**Module 10. Dictionaries and Sets**

**TRUE/FALSE**

1. A dictionary is an ordered collection of key-value pairs.

Answer: F

1. You can use both immutable and mutable data types as keys in a dictionary.

Answer: F

1. The dict.get() method returns the default value if the key is not found.

Answer: T

1. Using the del keyword can remove a key-value pair from a dictionary.

Answer: T

1. Sets in Python can contain duplicate elements.

Answer: F

1. Dictionaries are mutable, meaning you can change their content after creation.

Answer: T

1. The set.add() method can add multiple elements to a set at once.

Answer: F

1. Python sets maintain the order of elements as they were added.

Answer: F

1. The dict.update() method can be used to merge two dictionaries.

Answer: T

1. Removing an item from a set using discard() will raise an error if the item is not found.

Answer: F

1. The popitem() method in dictionaries returns an arbitrary key-value pair as a tuple.

Answer: T

1. You can create an empty set using curly braces {}.

Answer: F

1. The in keyword can be used to check for membership in a set.

Answer: T

1. The dict.keys() method returns a list of all keys in the dictionary.

Answer: F

1. You can use the pop() method to remove and return an arbitrary element from a set.

Answer: T

1. The set.intersection() method returns a new set with elements common to all sets.

Answer: T

1. A frozenset is a mutable version of a set.

Answer: F

1. Dictionaries in Python can have multiple entries with the same key.

Answer: F

1. Sets are particularly useful for membership tests and eliminating duplicates.

Answer: T

1. The dict.items() method returns a view object that displays a list of tuples, each containing a key-value pair.

Answer: T

1. The set.symmetric\_difference() method returns elements that are in either of the sets but not in both.

Answer: T

1. Using the dict.pop() method requires specifying the default value.

Answer: F

1. Sets in Python are implemented using hash tables.

Answer: T

1. You can iterate over the elements of a set using a for loop.

Answer: T

1. A dictionary’s popitem() method always removes the last inserted key-value pair.

Answer: F

**MULTIPLE CHOICE**

1. In a dictionary, you use a(n) \_\_\_\_\_\_\_\_\_\_ to locate a specific value.

a. datum

b. element

c. item

d. key

Answer: D

1. What is the correct structure to create a dictionary of months where each month will be accessed by its month number (for example, January is month 1, April is month 4)?

a. { 1 ; 'January', 2 ; 'February', ... 12 ; 'December'}

b. [ 1 : 'January', 2 : 'February', ... 12 : 'December' ]

c. [ '1' : 'January', '2' : 'February', ... '12' : 'December' ]

d. { 1, 2,... 12 : 'January', 'February',... 'December' }

Answer: B

1. What will be the result of the following code?

*ages = {'Aaron' : 6, 'Kelly' : 3, 'Abigail' : 1 }*

*value = ages['Brianna']*

a. False

b. -1

c. 0

d. KeyError

Answer: D

1. What is the number of the first index in a dictionary?

a. 0

b. 1

c. the size of the dictionary minus one

d. Dictionaries are not indexed by number.

Answer: D

1. Which method would you use to remove a key-value pair from a dictionary?

a. remove()

b. discard()

c. delete()

d. pop()

Answer: D

1. What will be the output of the following code?

*my\_set = {1, 2, 3, 4, 5}*

*my\_set.add(3)*

*print(len(my\_set))*

a. 4

b. 5

c. 6

d. 3

Answer: B

1. Which of the following methods can be used to add an element to a set?

a. add()

b. append()

c. insert()

d. update()

Answer: A

1. If you try to access a key that does not exist in a dictionary using square brackets, what happens?

a. Returns None

b. Raises a KeyError

c. Returns False

d. Returns an empty string

Answer: B

1. Which method can you use to get a list of all keys in a dictionary?

a. keys()

b. get\_keys()

c. key\_list()

d. all\_keys()

Answer: A

1. What does the set.union() method do?

a. Adds elements from another set

b. Returns a new set with elements from both sets

c. Removes elements present in another set

d. Finds common elements in both sets

Answer: B

1. What is the output of the following code?

*fruits = {"apple", "banana", "cherry"}*

*fruits.add("banana")*

*print(fruits)*

a. {'apple', 'banana', 'cherry', 'banana'}

b. {'apple', 'banana', 'cherry'}

c. {'apple', 'cherry'}

d. {'apple', 'banana', 'cherry', None}

Answer: B

1. Which method can be used to update a dictionary with elements from another dictionary?

a. add()

b. append()

c. extend()

d. update()

Answer: D

1. What will be the result of the following code?

*inventory = {'apples': 5, 'bananas': 8}*

*inventory['apples'] += 2*

*print(inventory['apples'])*

a. 5

b. 8

c. 7

d. 10

Answer: C

1. How can you create an empty set?

a. set()

b. {}

c. set([])

d. {[]}

Answer: A

1. Which of the following methods can be used to remove and return an arbitrary element from a set?

a. remove()

b. discard()

c. pop()

d. clear()

Answer: C

1. Which of the following is a valid key type for a Python dictionary?

a. list

b. set

c. dictionary

d. tuple

Answer: D

1. What is the output of the following code?

*d = {'a': 1, 'b': 2, 'c': 3}*

*print(d.get('d', 0))*

a. 1

b. 2

c. 3

d. 0

Answer: D

1. What is the purpose of the dict.pop() method?

a. Add a key-value pair

b. Update a key-value pair

c. Remove a key-value pair

d. Return the number of items

Answer: C

1. Which set operation will return elements that are in either set but not in both?

a. union()

b. intersection()

c. difference()

d. symmetric\_difference()

Answer: D

1. How can you check if a key exists in a dictionary?

a. if key in dict

b. if dict.contains(key)

c. if dict.has\_key(key)

d. if key.exists(dict)

Answer: A

1. What is the output of the following code?

*s = {1, 2, 3}*

*s.add(2)*

*print(s)*

a. {1, 2, 2, 3}

b. {1, 2, 3}

c. {1, 3}

d. {2, 3}

Answer: B

1. Which method removes all elements from a set?

a. delete()

b. clear()

c. remove()

d. discard()

Answer: B

1. What will be the result of the following code?

*d = {'a': 1, 'b': 2, 'c': 3}*

*d.update({'b': 4, 'd': 5})*

*print(d)*

a. {'a': 1, 'b': 2, 'c': 3}

b. {'a': 1, 'b': 4, 'c': 3}

c. {'a': 1, 'b': 4, 'c': 3, 'd': 5}

d. {'a': 1, 'b': 2, 'c': 3, 'd': 5}

Answer: C

1. What will be the result of the following code?

*my\_dict = {'a': 1, 'b': 2, 'c': 3}*

*print('d' in my\_dict)*

a. True

b. False

c. None

d. KeyError

Answer: B

1. How can you iterate over both keys and values of a dictionary?

a. for key, value in dict.items():

b. for key, value in dict.entries():

c. for key, value in dict.key\_values():

d. for key, value in dict.pairs():

Answer: A

1. What will be the output of the following code?

*my\_set = {1, 2, 3, 4, 5}*

*my\_set.remove(3)*

*print(my\_set)*

a. {1, 2, 3, 4, 5}

b. {1, 2, 4, 5}

c. {1, 2, 3, 5}

d. {2, 4, 5}

Answer: B

1. What method would you use to combine all unique elements from two sets?

a. combine()

b. merge()

c. union()

d. intersection()

Answer: C

1. What is the result of the following code?

*inventory = {'apples': 5, 'bananas': 8}*

*inventory.pop('apples')*

*print(inventory)*

a. {'bananas': 8}

b. {'apples': 5}

c. {'apples': 5, 'bananas': 8}

d. {}

Answer: A

1. Which method can be used to get the number of elements in a dictionary?

a. count()

b. length()

c. size()

d. len()

Answer: D

1. What does the set.difference() method do?

a. Returns elements that are only in the first set

b. Returns elements that are only in the second set

c. Returns elements common to both sets

d. Returns elements that are in either set but not both

Answer: A