



Bookmarks



Bookmark

► Overview

► Entrance
Survey

► Week 1

► Week 2

► Week 3

▼ Week 4

Lecture 7 -
Debugging - Time
48:59

Lecture Sequence

Lecture 8 -
Assertions and
Exceptions - Time
34:58

Lecture Sequence

Complete
Programming
Experience: Python
Loves Fruits**Problem Set 4**Problem Set due Jul 07,
2016 at 23:30 UTC

► Sandbox

Week 4 > Problem Set 4 > Dealing with hands

Dealing with Hands

(10 points possible)

****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

Copying Dictionaries

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.

```
1 def updateHand(hand, word):
2     """
3     Assumes that 'hand' has all the letters in word.
4     In other words, this assumes that however many times
5     a letter appears in 'word', 'hand' has at least as
6     many of that letter in it.
7
```

Unanswered

Test Cases Bug: The test cases take in a dictionary and a string parameter go update hand. For example, `updateHand({'a': 1, 'i': 1, 'm': 1, 'l': 2, 'q': 1, 'u': 1}, "quail")`

You have used 0 of 30 submissions

© All Rights Reserved



© edX Inc. All rights reserved except where noted. EdX, Open edX and the edX and Open EdX logos are registered trademarks or trademarks of edX Inc.

POWERED BY
OPENedX

