Your grade: 100%

Your latest: **100%** • Your highest: **100%** • To pass you need at least 70%. We keep your highest score.

Next item →

1. Which of the following expressions corresponds to a dictionary with no elements? 1 / 1 point

- [x] `dict()`
  - Correct
- [x] `{}`
  - Correct
- [ ] `[]`
- [ ] `None`

2. Given an existing dictionary `favorites`, what Python statement adds the key `"fruit"` to this dictionary with the corresponding value `"blackberry"`? 1 / 1 point

- ( ) `favorites = {"fruit" : "blackberry"}`
- (•) `favorites["fruit"] = "blackberry"`
- ( ) `favorites["fruit" = "blackberry"]`
- ( ) `favorites["fruit" : "blackberry"]`
- ( ) `favorites{"fruit" : "blackberry"}`

Correct

3. Which of the expressions below returns `True` when the dictionary `my_dictionary` contains the key `my_key` and `False` otherwise? 1 / 1 point

- [x] `my_key in my_dictionary`
  - Correct
- [ ] `my_dictionary.key(my_key)`
- [ ] `my_dictionary.has_key(my_key)`
- [ ] `my_dictionary[my_key]`

4. *Keys* in a dictionary can have which of the following types? 1 / 1 point

- [x] tuple
  - Correct
  - Tuples are immutable. Use them in place of lists as keys.
- [x] string
  - Correct
  - Strings are immutable.
- [x] int
  - Correct
  - Integers are immutable.
- [ ] list

5. *Values* in a dictionary can have which of the following types? 1 / 1 point

- [x] string
  - Correct
- [x] tuple
  - Correct
- [x] bool
  - Correct
- [x] dict
  - Correct

6. Consider the following dictionary: 1 / 1 point

```
instructor_ratings = {"Joe" : "awesome", "Scott" : "hmmm..."}
```

What happens when Python evaluates the expression `instructor_ratings["John"]`?

- (•) Since `"John"` is not a key in the dictionary, Python raises a KeyError exception.
- ( ) Python returns the value `None` since `"John"` is not a key in the dictionary.
- ( ) Since `"John"` is not a key in the dictionary, Python raises a syntax error.
- ( ) Since `"John"` is not a value in the dictionary, Python raises a KeyError exception.

Correct

7. Write a function `count_letters(word_list)` that takes as input a list of words that are composed entirely of lower case letters . This function should return the lower case letter that appears most frequently (total number of occurrences) in the words in `word_list`. (In the case of ties, return the earliest letter in alphabetical order.) 1 / 1 point

The Python code snippet below represents a start at implementing `count_letters` using a dictionary `letter_count` whose keys are the lower case letters and whose values are the corresponding number of occurrences of each letter in the strings in `word_list`.

```
def count_letters(word_list):
    """ See question description """

    ALPHABET = "abcdefghijklmnopqrstuvwxyz"

    letter_count = {}
    for letter in ALPHABET:
        letter_count[letter] = 0

    # enter code here
```

Complete your implementation of `count_letters` based on this snippet. As a test, `count_letters(["hello", "world"])` should return the letter `'l'` since `'l'` appears 3 times total in the strings `"hello"` and `"world"`.

When you are confident in your code, compute the lower case letter return by `count_letters(monty_words)` where `monty_words` is defined as shown.

```
monty_quote = "listen strange women lying in ponds distributing swords is no basis for

monty_words = monty_quote.split(" ")
```

Enter this **single letter** in the text box below. Do not include any spaces or enclosing quotes around the letter.

```
e
```

Correct
The letter `'e'` appears 20 times in the quote.