Dictionaries

In Python, a ***dictionary*** stores "`key: value`" pairs. In the following assignment, the key:value pairs are separated by commas and wrapped in curly braces. For example:

```
elements = {'C': 'carbon', 'H': 'hydrogen', 'O': 'oxygen', 'N': 'nitrogen'}
```

Key	Value
`'C'`	`'carbon'`
`'H'`	`'hydrogen'`
`'O'`	`'oxygen'`
`'N'`	`'nitrogen'`

In contrast to sequence types, a dictionary is a ***mapping*** type. Values are referenced by ***keys***, rather than by consecutive integer indexes.

Evaluate the following code statements in the order they are listed:

Python code	Output
`type(elements)`	<class 'dict'>
`elements.keys()`	dict_keys(['C', 'H', 'O', 'N'])
`elements.values()`	dict_values(['carbon', 'hydrogen', 'oxygen', 'nitrogen'])
`elements['C']`	'carbon'
`atom = 'N'`	
`elements[atom]`	'nitrogen'
`elements[N]`	NameError: name 'N' is not defined
`elements['nitrogen']`	KeyError: 'nitrogen'
`elements[1]`	KeyError: 1
`len(elements)`	4
`elements['B'] = 'boron'`	

1. List all the keys stored in the `elements` dictionary after completing the table.

The keys are `'C'`, `'H'`, `'N'`, `'B'`, and `'O'`.

2. What is the data type of the keys in the `elements` dictionary?

The keys are all strings. (Note: there is no "char" type in Python.)

3. Explain the reason for the error after entering each of the following lines:

a) `elements[N]` The letter N is treated as a variable; it's undefined.

b) `elements['nitrogen']` The string `"nitrogen"` is not one of the keys.

c) `elements[1]`

The integer 1 is also not a key, and a dictionary is not a sequence and does not use numeric indexes.

4. Write a Python expression that creates a dictionary for the seven days of the week, i.e., Sun=1, Mon=2, Tue=3, etc. Assign the dictionary to the variable `dow`.

```
dow = {'Sun': 1, 'Mon': 2, 'Tue': 3, 'Wed': 4, 'Thu': 5, 'Fri': 6, 'Sat': 7}
```

5. If you assign two different values to the same key (i.e., two assignment statements with one value each), which value is stored in the dictionary? Justify your answer with an example.

If you were to assign `dow['Sun'] = 8` followed by `dow['Sun'] = 0`, then 0 would replace the previous value.

6. Another way to store the data is to use two lists:

```
keys = ['C', 'H', 'O', 'N']
vals = ['carbon', 'hydrogen', 'oxygen', 'nitrogen']
```

What is a disadvantage of this approach? Explain your reasoning.

It's more difficult to insert new items: you have to write two assignment statements instead of one. It's even more difficult to update items: you have to determine the index of the key and replace the corresponding value in the other list.

7. Write down code that check that the value stored by variable `target` is a key in the dictionary `my_dict`:

```
if target in my_dict:
    # this code executes when the key is in the dictionary
else:
    # this code executes when the key is NOT in the dictionary
```

8. Write down code that check that the value stored by variable `target` is a value in the dictionary `my_dict`:

```python
for key in my_dict:
    if my_dict[key] == target:
        # this code executes when the key is in the dictionary
    else:
        # this code executes when the key is NOT in the dictionary
```

9. Below is a portion of the pirate dictionary. Write a function called `translate` that takes in the pirate dictionary and a phrase in English and returns its translation in pirate. For example, `translate(pirate_dict,'hello there students')` must return `'avast there swabbies'`.

English Word	Pirate Translation	English Word	Pirate Translation
'sir'	'matey'	'excuse'	'arr'
'hotel'	'fleabag inn'	'are'	'be'
'student'	'swabbie'	'lawyer'	'foul blaggart'
'students'	'swabbies'	'the'	'th''
'boy'	'matey'	'restroom'	'head'
'madam'	'proud beauty'	'my'	'me'
'professor'	'foul blaggart'	'hello'	'avast'
'restaurant'	'galley'	'is'	'be'
'your'	'yer'	'man'	'matey'

```python
def translate(pirate_dict: dict, phrase: str) -> str:
    phrase_pirate = ''
    for word in phrase.split():
        if word in pirate_dict:
            phrase_pirate += pirate_dict[word] + ' '
        else:
            phrase_pirate += word + ' '
    return phrase_pirate.strip()
```

10. Write a function called `freq_counts` that takes in a string and then prints a table of the letters of the alphabet in alphabetical order which occur in the string together with the number of times each letter occurs. Case should be ignored. A sample run of the program might look this this:

```
freq_counts('ThiS is String with Upper and lower case Letters.')

a  2
c  1
d  1
e  5
...
```

```python
def freq_counts(text: str) -> None:
    freq_dict = {}
    text = text.lower()
    for i in range(26):
        freq_dict[chr(97+i)] = 0
    for i in text:
        if i.isalpha():
            freq_dict[i] +=1
    for k in freq_dict:
        if freq_dict[k] != 0:
            print(k, freq_dict[k])
```

11. Rewrite the previous problem assuming that the order of the letters did not matter (try to think of a different more optimal implementation that does not keep track of every letter, only letters that have been seen):

```python
def freq_counts(text: str) -> None:
    freq_dict = {}
    text = text.lower()
    for i in text:
        if i.isalpha():
            if i not in freq_dict:
                freq_dict[i] = 0
            freq_dict[i] +=1

    for k in freq_dict:
        print(k, freq_dict[k])
```