**dictionary**

A **dictionary** in Python is a built-in data structure used to store collections of key-value pairs

1. **Key-Value Pairs**: Data is stored as pairs, where the key is used to retrieve the corresponding value.
2. **Mutable**: You can add, update, or delete key-value pairs.
3. **Dynamic**: It can grow or shrink in size as needed.
4. **Fast Lookups**: Dictionaries provide efficient retrieval of values based on their keys.

**Syntax:**

A dictionary is defined using curly braces {} or the dict() constructor.

#Example 1: Creating a Dictionary using Curly Braces:  
  
# dictionary  
a= {'a':123456 , 1:'abc'}  
print(a)  
#output:{'a': 123456, 1: 'abc'}  
  
# get()  
my = {'a': 1, 'b': 2, 'c': 3}  
print(my.get('b'))  
# Output: 2  
print(my.get('d'))  
# Output: None  
  
# update()  
k = {'a': 1, 'b': 2}  
k.update({'b': 10, 'c': 3})  
print(k)  
# Output: {'a': 1, 'b': 10, 'c': 3}  
# pop()  
  
archana = {'a': 1, 'b': 2, 'c': 3}  
value = archana.pop('b')  
print(value)  
# Output: 2  
print(archana)  
# Output: {'a': 1, 'c': 3}  
  
#popitem()  
h = {'a': 1, 'b': 2, 'c': 3}  
key\_value\_archana = h.popitem()  
print(key\_value\_archana)  
# Output: ('c', 3)  
print(h)  
# Output: {'a': 1, 'b': 2}  
  
#values()  
my\_dict = {'a': 1, 'b': 2, 'c': 3}  
values = my\_dict.values()  
print(list(values))  
# Output: [1, 2, 3]  
  
#keys()  
my\_dict = {'a': 1, 'b': 2, 'c': 3}  
keys = my\_dict.keys()  
print(list(keys))  
# Output: ['a', 'b', 'c']  
  
#items()  
my\_dict = {'a': 1, 'b': 2, 'c': 3}  
items = my\_dict.items()  
print(list(items))  
# Output: [('a', 1), ('b', 2), ('c', 3)]  
  
#SET:

A **set** in Python is a built-in data structure used to store unordered collections of unique elements. It is particularly useful for tasks involving membership testing, eliminating duplicate values, and performing mathematical set operations such as unions, intersections, and differences.

**Key Features of a Set:**

1. **Unordered**: The elements are stored without any specific order.
2. **Unique Elements**: A set does not allow duplicate values.
3. **Mutable**: You can add or remove elements from a set.
4. **Heterogeneous**: Sets can store elements of different data types, as long as they are immutable (e.g., numbers, strings, or tuples).

**Syntax:**

Sets are defined using curly braces {} or the built-in set() constructor.

#set example  
archana={1,2,8,4,2}  
k=set(archana)  
print(k)  
# output{1,2,8,4}  
# Add()  
  
my = {1, 2, 3}  
my.add(4)  
print(my)  
#output: {1, 2, 3, 4}  
# Clear()  
  
my1 = {1, 2, 3}  
my1.clear()  
print(my1)  
# output: set()  
  
# Copy()  
a = {1, 2, 3}  
new = a.copy()  
print(new) # output: {1, 2, 3}  
  
# difference  
set1 = {1, 2, 3}  
set2 = {3, 1, 5}  
print(set1.difference(set2)) # output: {2}  
  
# Discard()  
my1 = {1, 2, 3}  
my1.discard(2)  
my1.discard(4)  
print(my1)  
# Output: {1, 3}  
  
# Intersection()  
set1 = {1, 2, 3}  
set2 = {2, 3, 4}  
print(set1.intersection(set2))  
# 0utput: {2, 3}  
  
# isdisjoint  
set1 = {1, 2, 3}  
set2 = {4, 5, 6}  
print(set1.isdisjoint(set2))  
# Output: True  
  
# pop()  
hari = {1, 2, 3}  
element = archana.pop()  
print(element)  
# Output: 1  
  
print(archana)  
# Output: {2, 3}  
  
# remove()  
my\_set = {1, 2, 3}  
my\_set.remove(2)  
print(my\_set)  
# Output: {1, 3}  
  
# Symmetric difference()  
set1 = {1, 2, 3}  
set2 = {3, 4, 5}  
print(set1.symmetric\_difference(set2))  
# Output: {1, 2, 4, 5}  
# union  
set1 = {1, 2, 3}  
set2 = {3, 4, 5}  
print(set1.union(set2))  
# Output: {1, 2, 3, 4, 5}  
  
# Update()  
set1 = {1, 2, 3}  
set2 = {3,4, 5}  
set1.update(set2)  
print(set1)  
# Output: {1, 2, 3, 4, 5}  
# length  
b = {1, 2, 3, 4}  
print(len(b))  
# Output: 4  
# max()  
archana = {10, 20, 5, 7}  
print(max(archana))  
# Output: 20  
  
# sorted()  
j = {3, 1, 4, 2}  
print(sorted(j))  
# Output: [1, 2, 3, 4]