Learning Quiz 19: Dictionaries

Due Oct 30 at 1pm **Points** 5 **Questions** 5 **Available** until Dec 4 at 11:59pm **Time Limit** None
Allowed Attempts Unlimited

Instructions

Prior to completing this quiz, be sure to read:

- Section 6.1: Dictionaries (p. 166-177)

Please also go over Practice Problems 6.1-6.5 in the textbook (solutions at the end of the chapter) before attempting this quiz.

This quiz was created for learning purposes. You may attempt this quiz as many times as you need. Only submissions score *prior to the deadline* will count towards the final course grade. No late submissions will be accepted.

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	14 minutes	5 out of 5

Score for this attempt: **5** out of 5
Submitted Nov 1 at 12:20pm
This attempt took 14 minutes.

Question 1

1 / 1 pts

Lists and tuples in Python are known as containers. Another built-in type of container in Python is the **dictionary**. Dictionaries store items that are accessible using programmer-specified indices. They are especially useful for case where we want to associate an item with a specific ID or key term (as opposed to indices 0 through n.

Suppose we have three students whose ID numbers and names are 12345 Amy Boe, 23456 Carter Doe, and 34567 Emily Goe. To create a dictionary called *student*, we might use the following code:

```
student = {12345: 'Amy Boe', 23456: 'Carter Doe', 34567: 'Emily Goe'}
```

Note that we used curly brackets, separated all items by commas, and entered each entry with ID-colon-name.

Now, student[1] will no longer work because there is no index 1. To access 'Carter Doe', we need to type student[23456].

Consider the following dictionary:

```
friends = {1111: 'Ross Geller', 2222: 'Monica Geller', 3333: 'Rachel Green', 4444: 'Phoebe Buffay', 5555:
'Chandler Bing', 6666: 'Joey Tribbiani'}
```

Which of the following lines of code gives us 'Rachel Green'?

- friends['Rachel Green']
- friends[2]
- friends[4444]
- friends[3]
- friends[4]
- friends
- friends[3333]

Correct!

Question 2

1 / 1 pts

The general format for creating dictionaries is:

{<key1>: <value1>, <key2>: <value2>, ..., <keyi>: <valuei>}

Note that keys can be strings, integers, floats, and tuples. **Lists cannot be keys.**

```
dictionary1 = {392000123: 'apple', 392000987: 'banana', 392123123: 'orange'}
dictionary2 = {3920001.23: ['April', 'Andreas'], 3920009.87: ['Bobby', 'Black'], 3921231.23: ['Carson', 'C
arie']}
dictionary3 = {('x1','y1'): ['April', 'Andreas'],
               ('x2','y2'): ['Bobby', 'Black'],
               ('x3','y3'): ['Carson', 'Carie']}
dictionary4 = {('April','Andreas'): 3102221234,
               ('Bobby','Black'): 3102225678,
               ('Carson','Carie'): 3105555555}
```

Values can be strings, integers, floats, tuples, and/or lists. From our example in question 1, we might want to separate the names so that we can easily access the first and last name separately:

```
student = {12345: ['Amy','Boe'], 23456: ['Carter','Doe'], 34567: ['Emily','Goe']}
```

We would like to create a dictionary such that the key is an English word and the values are a list of Spanish translations. Fill in the blanks with ONE special symbol each to achieve this goal:

engtospan = { 'red': ['rojo', 'roja'], 'blue' : ['azul'], 'yellow': ['amarillo', 'amarilla'] }

Answer 1:

Correct!

{

Answer 2:

Correct!

:

Answer 3:

Correct!

[

Answer 4:

Correct!

}

Question 3

1 / 1 pts

Here is a list of dictionary methods. (Be sure to these out on your own dictionaries to see what happens).

```
"""
clear()    Removes all the elements from the dictionary
copy()     Returns a copy of the dictionary
fromkeys() Returns a dictionary with the specified keys and values
get()      Returns the value of the specified key
items()    Returns a list containing a tuple for each key value pair
keys()     Returns a list containing the dictionary's keys
pop()      Removes the element with the specified key
popitem()  Removes the last inserted key-value pair
setdefault()   Returns the value of the specified key. If the key does not exist: insert the key, with the
specified value
update()   Updates the dictionary with the specified key-value pairs
values()   Returns a list of all the values in the dictionary
"""
```

Consider the following dictionary:

```
student = {12345: ['Amy','Boe'], 23456: ['Carter','Doe'], 34567: ['Emily','Goe']}
```

Match the result with the dictionary method.

Correct!

student.items() [(12345, ['Amy', 'Boe']), (2345

Correct!

student.keys() [12345, 23456, 34567]

Correct!

student.values() [['Amy', 'Boe'], ['Carter', 'Doe']

Other Incorrect Match Options:

- (12345, ['Amy', 'Boe']), (23456, ['Carter', 'Doe']), (34567, ['Emily', 'Goe'])
- ['Amy', 'Boe'], ['Carter', 'Doe'], ['Emily', 'Goe']

Question 4

1 / 1 pts

In dictionaries, multiple keys can have the same value:

```
dict_sameval = {('April','Andreas'): 3102221234,
                ('Bobby','Black'): 3102221234,
                ('Carson','Carie'): 3102221234}
```

However, multiple values cannot have the same key:

```
dict_samekey = {('April','Andreas'): 3102221234,
                ('April','Andreas'): 3102225678,
                ('April','Andreas'): 3105555555}
```

in the event that you accidentally create a dictionary with more than one of the same key, Python will look at the latest on. In dict_samekey, dict_samekey[('April','Andreas')] refers to 3105555555.

Consider the case where we accidentally create duplicate keys pointing to the same value. pop() removes the pair with the given key.

```
student = {12345: ['Amy','Boe'], 23456: ['Carter','Doe'], 12345: ['Emily','Goe'], 12345: ['Willy', 'Zoe']}
student.pop(12345)
```

What does student[12345] refer to now?

Correct!

- (●) nothing, pop() removed index 12345 in the dictionary completely
- ( ) ['Amy','Boe']
- ( ) ['Carter','Doe']
- ( ) ['Emily','Goe']

Question 5

1 / 1 pts

To use the update() method, we need to have a second dictionary to update with. In the following code, the update is automatically done to d1 and we do not need to re-store the update in d1:

```
d1 = {12345: ['Amy','Boe'], 23456: ['Carter','Doe'], 34567: ['Emily','Goe'], 45678: ['Willy', 'Zoe']}
d2 = {56789: ['Joe','Poe'], 67890: ['Manny','Roe'], 45678:['William','Zoe']}
d1.update(d2)
```

Both original dictionaries d1 and d2 have the key 45678. What happened to key 45678 after the update?

- ( ) The value in d1 replaced the value in d2 for key 45678
- ( ) Key 45678 was removed in d2 and updated in d1
- ( ) Key 45678 was removed in d1 and updated in d2
- ( ) Nothing changed in both dictionaries

Correct!

- The value in d2 replaced the value in d1 for key 45678

Quiz Score: **5** out of 5