Python Practice in lecture 6

Words Game

概述（Introduction）

在本练习中，您将实现一个单词游戏 hand！不要被这个作业文档的长度吓倒。请多阅读并理解。这里简单地描述单词游戏：这个游戏很像拼字游戏。字母交给玩家，然后玩家用字母构造一个或多个单词。每个有效单词都根据单词的长度和该单词中的字母数量获得分数(points 点数)。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！

规则（Dealing）

（1）玩家随机抽取的字母表（长度 HAND_SIZE）中字母，其可能包括特定字母的多个实例。HAND_SIZE 是一个常数，是指 hand 的字母数量。

（2）玩家将其排列成尽可能多的单词，但最多使用每个字母一次，如果字母表（hand 中与多个字母，可以重复使用，但只能使用到其重复的数量，请参见下面的例子）。

（3）有些字母可能未使用，但所组单词的长度（Size）会影响其最后的得分。

得分（Scoring）

（1）得分是每个单词形成的分数之和。

（2）单词的分数由两个部分组成：

第一：单词中字母数量的分数

第二：[7 * word_length - 3 * (n-word_length)] 或者 1，取两者较大值，其中：

word_length 是单词中使用的字母数， n 是当前获取的可用字母数

（3）评分，如 A 分值为 1，B 为 3，C 是 3，D 是 2，E 是 1，等等。有字典 SCRABBLE_LETTER_VALUES，将每个小写字母映射到拼字字母值。

（4）举例

如果 n=6，拼词包括 1 个'w'、2 个'e'和 1 个"d"（以及另外两个字母），则组合成"weed"一词得到的值为 176：(4+1+1+2）*(7*4 – 3*(6-4))= 176。第一项是所用每个字母的值之和;第二想是奖励较长单词的特殊计算，并扣减任何遗留下来的字母分值。

再如：如果 n=7，组合的"it"一词，计算第一项得到字母分值 2。第二项 1，因为 7*2 - 3*（7 - 2） = -1，小于 1，所以取 1.

开始（Getting Started）

1. 下载并保存 ps3.zip。这包括 python 文件 ps3.py，该文件提供了一组初始过程和模板。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。该代码从文件中加载有效单词的列表，然后调用 play_game 函数，你需要设计游戏所需的函数。如果一切正常，在一个小延迟之后，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(导致错误的原因可能是没有相关文件或目录),请确保将 words.txt 保存在与 ps3.py 相同的目录中!

3. 文件 ps3.py 有多个已经完成的函数，供你在编写解决方案时使用。可忽略注释之间的代码，但建议认真阅读并理解其内容（请回忆本课程的思想：功能（函数）化、分解和抽象的概念）。

4. 这个练习希望运用“结构化”：编写多个模块化的函数，然后将它们组合在一起以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. 这里有一些提示，将有助于编写必需的函数。当然，最后提交的文件中，可以但也不必删除它们。

在阅读和理解实验要求时，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。

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）。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。

如果使用最初提供的 ps3.py 运行 test_ps3.py，所有测试都会是 FAILURE。

这些是提供的测试函数如下：

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）

首先通过一个函数，计算单个单词的分数。请在ps3.py完成get_word_score代码。前述“得分（Scoring）”中已经给出了评分单词的规则。

使用ps3.py顶部定义的SCRABBLE_LETTER_VALUES字典。别以为总是有7个字母！参数 n 是输入单词时hand中的字母总数。我们已经知道了str.lower函数的作用，这里可能有用：

```
s = “My string”
print(s.lower())
>>> ‘my string’
```

测试：如果计算单词分数的功能实现正确，运行 test_ps3.py，则 test_get_word_score（）测试将通过。也应该测试get_word_score的函数，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，请认真阅读问题 2**，尽管如下描述的大多数函数已经提供了。**

组词（Representing hands）

组词（The Hand）是游戏玩家持有的一组字母，最初是一组随机字母。例如，玩家可以从以下开始：a，q，l，m，u，i，l。在程序中，hand 被表示为（python）字典：（字典的）键（keys）是（小写）字母，值（字典的 values）是该 hand 重复特定字母的次数。例如，上述 hand 可以表示为字典：

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

请注意 hand 中的重复的字母"l"是如何表示的。

使用字典表示形式，我们已经知道其访问值（values）的方法是 hand['a']，其中"a"是想要找到的键。但是，仅当该键是在字典中时才有效，否则会出错。

为了避免这种情况，可以改为使用函数调用 hand.get('a',0)。如果不确定键是否在字典中，则这是"安全"的访问方法。d.get（键，默认）返回键的值（如果键位于字典 d 中。如果键不在字典里，它返回"None"，get 方法不会引发 KeyError。

将字词转换为字典表示形式

已被定义的一个有用函数 get_frequency_dict，位于 ps3.py 顶部。当给定一串字母作为输入时，它返回一个字典，其中键是字母，并且值是该字母在输入字符串中表示的次数。例如：

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

如此，就是字典 hand 的形式。

显示一个 hand

给定一个表示为字典的 hand，希望能够以用户友好的方式显示它。display_hand 函数提供了这方面的实现。请仔细阅读此函数，了解它的作用及其工作原理。

产生一个随机 hand（Generating a random hand）

hand 是随机选择的一组字母，通过 deal_hand 函数，生成随机字典 hand。该函数采用正整数 n 作为输入，并返回表示 n 个小写字母 hand 的新字典。同样，请仔细阅读此函数，并了解它的作用及其工作原理。

从 hand 中取出字母：请编写代码实现之

玩家从 n 个字母的 hand 开始，当拼出单词时，来自该集的字母已被使用。例如，hand 如下所示：

a、q、l、m、u、i、l

玩家可以选择拼出单词"quail"，在 hand 中还留下的字母是：l，m。

（1）编写一个函数 `update_hand`：以 hand 和一个单词作为输入，使用该 hand 的字

母拼写单词，并返回一个新 hand 只包含剩余的字母。如此若成功的话，该函数不应修改原来的 hand。例如：

```
>>> hand = {'a':1, 'q':1, 'l':2, 'm':1, 'u':1, 'i':1}
>>> display_hand(hand)
a q l l m u i
          ## 请设计函数update_hand，输入为hand和一个单词
>>> new_hand = update_hand(hand, 'quail')
>>> 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
```

注：在上例中，调用 update_hand 后，new_hand 的值可以是字典{'a':0,"q":0,"l":1",'m':1,"u":0,"i":0}。具体的值取决于设计和实现；无论哪种情况，display_hand()的前后输出都应该是相同。

重要提示：

（2）另一方面，如果 update_hand 中输入的是一个无效的单词，要么因为它不是一个真正的单词，要么是因为其使用的字母有不在 hand 中的，这种情况将失去这个词的位于 hand 中的字母，以作为猜错的惩罚。请确保上述功能的实现！

不要假定给出的单词可以且只使用存在于 hand 中的字母。例如：

```
>>> hand = {'j':2, 'o':1, 'l':1, 'w':1, 'n':2}
>>> display_hand(hand)
j j o l w n n
>>> hand = update_hand(hand, 'jolly')
       ## 单词 jolly 不能在 hand 中被组词，因为 y 不在 head 中
>>> hand
{'j':1, w':1, 'n':2}
>>> display_hand(hand)  ## 原来 hand 中的 j o l l 被移走（remove）
j w n n
```

请根据代码中的规范实现 update_hand 函数。

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

测试：

确保test_update_hand测试通过。可能要还需要其他合理的输入来测试update_hand的实现。

问题 3： 有效单词（Valid words）

此时，还没有编写任何代码来验证玩家给出的单词是否遵守规则。***有效单词***是指位于单词列表中的（忽略单词大小写）的、完全由 hand 中的现有字母组成。

请编写is_valid_word 函数的代码

测试：

请使用 test_is_valid_wordtests 测试 is_vaild_word 函数。可能还应使用其他合理的输入来测试。另外请思考：如果希望通过多次调用实现测试，如果做？

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，虽然在评分时它确实算作已用或未使用的字母。

在游戏中，希望使用通配符的玩家应输入"*"（无引号），而不是预期字母。单词验证代码不应对预期元音应是什么进行任何假设，而应验证至少可以使用通配符作为所需位置的元音创建一个有效单词。

如下的例子（请仔细阅读并理解之），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
```

请修改 deal_hand 函数以支持始终在 hand 中提供一个通配符。请注意，deal_hand 目前确保三分之一的字母是元音，其余字母是辅音。保持辅音计数不变，并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。

然后修改 is_valid_word 函数以支持通配符。提示：检查通过用其他元音替换通配符可以形成哪些可能的单词。您可能需要查看字符串模块 find()函数的文档，并在找不到字符时记下其行为。在文件顶部为所定义的常量 VOWELS ，也许对你编程有帮助。

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，玩家必须键入"!!"（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）

游戏可以使用多个 hands。需要**设计并实现 substitute_hand 和 play_game 函数**。应考虑用 HAND_SIZE 常量来确定 Hand 中的字母数。如前所述，不要以为 hand 是固定的 7 个字母！设计要求是保持代码模块化 -- 如果文字游戏为 10 个或 4 个字母，只需改变 HAND_SIZE 的价值！

实现替换时，可能需要检查与字典（如 .key）关联的方法/函数，或查看 del 关键字。您可能还希望查看 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。

请注意，这里没有为此问题提供伪代码。但是，当设计这些函数时，应该先考虑这些函数的编写要求。

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！