counting_treasure_v1_r1

September 14, 2022

[96]:
```
def identify_type(d, result):
    """This method is using to identify thr types of values of a dictionary.
        We suppose *d* is the data we want to identify.
        We suppose *result* is a counter we passed in.
    """

    if isinstance(d, dict):
        for key, value in d.items():
            if isinstance(d[key], int):
                result.update({key: d[key]})
            else:
                result.update({key: len(d[key])})
                identify_type(d[key], result)
    elif isinstance(d, list):
        for item in d:
            if not isinstance(item, dict):
                raise TypeError("\'int\' object is not iterable")
            identify_type(item, result)
    elif isinstance(d, tuple):
        for item in d:
            if not isinstance(item, dict):
                raise TypeError("\'int\' object is not iterable")
            identify_type(item, result)

    return result
```

[89]:
```
from collections import Counter

def dict_sort(d):
    result = {}
    for k in sorted(d.keys()):
        # in modern Python, dicts remember the order in which their keys
        # were added, and use that order when being printed
        result[k] = d[k]
    return result

def count_treasure(box):
    # HINT: use a `Counter` to store your results while working
    result = Counter()

    ## YOUR CODE HERE
    result =  identify_type(box, result)


    # HINT: use `dict_sort(result)` at the end to sort and
    # convert to an ordinary `dict`
    return dict_sort(result)
```

[ ]:
```
import doctest
doctest.testmod()
```

[ ]:
```

```

[97]:
```
a = {'coins': 10,
     'bags': [{'coins': 2}, {'coins': 5}]
    }

# output: {'bags': 2, 'coins': 17}

count_treasure(a)
```

[97]:
```
{'bags': 2, 'coins': 17}
```

[98]:
```
b = {'coins': 10, 'diamonds': 10}

# output {'coins': 10, 'diamonds': 10}

count_treasure(b)
```

[98]:
```
{'coins': 10, 'diamonds': 10}
```

[99]:
```
c = {'bags': [{'bags': [{'coins': 10}]}]}

# output {'bags': 2, 'coins': 10}
count_treasure(c)
```

[99]:
```
{'bags': 2, 'coins': 10}
```

[100]:
```
d = {
    'coins': 10,
                'rubies': 10,
                'enchanted pouches': [{
                    'coins': 10,
                    'rubies': 10,
                    'treasure chests': (
                        {'coins': 1000},
                        {'coins': 1000},
                        {'coins': 1000}
                    ) # this was a tuple of 3 treasure chests
                }] # this was a list of 1 enchanted pouches
               }

# output {'coins': 3020, 'enchanted pouches': 1, 'rubies': 20, 'treasure␣
 ↪chests': 3}

count_treasure(d)
```

```
[100]: {'coins': 3020, 'enchanted pouches': 1, 'rubies': 20, 'treasure chests': 3}
```

```
[92]: e = {'bags': (10, 20, 30)}

      type(e)
```

```
[92]: dict
```

```
[101]: count_treasure(e)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [101], in <cell line: 1>()
----> 1 count_treasure(e)

Input In [89], in count_treasure(box)
     14 result = Counter()
     16 ## YOUR CODE HERE
---> 17 result =  identify_type(box, result)
     20 # HINT: use `dict_sort(result)` at the end to sort and
     21 # convert to an ordinary `dict`
     22 return dict_sort(result)

Input In [96], in identify_type(d, result)
     11         else:
     12             result.update({key: len(d[key])})
---> 13             identify_type(d[key], result)
     14 elif isinstance(d, list):
     15     for item in d:

Input In [96], in identify_type(d, result)
     20     for item in d:
     21         if not isinstance(item, dict):
```

```
---> 22             raise TypeError("\'int\' object is not iterable")
     23         identify_type(item, result)
     25 return result

TypeError: 'int' object is not iterable
```

```
[ ]:
```