**1. List Creation:**

1. Create a list named age\_list with five integer elements. For eg., [24, 25, 26, 27, 28]
2. Create a list named name\_list with five string elements.

# LIST:

age\_list=[24,25,26,27,28]

name\_list=['murali','ramya','rajesh','suresh','ganesh']

**2. List Operations / Modifications:**

1. Append the string "Yazhini" to name\_list.

name\_list.append('yazhini')

print(name\_list)

1. Insert the element 30 at index 2 in age\_list.

age\_list.insert(2,30)

print(age\_list)

1. Remove the string "Yazhini" from name\_list.

name\_list.remove('yazhini')

print(name\_list)

#name\_list.pop(6)

#del name\_list(6)

Pop the last element from age\_list.

age\_list.pop(5)

print(age\_list)

1. Extend the age\_list with additional ages [29, 30, 26].

age\_list1= [29,30,26]

age\_list.extend(age\_list1)

print(age\_list)

1. Sort age\_list in descending order.

age\_list.sort(reverse=True)

print(age\_list)

1. Find Max age, Min age and sum of all ages from age\_list.

maximum =max(age\_list)

print(maximum)

minimum= min(age\_list)

print(minimum)

1. Make a copy of age\_list and store as age\_list\_copy

Sum = sum(age\_list)

print(Sum)

1. Make a copy of age\_list and store as age\_list\_copy

age\_copy=age.copy()

print(age\_copy)

1. Clear all elements from age\_list\_copy.

age\_copy.clear()

print(age\_copy)

**3. Accessing List Elements:**

Print the first element of name\_list.

print(name\_list[0])

print(name\_list[4])

print(name\_list[2:])

print(name\_list[::-1])

1. Print length of list\_dup.

list\_dup = [1,2,1,1,3,1,2,5,9,1]        # (duplicate)

print(list\_dup.count(1))

print(len(list\_dup))

**Output:**

**['murali', 'ramya', 'rajesh', 'suresh', 'ganesh', 'yazhini']**

**[24, 25, 30, 26, 27, 28]**

**['murali', 'ramya', 'rajesh', 'suresh', 'ganesh']**

**[24, 25, 30, 26, 27]**

**[24, 25, 30, 26, 27, 29, 30, 26]**

**[30, 30, 29, 27, 26, 26, 25, 24]**

**30**

**24**

**217**

**[30, 30, 29, 27, 26, 26, 25, 24]**

**[]**

**murali**

**ganesh**

**['rajesh', 'suresh', 'ganesh']**

**['ganesh', 'suresh', 'rajesh', 'ramya', 'murali']**

**5**

**10**

#DICTIONARY:

**1. Dictionary Creation:**

1. Create a dictionary named student\_marks that maps the names of five students to their marks (use scale of from 0 to 100).
2. Access and print the mark of a specific student, of your choice.

Student\_marks= {'sam': 95, 'ramesh': 95, 'suresh': 88, 'durga': 68, 'murali': 75}

#print(Student\_marks['suresh'])

print(Student\_marks.get('suresh'))

**2. Dictionary Modifications & Access:**

1. Add a new student "Janani" with a mark of 80 to the student\_marks dictionary.

Student\_marks['janani']=80

print(Student\_marks)

1. Update the mark of any one older student to 82.

Student\_marks['suresh']=82

print(Student\_marks)

1. Use the keys(), values(), and items() methods to print all keys, values, and key-value pairs in the student\_marks dictionary.

print(Student\_marks.keys())

print(Student\_marks.values())

print(Student\_marks.items())

output:

88

{'sam': 95, 'ramesh': 95, 'suresh': 88, 'durga': 68, 'murali': 75, 'janani': 80}

{'sam': 95, 'ramesh': 95, 'suresh': 82, 'durga': 68, 'murali': 75, 'janani': 80}

dict\_keys(['sam', 'ramesh', 'suresh', 'durga', 'murali', 'janani'])

dict\_values([95, 95, 82, 68, 75, 80])

dict\_items([('sam', 95), ('ramesh', 95), ('suresh', 82), ('durga', 68), ('murali', 75), ('janani', 80)])

[ ]

**1. Sets:**

1. Create a set called my\_set with following values:     
    ['a','e','i','o','u','a','a','i']

Analyse the output and provide explanation for the same.

my\_set = {'a','e','i','o','u','a','a','i'}

print(my\_set)

#print(list(set(my\_set)))

1. Attempt to change the value of my\_set[4] = 's'. If code throws an error, provide an explanation.

#my\_set[4]=0                             ERROR:TypeError: 'set' object does not support item assignment

print(my\_set)

1. Create two sets:   
    set1 with values: {1, 3, 5, 7, 9}   
    set2 with values: {2, 3, 5, 8, 10}

set1={1,3,5,7,9}

set2={2,3,5,8,10}

1. Compute the union of set1 and set2, and print the result.

print(set1.union(set2))

1. Perform an intersection operation between set1 and set2, and print the result.

print(set1.intersection(set2))

**2. Frozensets:**

1. Create a frozenset named immutable\_set containing the integers: **1**, **2**, **3**, **4**, **5**. Print the frozenset.

#frozensets:

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

immutable\_set=frozenset(set)

print(immutable\_set)

1. Try to add an element (e.g., **6**) to the immutable\_set frozenset. Observe and describe any exceptions that occur.

print(immutable\_set.add(6))

# We cannot add or remove elements in a frozen set.

Output:

{'e', 'o', 'i', 'u', 'a'}

{'e', 'o', 'i', 'u', 'a'}

{1, 2, 3, 5, 7, 8, 9, 10}

{3, 5}

frozenset({1, 2, 3, 4, 5})

---------------------------------------------------------------------------

AttributeError Traceback (most recent call last)

<ipython-input-4-d4885b7ab38b> in <cell line: 18>()

**16** immutable\_set=frozenset(set)

**17** print(immutable\_set)

---> 18 print(immutable\_set.add(6))

**19**

**20**

AttributeError: 'frozenset' object has no attribute 'add'