**1.What is a tuple in Python? Provide an example of a tuple and explain its immutability.**

**A.** In python a tuple is collection of ordered and immutable data enclosed in parenthesis ().

as an example,

t = (1,2,3,5,2,7)

print(t)

--> (1,2,3,5,2,7)

Once a tuple created, we can't modify any elements of tuple.so tuples are Immutable.

This property ensures that the data inside a tuple remains constant throughout its lifetime.

**2.How can you add a new element to an existing tuple in Python? Explain with an example.**

A. In python, we can add a new element in existing tuple. It can be tricky due to the immutability of tuples. However, there are a few ways to achieve this:

(1). using + operator :

* You can create a new tuple by concatenating the original tuple with another tuple containing the new element.

t1 = (1,2,3)

t2 = 4

t1 = t1 + (t2,)

print(t1)

--> (1,2,3,4)

(2). using += operator:

t1 = (1,2,3)

t2 = 4

t1 += (t2,)

print(t1)

--> (1,2,3,4)

**3.What is the difference between a list and a tuple in Python? Explain with suitable examples.**

**A.** In python, differences between a list and a tuple are :

* Lists are created using square brackets [ ]. Tuples are created using parenthesis ().

list1 = [1,2,3,True,9]

tuple1 = (1,2,3,4,7)

* List is mutable. we can add, remove, assign any element in list. but tuples are immutable. we can't modify elements in tuple.

list1[2] = 4

print(list1) # output: (1,2,4,True,9)

# tuples[2] = 10 # we will get an error

## for tuples, we can't modify.

* Use Cases: we use lists when we need to perform operations like insertion, deletion, or modification. and we use tuples when we want to store fixed sequences of values that should not change.
* Lists are suitable for dynamic data. And Tuples are appropriate for data that remains constant.

**4.How can you convert a tuple to a list in Python? Provide a code example.**

**A.** we can convert a tuple to a list using list() function.

examples,

t = (1,2,3)

print(type(t)) #output: tuple

l = list(t)

print(l) #output: [1,2,3]

**5.Explain the concept of dictionary in Python. Provide an example of a dictionary and its key-value pairs.**

**A.** In python, a dictionary is a data structure that allows to store the data in key: value pairs. Each key corresponds to a specific value, and we can use the key to retrieve its associated value.

* dictionary is an unordered collection of items.
* keys are immutable. but values can be of any type.
* keys: tuple, boolean, int, float, string or variable name.
* Special case characters (like @, #, $, %) are not allowed as a key. only tuple can be key but list, set, dictionary are not allowed(unhashable).

examples,

d = {"name" : "sharmi" , "emiil\_id" : "ksharmi1998@gmail.com" , "number" :123456, 1: "abc"}

print(d["name"]) # output: "sharmi"

**6.How can you check if a key exists in a dictionary in Python? Provide a code example.**

**A.** we can check if a key exists in a dictionary in python using "in" operator. It will return boolean value.

d = {"name" : "sharmi" , "emiil\_id" : "ksharmi1998@gmail.com" , "number" :123456, 1: "abc"}

"name" in d.keys() # output: True

here, we are checking if the key "name" exists in the "d" dictionary keys.

If it returns True then we get to know the key exists.

**7.Explain the difference between 'del' and 'pop' when used with dictionaries in Python. Provide examples.**

**A.** In dictionary, when we use "del" method,

* it will remove the key:value pair. It doesn’t return any value, it simply removes the key along with its associated value.

my\_dict = {'apple': 5, 'banana': 3, 'cherry': 8}

# Removing the key 'banana' using del

del my\_dict['banana']

# Now my\_dict contains only {'apple': 5, 'cherry': 8}

when we use "pop" method,

* it also removes key:value pair form dictionary. but it returns the corresponding value of the removed key of the dictionary.

# Using pop to remove and get the value of 'banana'

my\_dict.pop('banana') #output: 3

# Now my\_dict contains only {'apple': 5, 'cherry': 8}

# return the value of the key 'banana' = 3

**8.What are sets in Python? Provide an example and explain their characteristics.**

**A.** In python, a set is a versatile data structure that allows to store elements in a single variable. set is created with curly braces {}.

examples,

s = {1,8,5,2,3,3,9,8,7}

print(s) # output: {1,8,5,2,3,9,7}

Characteristics of set are:

* unordered- items of set do not have a defined order.
* Unchangeable (Immutable): Once a set is created, you cannot change its items directly.
* No indexing: set does not support Indexing and slicing. because set is unordered.
* No duplicate: set automatically remove the duplicate elements.

**9.Explain the use of 'union' and 'intersection' operations with sets in Python. Provide examples.**

**A.**

'union' and 'intersection' operations with sets in Python:

Union:

* The union of two sets combines all unique elements from both sets into a new set.
* It includes elements that are present in either set (or both).
* The | operator is used for union.
* Examples:

A = {0, 2, 4, 6, 8}

B = {1, 2, 3, 4, 5}

union\_result = A | B

print("Union:", union\_result) # Output: {0, 1, 2, 3, 4, 5, 6, 8}

Intersection:

* The intersection of two sets contains elements that are common to both sets.
* It creates a new set with shared elements.
* The & operator is used for intersection.
* Example:

A = {0, 2, 4, 6, 8}

B = {1, 2, 3, 4, 5}

intersection\_result = A & B

print("Intersection:", intersection\_result) # Output: {2, 4}

**10.How can you remove an element from a set in Python? Provide a code example.**

**A.** In Python, you can remove an element from a set using either the remove() or the discard() method.

examples:

# Create a set

s = {1,2,3,4,5}

# Remove item from the set

s.remove(1)

print(s) #output: {2,3,4,5}

#using discard() method

s.discard(5)

print(s) #output: {2,3,4}

**11.What is the difference between a set and a frozen set in Python? Explain with examples.**

**A.** Differences between a set and a frozen set are:

* sets are mutable, meaning you can change their elements using built-in functions like add(), remove(), etc. But the frozensets are immutable, meaning we can't change their elements.

s = {1,2,3,4,"shar", True}

s.remove(4)

print(s) # output: {1,2,3,"shar", True}

* Sets cannot be used as dictionary keys or as elements of another set. But Frozensets can be used as dictionary keys or as elements of another set.

animals = frozenset(["cat", "dog", "lion"])

print("cat" in animals) # Output: True

print("elephant" in animals) # Output: False

**12.Explain the concept of nested dictionaries in Python with a suitable example.**

**A.** A nested dictionary is a dictionary inside another dictionary. It’s a way to organize data in a hierarchical structure, allowing you to group related information together. Each inner dictionary is associated with a unique key within the outer dictionary. the key points about nested dictionaries:

* A nested dictionary is created by placing dictionaries inside other dictionaries.
* It’s a collection of dictionaries within a single dictionary.

Example:

Let’s create a nested dictionary to represent information about people. Each person is identified by a unique ID, and their details (name, age, and sex) are stored in an inner dictionary:

people = {

1: {'name': 'John', 'age': '27', 'sex': 'Male'},

2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}

}

here, 1 and 2 are the unique IDs (keys) for each person.

and the inner dictionaries contain keys like 'name', 'age', and 'sex' with corresponding values.

**13.How would you merge two dictionaries in Python? Provide a code example.**

**A.** Merging two dictionaries can be done using various techniques,

* using update() method- It modifies the first dictionary in place and returns None.

dict1 = {'a': 10, 'b': 8}

dict2 = {'d': 6, 'c': 4}

dict1.update(dict2)

print(dict1) # Output: {'a': 10, 'b': 8, 'd': 6, 'c': 4}

* Using Dictionary Unpacking:

dict1 = {'a': 10, 'b': 8}

dict2 = {'d': 6, 'c': 4}

merged\_dict = {\*\*dict1, \*\*dict2}

print(merged\_dict) # Output: {'a': 10, 'b': 8, 'd': 6, 'c': 4}