Congratulations! You passed!

TO PASS 70% or higher

Keep Learning

GRADE
85.71%

Nested Representations for Tabular Data

LATEST SUBMISSION GRADE

85.71%

1. Given a dictionary `my_dict` and a possible key `my_key`, which expression below returns the same result as the expression `my_key in my_dict`? **1 / 1 point**

- [ ] `my_key in my_dict.items()`
- [x] `my_key in my_dict.keys()`
- [ ] `my_dict contains my_key`
- [ ] `my_key in my_dict.values()`

Correct

The `keys()` method returns a list of keys.

2. We often want to loop over all the key/value pairs in a dictionary. Assume the variable `my_dict` stores a dictionary. One way of looping like this is as follows: **1 / 1 point**

```
for key in my_dict:
    value = my_dict[key]
    …
```

However, there is a better way. We can instead write the following:

```
for key, value in ???:
    …
```

What code should replace the question marks so that the two forms are equivalent?

- [x] `my_dict.items()`
- [ ] `my_dict.keys_values()`
- [ ] `items(my_dict)`
- [ ] `my_dict.keys()`

Correct

3. Consider the following dictionary in Python. **1 / 1 point**

```
my_dict = {0 : 0, 5 : 10, 10 : 20, 15 : 30, 20 : 40}
```

What is the difference between the expressions my_dict[25] and my_dict.get(25)?

- Both expressions return the value None since 25 is not a valid key in the dictionary.
- The expression my_dict[25] raises a KeyError since 25 is not a valid key while the expression my_dict.get(25) returns None in this case.
- Both expressions raise KeyErrors since 25 is not a valid key in the dictionary.
- Both expressions return the value 50.

Correct

Note that the get() method also takes an optional second parameter that is the returned value if the supplied key not in the dictionary.

4. Two-dimensional mathematical data structures can be easily represented as a list of lists in Python. A matrix is a rectangular array of items arranged in vertical rows and horizontal columns. The following snippet of Python code generates and prints a list of lists that models a matrix with three rows and five columns. **1 / 1 point**

```
NUM_ROWS = 3
NUM_COLS = 5

# construct a matrix
my_matrix = []
for row in range(NUM_ROWS):
    new_row = []
    for col in range(NUM_COLS):
        new_row.append(row * col)
    my_matrix.append(new_row)

# print the matrix
for row in my_matrix:
    print(row)
```

Mathematically, each entry in a matrix can be indexed by its corresponding row number and column number where these indices start at one.

Which Python expression below returns the value of the entry in the second row and fifth column of the matrix my_matrix?

- my_matrix[2][5]
- my_matrix[4][1]
- my_matrix[5][2]

Correct

5. A matrix is *square* if it has the same number of rows and columns. The *diagonal* of a square matrix consists of those items in the matrix whose row and column indices are equal. Finally, the trace of a matrix is the sum of the items on the matrix's diagonal.

0 / 1 point

Write a function trace(matrix) that takes a square matrix matrix and returns the value of its trace. Then use your implementation of trace() and compute the value of trace(my_matrix) for instances of my_matrix as defined by the code snippet provided in the previous question.

As test, trace(my_matrix) should return 30 when trace(my_matrix) has five rows and columns. Enter in the box below the value returned by trace(my_matrix) when trace(my_matrix) has twenty five rows and columns.

150

Incorrect

6. Dictionaries and lists can also be used in combination to create representations for 2D data in Python. As in the case of lists of lists, individual items in the 2D data structure can be referenced via two indices.

1 / 1 point

Which of the expressions below are **dictionaries of lists** (i.e. dictionaries whose values are lists)?

- [ ] {'d' : "def", 'e' : "efg", 'f' : "fgh"}
- [ ] [{0 : 0, 1 : 0, 2 : 0}, {0 : 0, 1 : 1, 2 : 2}, {0 : 0, 1 : 2, 2 : 4}]
- [x] {0 : [], 1 : [1], 2 : [2, 2], 3 : [3, 3, 3]}

Correct
This expression is a dictionary has values that are list of integers.

- [x] {'a' : ['a', 'b', 'c'], 'b' : ['b', 'c', 'd'], 'c' : ['c', 'd', 'e']}

This expression is a dictionary has values that are a lists of strings.

7. Finally, dictionaries of dictionaries can also be used to represent 2D tabular data such as matrices. The following snippet of Python code generates and prints a dictionary of dictionaries that models a matrix with three rows and five columns. **1 / 1 point**

```
NUM_ROWS = 3
NUM_COLS = 5

# construct a matrix
my_matrix = {}
for row in range(NUM_ROWS):
    row_dict = {}
    for col in range(NUM_COLS):
        row_dict[col] = row * col
    my_matrix[row] = row_dict

print(my_matrix)

# print the matrix
for row in range(NUM_ROWS):
    for col in range(NUM_COLS):
        print(my_matrix[row][col], end = " ")
    print()
```

Note that the same expression my_matrix[row][col] can be used to reference an entry in the matrix independent of whether the matrix is represented as a list of lists or a dictionary of dictionaries.

Which option below corresponds to the value of my_matrix as computed by the snippet above when NUM_ROWS = 5 and NUM_COLS = 9?

Hint: We highly recommend that you use copy and paste to transfer each option into your Python IDE to aid in answering this problem. Remember that you can use the equal operator == to compare objects in Python.

- [x]
```
{2: {6: 12, 2: 4, 0: 0, 7: 14, 5: 10, 3: 6, 8: 16, 4: 8, 1: 2}, 4: {0: 0, 3: 12,
  2: 8, 6: 24, 4: 16, 5: 20, 8: 32, 7: 28, 1: 4}, 1: {2: 2, 5: 5, 3: 3, 8: 8, 4:
  4, 1: 1, 7: 7, 0: 0, 6: 6}, 3: {4: 12, 0: 0, 8: 24, 6: 18, 7: 21, 3: 9, 5: 15,
  1: 3, 2: 6}, 0: {8: 0, 1: 0, 6: 0, 2: 0, 4: 0, 5: 0, 3: 0, 0: 0, 7: 0}}
```

- [ ]
```
[[0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 1, 2, 3, 4, 5, 6, 7, 8], [0, 2, 4, 6, 8, 10,
  12, 14, 16], [0, 3, 6, 9, 12, 15, 18, 21, 24], [0, 4, 8, 12, 16, 20, 24, 28,
  32]]
2
```

- [ ]
```
{1: {1: 2, 7: 2, 3: 2, 8: 2, 0: 2, 5: 2, 2: 2, 6: 2, 4: 2}, 2: {5: 4, 1: 4, 8: 4
  , 0: 4, 3: 4, 2: 4, 4: 4, 7: 4, 6: 4}, 3: {4: 6, 2: 6, 1: 6, 5: 6, 0: 6, 7: 6,
  8: 6, 3: 6, 6: 6}, 0: {0: 0, 5: 0, 6: 0, 8: 0, 1: 0, 3: 0, 2: 0, 4: 0, 7: 0},
  4: {3: 8, 8: 8, 7: 8, 5: 8, 4: 8, 1: 8, 2: 8, 6: 8, 0: 8}}
```

- [ ]
```
{1: {7: 7, 4: 4, 3: 3, 8: 8, 6: 6, 5: 5, 2: 2, 0: 0, 1: 1}, 0: {0: 0, 7: 0, 3: 0
  , 4: 0, 8: 0, 6: 0, 5: 0, 1: 0, 2: 0}, 2: {0: 0, 8: 16, 5: 10, 2: 4, 7: 14, 4:
  8, 1: 2, 3: 6, 6: 12}, 3: {1: 3, 7: 21, 2: 6, 8: 24, 3: 9, 4: 12, 6: 18, 0: 0,
  5: 18}, 4: {3: 12, 7: 28, 0: 0, 2: 8, 1: 4, 4: 16, 6: 24, 5: 20, 8: 32}}
```

✓ **Correct**