一轮游戏的字母集
        hands.append(deal_hand(HAND_SIZE))

    # hands.append({'a':1,'c':1,'i':1,'*':1,'p':1,'r':1,'t':1})
    # hands.append({'d':2,'*':1,'a':1,'o':1,'u':1,'t':1})

    i=0
    while i<hand_number:
        hand=hands[i]
        if not replay:
            print("Current hand: ",end='')
            display_hand(hand)
        op=''

        if substitute_chance>0 and not replay:#还有替换机会并且不是在重玩某一轮
            while True:#处理用户输入错误
                op=input("Would you like to substitute a letter?")
                if op=='no' or op=='n':
                    break
                elif op=='yes' or op=='y':
                    substitute_chance-=1#减去一次替换机会
                    letter=''#用户想要替换的字母

                    while True:#处理用户输入错误
                        #提示用户输入想要替换的字母
                        letter=input("Which letter would you like to replace: ")
                        if len(letter)==1 and (letter=='*' or letter in alphabet:
                            break
                        else:
                            print("Bad input, please try again!")

                                        #更新该轮游戏的字母集
                    hand=substitute_hand(hand,letter)
                    hands[i]=hand
                    break
                else:
                    print("Bad input, please try again!")
```

```
        print()
        score=play_hand(hand,word_list)#展示该轮游戏得分

        #print('\nRan out of letters')
        print('Total score for this hand: %d'%score)
        print('----------')

        if len(scores)==i:
            scores.append(score)
        elif i<len(scores):#选取最高分
            scores[i]=max(score,scores[i])

        if replay_chance>0:#用户还有一次重玩机会时
            while True:#处理用户输入错误
                op=input("Would you like to replay the hand?")
                if op=='yes' or op=='y':
                    i-=1
                    replay_chance-=1#重玩机会减1
                    replay=True#表示下一轮游戏是重玩的
                    break
                elif op=='no' or op=='n':
                    replay=False#表示下一轮游戏不是重玩的
                    break
                else:
                    print("Bad input, please try again!")
        else:#表示下一轮游戏不是重玩的
            replay=False

        i+=1
    total_score=0
    for score in scores:#计算总分
        total_score+=score
    print("Total score over all hands: %d"%total_score)
```

三、 实验思考

- 在函数is_valid_word(word, hand, word_list)中，最后检查用户输入的单词是否在word_list中时，由于通配符（可以是a、e、i、o、u中的任一个）的存在，因此可能需要将每个元音字母代入到单词中，再检查这个单词是否在word_list中，这样子一共需要检查五次，效率较低。为解决这个问题，可以使用正则表达式，用[aeiou]替换‘*’，再与word_list中的每一个单词进行匹配。如用户输入的单词为‘ic *’，第一种方式需要分别检查‘ica’、‘ice’、‘ici’，‘ico’，‘icu’，而第二种方式替换为‘ic[aeiou]’后，使用re.fullmatch方法进行匹配，只需要与word_list中的每个单词进行一次匹配即可，从而提高了效率。
- 当对字典类型使用[] 操作符时，字典中可能没有该键，从而会引起一个KeyError。这个问题可以使用dict.get(key[，default]) 方法解决，如果字典中没有该键时，该方法会返回参数中的default值（default值默认为None），从而永远不会引起KeyError。