DICTIONARY :-

#@ Dictionary :-

#syntax:-

'''

'''

person = {

    'name':'Atmiya',

    'no':9081,

    'rollno':12,

    'city':'Rajkot'

}

print('Simple Dictionary -> ',person)

#we can also access like given below

print('\nusing key -> ',person['name'])

#we can also access using get() function

data = person.get('name')

print('\nusing get()-> ',data)

#we add new key and value pair.

person['branch']='MCA'

print('New Dictionary ->',person)

#we can also modifying it

person['branch']='BCA'

print('New modified Dictionary :',person)

#after del the key and value

del person['branch']

print('Deleted Dictionary Key and Value ->',person)

#Looping through a dictionary

#1) Looping all key-value pairs in a dictionary

print(person.items())

#using key and value

for key,value in person.items():

    print(f'Key={key}:Value={value}')

#access keys only

for key in person.keys():

    print(key)

#access values only

for value in person.values():

    print('\n',value)

**SET :-**

#@ Set :-

name = set(['ABC',12,'A',True])

print(name)

#getting size of Set

print("size of Set",len(name))

#adding element to a set using add()

name.add('ATMIYA')

print('After adding element ',name)

#removing element from set using remove()

name.remove('ATMIYA')

print('After removing element from the Set',name)

#To make it more convenient, the set has the discard() method that allows you to remove an element. And it doesn’t raise an error if the element is not in the list

name.discard('ATMIYA')

print('using discard() ',name)

#return an element from the set using pop()function

pop\_value = name.pop()

print('Returned Value :',pop\_value)

#removing all elements from the set using clear()

name.clear()

print(name)

#To make a set immutable, you use the built-in function called frozenset(). The frozenset() returns a new immutable set from an existing one

# skills = {'Problem solving', 'Software design', 'Python programming'}

# skills = frozenset(skills)

# print(skills)

#Looping through set elements

skills = {'Problem solving', 'Software design', 'Python programming'}

for value in skills:

    print(value)

#To access the index of the current element inside the loop, you can use the built-in enumerate() function

for key,value in enumerate(skills):

    print(f'{key},{value}')

#Union sets using union() method

number1= set([1,2,3,4,5])

number2 = set([6,7,8,9,10])

numberUnion = number1.union(number2)

print('using union() : ',numberUnion)

#using | operator.

numberPipe = number1 | number2

print('using | : ',numberPipe)

#The union() method vs. set union operator

rates = {1, 2, 3} #set

ranks = [2, 3, 4] #list

ratings = rates.union(ranks)

print(ratings)

#above code will easily executed.

rates = {1, 2, 3} #set

ranks = [2, 3, 4] #list

ratings = rates | ranks

#above code will generate the error.

#Using Python set intersection() method to intersect two or more sets

name1 = {'PYTHON','JAVA','C'}

name2= {'PYTHON','C++','.NET'}

names = name1.intersection(name2)

print('using intersection()',names)

#Using Python set intersection (&) operator to intersect two or more sets

name1 = {'PYTHON','JAVA','C'}

name2= {'PYTHON','C++','.NET'}

names = name1 & name2

print('using intersaction operator',names)

#Using Python Set difference() method to find the difference between sets

name1 = {'PYTHON','JAVA','C'}

name2= {'PYTHON','C++','.NET'}

names = name1.difference(name2)

print('using difference()',names)

#Using Python Set difference (-) operator method to find the difference between sets

name1 = {'PYTHON','JAVA','C'}

name2= {'PYTHON','C++','.NET'}

names = name1-name2

print('using difference operator ',names)

#Using the symmetric\_difference() method to find the symmetric difference of sets

name1 = {'PYTHON','JAVA','C'}

name2= {'PYTHON','C++','.NET'}

names = name1.symmetric\_difference(name2)

print('using symmetric difference :',names)

#Using the symmetric difference operator(^) to find the symmetric difference of sets

name1 = {'PYTHON','JAVA','C'}

name2= {'PYTHON','C++','.NET'}

names = name1 ^ name2

print('using symmetric difference operator :',names)

#Introduction to the Python issubset() method

name1 = {'PYTHON','JAVA','C'}

name2= {'PYTHON','C++','.NET'}

names = name2.issubset(name2)

print('using issubset() :',names)

#using subset operator

name1 = {'PYTHON','JAVA','C'}

name2= {'PYTHON','C++','.NET'}

namesSubset = name2 <= name2

print('using subset operator :',namesSubset)

namesProperSubset = name1 < name2

print('Proper subset :',namesProperSubset)

#Introduction to the Python issubset() method

name1 = {'PYTHON','JAVA','C'}

name2= {'PYTHON','C++','.NET'}

names = name2.issuperset(name2)

print('using issuperset() :',names)

#using subset operator

name1 = {'PYTHON','JAVA','C'}

name2= {'PYTHON','C++','.NET'}

namesSuperset = name2 >= name2

print('using superset operator :',namesSuperset)

namesProperSuperset = name1 > name2

print('Proper superset :',namesProperSuperset)

#Introduction to Python disjoint sets

odd\_numbers = {1, 3, 5}

even\_numbers = {2, 4, 6}

result = odd\_numbers.isdisjoint(even\_numbers)

print(result)

#above code will give True bcz there are no common elements.

letters = {'A', 'B', 'C'}

alphanumerics = {'A', 1, 2}

result = letters.isdisjoint(alphanumerics)

print(result)

#above code will give False bcz there are common elements.