Python Practice Set 3

Words Game

概述（Introduction）

本练习包含了需求分析、代码架构、代码编写、测试等“编程”的全过程。

本练习使用 list 和 dict 实现一个单词游戏 hand！不要被文档的长度吓倒：我们尽可能把问题、要求表达清楚，请仔细浏览阅读、理解。先简单介绍拼词(hand)游戏：一组随机字母（即 hand）给 player， player 用这些字母构造一个或多个单词。程序根据规则计算拼词游戏得分 (points)。

先不要一开始就编码；请先理解问题、设计出解决方法，再进行编码！ 也许你知道有比 PS3 给出的算法更简单的方法，但本课程仍然要求你按照给出的程序架构完成你的设计。

你编写的代码应尽可能符合 Python 规范，如必要的注释、docstrings 等。*可以用 AI 帮助找出代码中的问题，但不能让 AI 生成代码（注意，你签署过 Honor Rules）。*

如同 PS2，人文社科专业的同学可以组队（2 人，都是人文社科专业）完成 PS3。

规则（Dealing）

（1）hand 是字典结构、随机抽取的字母表，长度 HAND_SIZE（常数）是指 hand 中的字母数量。

（2）player 将 hand 中的字母拼成尽可能长的合法单词，hand 中的每个字母只能用一次。如果 hand 中有重复字母，可重复使用，但只能使用到其重复的数量。拼了一个单词后，如果还有剩下的字母可继续拼词，直到所有字母用完。请参见下面的例子。

得分（Scoring）

（1）（所拼）**单词***（以下默认为单词，请关联上下文）*的分数由两个部分组成：

第一：单词中字母的分数之和。

第二：[7 * word_length - 3 * (n-word_length)] 或者 1，取两者较大值，其中：word_length 是单词的字母数， n 是 hand 中的字母总数。n-word_length 是未用的字母数。

（2）字母分依据字母分值表 SCRABBLE_LETTER_VALUES，每个小写字母分值如下：

Letters (a, e, i, l, n, o, r, s, t, u) are 1 point.
Letters (d, g) are 2 points.
Letters (b, c, m, p) are 3 points.
Letters (f, h, v, w, y) are 4 points.
Letter (k) is 5 points.
Letters (j, x) are 8 points.
Letters (q, z) are 10 points.

（3）举例

如 n=6，hand 有 1 个'w'、2 个'e'、 1 个"d"、1 个’t’、一个’i’，则组合成"weed"，其得分为 176：

(4+1+1+2）* (7*4 – 3*(6-4)) = 176。

上述第一个括号项是单词所用每个字母的分值之和; 第二个括号项是奖励较长单词、扣减遗留字母数的特殊计算，如果计算结果最低分 1 分（小于 1 分的，则得 1 分）。

余下的组合"it"一词，计算第一项得到字母分值 2。第二项 2*2 - 3*（2 -0）= 4，合计为 5。将两次拼词的得分求和就是游戏的得分。

开始（Getting Started）

1. 下载并保存 ps3.zip，其中包括 ps3.py，提供了一组初始过程、部分函数代码和模板，以及需要你编写的函数代码。

ps3.zip 还包括用于测试代码的文件 test_ps3.py，以及一个合法单词 words.txt 的文件。

除非特别情况，否则请不要更改或删除所下载文件中的任何内容。

2. 运行 ps3.py，请不要对其进行任何修改，以确保一切设置正确。该代码从 words.txt 中加载有效单词的列表，然后调用 play_game 函数。在稍有延迟后，应该看到以下输出显示：

```
Loading word list from file...
        83667 words loaded.
        play_game not yet implemented.
```

如果输出 IOError（错误原因可能是没有相关文件或目录），请确保 words.txt 与 ps3.py 在相同的目录中！

3. 文件 ps3.py 有多个已经完成的函数，供参考、使用。你也可以重写它们的代码。

4. PS3 的“结构化”：编写多个函数，组合在一起形成完整的游戏代码。

请不要等到整个代码准备好再运行，而是分阶段设计并先测试一个函数后，再继续编写下一个函数。此方法称为软件的单元测试，它对初学者设计和测试代码很有用。

5. PS3 有 docstrings 或伪代码供编码时参考、使用。提交的作业文件中，可保留它们。

6. 文档接下来给出了设计(编码)要求（**问题 1 到问题 4**）。

可运行 test_ps3.py 来检查到目前为止的工作，这里测试函数用来帮助衡量所被测试的函数是否达到设计要求。测试函数的程序设计/测试常用的方法。

如果代码通过单元测试，将显示" SUCCESS"，否则，显示 FAILURE。这些测试并非详尽无遗，可能还需要其他方式测试代码（例如，使用不同的测试值）。*换言之，测试程序只能测试程序错误，但不能保证程序没有错误。*我们将对你提交的代码进行测试，也不一定是用这里给出的测试函数和数据。

如果对最初的 ps3.py 运行 test_ps3.py，所有测试都会是 FAILURE。

测试函数如下：

test_get_word_score	**测试get_word_score的实现**
test_update_hand	**测试update_hand的实现**
test_is_valid_word	**测试is_valid_word的实现**
test_wildcard	**测试为支持通配符（wildcards）所做的修改（稍后有进一步的介绍）**

问题 1：单词分数（Word scores）

首先通过get_word_score函数，计算单个单词的分数。请在ps3.py中完成它的代码。前述“得分（Scoring）”中已经给出了计算得分的规则。

使用ps3.py顶部定义的SCRABBLE_LETTER_VALUES字典。

注意，源代码中HAND_SIZE设为7，别以为总是有7个字母！我们已经知道了str.lower函数的作用，例如：

```
s = "My String"
print(s.lower())     ## 转换为小写字母: my string
```

测试：如果计算单词分数的函数正确，运行 test_ps3.py则test_get_word_score（）测试将通过。也应测试get_word_score函数，再使用一些合理的英语单词进行测试。请注意，通配

符 ("*") 测试将由于 KeyError 而崩溃。不过，这没事--此问题将在“问题4”中解决。

问题 2：组词处理（Dealing with hands）

在开始编写解决方案之前，请认真阅读如下所述，如下大多数函数已经提供代码了。

1. 什么是组词（Representing hands）

组词(Hand)是游戏 player 持有的一组随机字母。例如，player 可以从以下 hand 开始：a，q，l，m，u，i，l。hand 被表示为字典：键(keys)是小写字母，值(value)是该字母出现的次数。这里：

```
hand = {'a':1, 'q':1, 'l':2, 'm':1, 'u':1, 'i':1}
```

请注意 hand 中重复字母"l"是如何表示的。

hand 字典访问其值（values）的方法是 hand[key]。但仅当 key 在字典中才有效，否则会出错。避免出错可改用字典方法 dict.get()。如 hand.get('a',0)，'a'在字典中则返回'a'的值,如果'a'不在字典 hand 里，则返回"None"，dict.get 方法不会引发 KeyError。

2. 将单词转换为字典表示形式

已定义了函数 get_frequency_dict(string)，位于 ps3.py 顶部。string 作为输入参数，它返回一个字典：键是字符串中的字母，值是该字母在输入字符串中出现的次数。例如：

```
get_frequency_dict("hello")
{'h': 1, 'e': 1, 'l': 2, 'o': 1}
```

如此，就是字典 hand 的形式。

3. 显示 hand

display_hand 函数将给定的字典 hand，以用户友好的方式显示它。例如：

```
hand = {'a':1, 'q':1, 'l':2, 'm':1, 'u':1, 'i':1}
>>> display_hand(hand)
```

a、q、l、m、u、i、l

请仔细阅读此函数，了解它的作用及其工作原理。

4. 产生一个随机 hand（Generating a random hand）

deal_hand 函数生成随机字典 hand：正整数 n 为输入参数，返回 n 个小写字母的新 hand 字典。同样，请仔细阅读此函数，并了解它的作用及其工作原理。

5. 从 hand 中取出字母（请你编写代码实现之）

Player 只能使用给定的 hand 中的字母（input 输入）拼出一个单词后，要将用过的字母从 hand 中去除，剩下字母组成了一个新的 hand。例如原 hand 中有如下字母：

a、q、l、m、u、i、l

player 输入(拼出)单词"quail"，留下的字母是 l，m。

（1）**编写函数 update_hand(hand,word)**

以 hand 和一个单词 word 作为输入，使用该 hand 的字母拼写单词后，返回只包含剩余字母的 new_hand。该函数不应修改原来的 hand。例如（在 IDLE 中的示例）：

```
>>> hand = {'a':1, 'q':1, 'l':2, 'm':1, 'u':1, 'i':1}
    ## 函数update_hand，参数为hand和一个单词
>>> new_hand = update_hand(hand, 'quail')   ## 剩余字母的new_hand
```

```
>>> new_hand
{'l': 1, 'm': 1}
>>> display_hand(new_hand)
l m
>>> display_hand(hand)        ## hand字典保持不变
a q l l m u i
```

提示：调用 update_hand 后，new_hand 可为 {'a':0,"q":0,"l":1,'m':1,"u":0,"i":0} 或 {"l":1, "m":1}，这取决于你的设计和实现。无论如何 display_hand()的前后输出都应该是相同，即不要修改 hand 字典。

（2）猜错的惩罚

如果 update_hand 中输入的是一个无效的单词（不在 words.txt 中），要么它不是一个真正的单词，要么是使用了不在 hand 中的字母。

作为猜错的惩罚，将失去这个单词在 hand 中的字母。**请实现此功能**。例如：

```
>>> hand = {'j':2, 'o':1, 'l':1, 'w':1, 'n':2}  #hand 字典
>>> hand = update_hand(hand, 'jolly')  # 单词 jolly 错了，hand 中没有 y
>>> hand
    {'j':1, w':1, 'n':2} ## 原 hand 中 joll 被移走(remove),剩下:j w n n
```

请根据代码中的规范（docstrings）实现 update_hand 函数。

HINT: You may wish to review ".copy" method of Python dictionaries.

测试：确保test_update_hand测试通过。你可以设计其他合理的输入测试update_hand。

问题 3： 有效单词（Valid words）

有效单词是指在 words.txt 中(忽略大小写)完全由 hand 中的现有字母组成。

请编写is_valid_word 函数的代码

Hint: 请阅读位于 ps3 头部的函数 load_words()，它已经把 words.txt 文件的合法单词放入 wordlist 的列表中了。你可以通过“Pythonic”方法检查 player 所拼单词的有效性。

测试：请使用 test_is_valid_wordtests 测试 is_vaild_word 函数。

问题 4 通配符

允许 hand 包含通配符字母(*)。 通配符只能替换元音。hand 最初只能包含一个通配符作为其字母之一。使用通配符（不同于所有其他字母）不得分。

在游戏中， player 可输入"*"（无引号）代替一个元音字母。此时，验证有“*”的单词的有效性，应验证通配符所在位置被任意一个元音替换后，单词是否有效。

如下的例子（请仔细阅读、理解），显示了通配符在 hand 的上下文中应该如何表现，它将问题 5 中实现。不要担心----只是提醒如何处理通配符。

Example #1: A valid word made without the wildcard

```
Current Hand:  c o w s * z
Enter word, or "!!" to indicate that you are finished:
cows "cows" earned 198 points. Total: 198 points

Current Hand:  * z
```

```
Enter word, or "!!" to indicate that you are  finished: !!
Total score: 198 points
```

Example #2: A valid word made using the wildcard

```
Current Hand:  c o w s * z
Enter word, or "!!" to indicate that you are finished: c*ws
"c*ws" earned 176 points. Total: 176 points

Current Hand:  o z
Enter word, or "!!" to indicate that you are finished: !!
Total score: 176 points
```

Example #3: An invalid word with a wildcard

```
Current Hand:  c o w s * z
Enter word, or "!!" to indicate that you are finished: c*wz
That is not a valid word. Please choose another word.

Current Hand: o s
Enter word, or "!!" to indicate that you are finished: !!
Total score: 0 points
```

Example #4: Another invalid word with a wildcard

```
Current Hand:  c o w s * z
Enter word, or "!!" to indicate that you are finished: *ows
That is not a valid word. Please choose another word.

Current Hand: c z
Enter word, or "!!" to indicate that you are finished: !!
Total score: 0 points
```

(1) 修改函数 deal_hand 以支持始终在 hand 中提供一个通配符。

请注意，deal_hand 已有代码是确保三分之一的字母是元音（代码头部定义的常量 VOWELS），其余字母是辅音。应保持辅音字母数不变，将其中一个元音替换为通配符。*注意，还需要修改文件顶部定义的一个或多个常量，以考虑通配符。*

(2) 修改 is_valid_word 函数支持通配符。

Hint：检查各个元音置换通配符位置，可能形成的单词是否有效；或查看字符串模块 find()函数帮助文档，看看它找不到字符时有什么返回。

测试：确保 test_wildcard 测试通过。可能还其他一些合理的输入来测试。

问题 5：玩拼词（Playing a hand）

现在，你可以准备好开始编写与游戏者交互的代码。

请编写 play_hand 函数的代码：允许用户对 hand 拼词游戏，当然可以先编写帮助函数 calculate_handlen（或五行代码就可以完成，提示用户如何玩这个游戏）。

要提前结束 hand 游戏，player 键入"!! "（两个感叹号，没有引号）。

请注意，行号后的 #BEGIN PSEUDOCODE 有一堆，当然，你应该知道是伪代码！这是为了帮助你编写函数代码。

测试：就像玩游戏一样：运行程序，调用 play_hand 函数（使用 hand 和 word_list）

注意：程序输出应与以下示例匹配。不应打印无关的"None"。

Example #1

```
Current Hand: a j e f * r x
Enter word, or "!!" to indicate that you are finished: jar
"jar" earned 90 points. Total: 90 points

Current Hand: * f x e
Enter word, or "!!" to indicate that you are finished: f*x
"f*x" earned 216 points. Total: 306 points

Current Hand: e
Enter word, or "!!" to indicate that you are finished: !!
Total score: 306 points
```

Example #2

```
Current Hand: a c f i * t x
Enter word, or "!!" to indicate that you are finished: fix
"fix" earned 117 points. Total: 117 points

Current Hand: a c t *
Enter word, or "!!" to indicate that you are finished: ac
That is not a valid word. Please choose another word.

Current Hand: t *
Enter word, or "!!" to indicate that you are finished: *t
"*t" earned 14 points. Total: 131 points

Ran out of letters. Total score: 131 points
```

问题 6：游戏（Playing a game）

游戏可以使用不同类型的 hands。

设计并实现 substitute_hand 和 play_game 函数。（参考 ps3 相应函数的 docstrings）。

解释：HAND_SIZE 常量确定 Hand 中的字母数。如前所述，hand 不是固定 7 个字母！设计要求是保持代码模块化 --改变 HAND_SIZE 就可以使得 hand 中的字母数量概念！

函数 **substitute_hand** 实现替换时，可以创建不包含 hand 中字母的其他字母的字符串 xString，从 xString 中随机选择一个字母，其 value 是被替换字母的 value，可查看 deal_hand 的代码，以了解如何从一组元素（如字符串）中随机选择元素。

测试：像玩游戏一样。使用 HAND_SIZE 的不同值，并确保仅修改 HAND_SIZE，就可以玩不同 hand 大小的单词游戏，可使用如下测试。

Example

```
Enter total number of hands: 2 Current hand: a c i * p r t
Would you like to substitute a letter? no

Current hand: a c i * p r t
Please enter a word or '!!' to indicate you are done: part
"part" earned 114 points. Total: 114 points

Current hand: c i *
Please enter a word or '!!' to indicate you are done: ic*
"ic*" earned 84 points. Total: 198 points

Ran out of letters
Total score for this hand: 198
----------
Would you like to replay the hand? no
Current hand: d d * l o u t

Would you like to substitute a letter? yes
Which letter would you like to replace: l

Current hand: d d * a o u t
Please enter a word or '!!' to indicate you are done: out
"out" earned 27 points. Total: 27 points

Current hand: d d * a
Please enter a word or '!!' to indicate you are done: !!
Total score for this hand: 27
----------
Would you like to replay the hand? yes Current hand: d d * a o u t
Please enter a word or '!!' to indicate you are done: d*d
"d*d" earned 36 points. Total: 36 points

Current hand: a o u t
Please enter a word or '!!' to indicate you are done: out

"out" earned 54 points. Total: 90 points


Current hand: a
Please enter a word or '!!' to indicate you are done: !!
Total score for this hand: 90
----------
Total score over all hands: 288
```

到此：本次作业就完成了！