DEALING WITH HANDS (10/10 points)

Please read this problem entirely!! The majority of this problem consists of learning how to read code, which is an incredibly useful and important skill. At the end, you will implement a short function. Be sure to take your time on this problem - it may seem easy, but reading someone else's code can be challenging and this is an important exercise.

REPRESENTING HANDS

A **hand** is the set of letters held by a player during the game. The player is initially dealt a set of random letters. For example, the player could start out with the following hand: `a, q, l, m, u, i, l`. In our program, a hand will be represented as a dictionary: the keys are (lowercase) letters and the values are the number of times the particular letter is repeated in that hand. For example, the above hand would be represented as:

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

Notice how the repeated letter `'l'` is represented. Remember that with a dictionary, the usual way to access a value is `hand['a']`, where `'a'` is the key we want to find. However, this only works if the key is in the dictionary; otherwise, we get a `KeyError`. To avoid this, we can use the call `hand.get('a',0)`. This is the "safe" way to access a value if we are not sure the key is in the dictionary. `d.get(key,default)` returns the value for `key` if `key` is in the dictionary `d`, else `default`. If `default` is not given, it returns `None`, so that this method never raises a `KeyError`. For example:

```
>>> hand['e']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'e'
>>> hand.get('e', 0)
0
```

CONVERTING WORDS INTO DICTIONARY REPRESENTATION

One useful function we've defined for you is `getFrequencyDict`, defined near the top of `ps4a.py`. When given a string of letters as an input, it returns a dictionary where the keys are letters and the values are the number of times that letter is represented in the input string. For example:

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

As you can see, this is the same kind of dictionary we use to represent hands.

DISPLAYING A HAND

Given a hand represented as a dictionary, we want to display it in a user-friendly way. We have provided the implementation for this in the `displayHand` function. Take a few minutes right now to read through this function carefully and understand what it does and how it works.

GENERATING A RANDOM HAND

The hand a player is dealt is a set of letters chosen at random. We provide you with the implementation of a function that generates this random hand, `dealHand`. The function takes as input a positive integer `n`, and returns a new object, a hand containing `n` lowercase letters. Again, take a few minutes (right now!) to read through this function carefully and understand what it does and how it works.

REMOVING LETTERS FROM A HAND (YOU IMPLEMENT THIS)

The player starts with a hand, a set of letters. As the player spells out words, letters from this set are used up. For example, the player could start out with the following hand: **`a, q, l, m, u, i, l`**. The player could choose to spell the word **`quail`** . This would leave the following letters in the player's hand: **`l, m`**. Your task is to implement the function `updateHand`, which takes in two inputs - a `hand` and a `word` (string). `updateHand` uses letters from the hand to spell the word, and then returns a copy of the `hand`, containing only the letters remaining. For example:

```
>>> hand = {'a':1, 'q':1, 'l':2, 'm':1, 'u':1, 'i':1}
>>> displayHand(hand) # Implemented for you
a q l l m u i
>>> hand = updateHand(hand, 'quail') # You implement this function!
>>> hand
{'a':0, 'q':0, 'l':1, 'm':1, 'u':0, 'i':0}
>>> displayHand(hand)
l m
```

Implement the `updateHand` function. Make sure this function has no side effects: i.e., it must not mutate the hand passed in. Before pasting your function definition here, be sure you've passed the appropriate tests in `test_ps4a.py`.

Hints

Testing

Testing: Make sure the `test_updateHand()` tests pass. You will also want to test your implementation of `updateHand` with some reasonable inputs.

Copying Dictionaries

You may wish to review the ".copy" method of Python dictionaries (review this and other Python dictionary methods here).

Help

Your implementation of updateHand should be short (ours is 4 lines of code). It does not need to call any helper functions.

Canopy specific instructions: If you modify code in `ps4a.py` go to

```
Run -> Restart Kernel (or hit the CTRL with the dot on your keyboard)
```

before running `test_ps4a.py`. **You have to do this every time you modify the file** `ps4a.py` **and want to run the file** `test_ps4a.py`, otherwise changes to the former will not be incorporated in the latter.

```
def updateHand(hand, word):
    listy = []
    nhand = hand.copy()

    for i in range(len(word)):
       listy.append(word[i])


    for x in listy:
       for k,v in nhand.items():
          if x == k:
            v -= 1
            nhand[k]=v
    return nhand

```

Correct

Test results

Hide output

CORRECT

Test 1

Function call: updateHand({'a': 1, 'i': 1, 'm': 1, 'l': 2, 'q': 1, 'u': 1}, quail)

Output:

```
{'a': 0, 'q': 0, 'u': 0, 'i': 0, 'm': 1, 'l': 1}
```

Test 2

Function call: updateHand({'a': 2, 'c': 2, 'l': 2, 'p': 3, 'r': 2, 't': 2}, claptrap)

Output:

```
{'a': 0, 'p': 1, 'c': 1, 'r': 1, 't': 1, 'l': 1}
```

Test 3

Function call: updateHand({'g': 1, 'd': 1, 'o': 1}, dog)

Output:

```
{'o': 0, 'd': 0, 'g': 0}
```

```
Test 4
```

Re-testing last test to see if you mutate the original hand

Output:

```
{'o': 0, 'd': 0, 'g': 0}
```

```
Test 4
```

Function call: updateHand({'q': 3, 'i': 3, 'r': 3, 'e': 3, 't': 3, 'w': 3, 'p': 3, 'y': 3, 'u': 3, 'o': 3}, typewriter)

Output:

```
{'e': 1, 'i': 2, 'o': 3, 'q': 3, 'p': 2, 'r': 1, 'u': 3, 't': 1, 'w': 2, 'y': 2}
```

```
Random Test 1
```

Function call: updateHand({'a': 1, 'e': 2, 'f': 1, 'k': 1, 'l': 1, 'p': 2, 'w': 1, 'y': 1}, apple)

Output:

```
{'a': 0, 'e': 1, 'f': 1, 'k': 1, 'l': 0, 'p': 0, 'w': 1, 'y': 1}
```

```
Random Test 2
```

Re-testing last test to see if you mutate the original hand

Output:

```
{'a': 0, 'e': 1, 'f': 1, 'k': 1, 'l': 0, 'p': 0, 'w': 1, 'y': 1}
```

```
Random Test 3
```

Function call: updateHand({'b': 1, 'h': 1, 'm': 1, 'l': 1, 'o': 1, 'p': 1, 'u': 1, 'y': 2}, plum)

Output:

```
{'b': 1, 'h': 1, 'm': 0, 'l': 0, 'o': 1, 'p': 0, 'u': 0, 'y': 2}
```

```
Random Test 4
```

Re-testing last test to see if you mutate the original hand

Output:

```
{'b': 1, 'h': 1, 'm': 0, 'l': 0, 'o': 1, 'p': 0, 'u': 0, 'y': 2}
```

```
Random Test 5
```

Function call: updateHand({'c': 1, 'e': 3, 'g': 1, 'f': 1, 'h': 1, 'k': 1, 't': 2}, teeth)

Output:

```
{'c': 1, 'e': 1, 't': 0, 'g': 1, 'f': 1, 'h': 0, 'k': 1}
```

```
Random Test 6
```

Re-testing last test to see if you mutate the original hand

Output:

```
{'c': 1, 'e': 1, 't': 0, 'g': 1, 'f': 1, 'h': 0, 'k': 1}
```

```
Random Test 7
```

Function call: updateHand({'m': 1, 'l': 2, 'o': 1, 'p': 1, 'r': 1, 'u': 1, 't': 1, 'v': 1}, plum)

Output:

```
{'m': 0, 'l': 1, 'o': 1, 'p': 0, 'r': 1, 'u': 0, 't': 1, 'v': 1}
```

```
Random Test 8
```

Re-testing last test to see if you mutate the original hand

Output:

```
{'m': 0, 'l': 1, 'o': 1, 'p': 0, 'r': 1, 'u': 0, 't': 1, 'v': 1}
```

Hide output

Check Save *You have used 5 of 30 submissions*

Show Discussion New Post

EdX offers interactive online classes and MOOCs from the world's best universities. Online courses from MITx, HarvardX, BerkeleyX, UTx and many other universities. Topics include biology, business, chemistry, computer science, economics, finance, electronics, engineering, food and nutrition, history, humanities, law, literature, math, medicine, music, philosophy, physics, science, statistics and more. EdX is a non-profit online initiative created by founding partners Harvard and MIT.

About edX

About

News

Contact

FAQ

edX Blog

Donate to edX

Jobs at edX

Follow Us

Twitter

Facebook

Meetup

LinkedIn

Google+