**Set**

1. How do you create an empty set?

my\_set = set() # Using {} creates an empty dictionary, not a set.

1. How do you add an element to a set?

my\_set.add(element)

1. How do you remove an element from a set?

my\_set.remove(element) # Raises a KeyError if the element is not found.

# Alternatively, we can use discard() to avoid an error:

my\_set.discard(element)

1. How do you check if an element exists in a set?

|  |
| --- |
| my\_set= {1,2,3,4}  if 3 in my\_set:  print("3 is in the set")  output:  3 is in the set |

1. How do you find the length of a set?

length = len(my\_set)

1. How do you convert a list into a set?

my\_set = set(my\_list)

1. How do you clear all elements from a set?

my\_set.clear()

1. How do you check if one set is a subset of another set?

|  |
| --- |
| my\_set= {1,2,3,4}  A2={1,2,3,4,5,6,7}  is\_subset = my\_set.issubset(A2)  print(is\_subset)  output:  True |

1. How do you check if one set is a superset of another set?

is\_superset = my\_set.issuperset(A2)

1. How do you create a set from a string?

my\_set = set(my\_string)

1. How do you find the union of two sets?

union\_set = my\_set.union(A2)

1. How do you find the intersection of two sets?

intersection\_set = my\_set.intersection(A2)

1. How do you find the difference between two sets?

difference\_set = my\_set.difference(another\_set)

# or

difference\_set = my\_set - another\_set

1. How do you find the symmetric difference of two sets?

sym\_diff\_set = my\_set.symmetric\_difference(another\_set)

1. How do you remove duplicate elements from a list using a set?

unique\_list = list(set(my\_list))

1. How do you check if two sets are disjoint?

are\_disjoint = my\_set.isdisjoint(another\_set)

1. How do you copy a set?

copy\_set = my\_set.copy()

1. How do you iterate over the elements of a set?

|  |
| --- |
| my\_set= {1,2,3,4}  for element in my\_set:  print(element)  output:  1  2  3  4 |

1. How do you freeze a set to make it immutable?

frozen\_set = frozenset(my\_set)

1. How do you find common elements between multiple sets?

common\_elements = set1.intersection(set2, set3, ...) # Multiple sets

1. How do you find the Cartesian product of two sets?

|  |
| --- |
| set1 = {1, 2}  set2 = {'a', 'b'}  cartesian\_product = {(x, y) for x in set1 for y in set2}  print(cartesian\_product)  output:  {(2, 'a'), (1, 'b'), (1, 'a'), (2, 'b')} |

1. How do you check if a set is a subset of another set using a set operation?

is\_subset = my\_set <= another\_set

1. How would you efficiently merge several sets into one?

merged\_set = set1.union(set2, set3, ...)

1. How do you find the smallest and largest elements in a set?

smallest = min(my\_set)

largest = max(my\_set)

1. How do you remove multiple elements from a set at once?

|  |
| --- |
| my\_set = {1, 2, 3, 4, 5}  elements\_to\_remove = {2, 4}  for element in elements\_to\_remove:  my\_set.discard(element)  print(my\_set)  Output:  {1, 3, 5} |

1. How do you perform set operations (union, intersection, etc.) on more than two sets?

union\_set = set1.union(set2, set3, ...)

intersection\_set = set1.intersection(set2, set3, ...)

1. How do you find all unique subsets of a given set?

|  |
| --- |
| Using Bit manipulation technique:  def find\_subsets(s):  s = list(s) # Convert the set to a list to index its elements  subsets = []  for i in range(1 << len(s)): # Loop over all possible subsets (2^n possibilities)  subset = {s[j] for j in range(len(s)) if i & (1 << j)}  return subsets  my\_set = {1, 2, 3}  unique\_subsets = find\_subsets(my\_set)  for subset in unique\_subsets:  print(subset)  output:  set()  {1}  {2}  {1, 2}  {3}  {1, 3}  {2, 3}  {1, 2, 3} |

1. How do you implement custom set-like behavior in a class?
2. How do you check if two sets are equal?

are\_equal = my\_set == another\_set

1. How do you use sets to solve a problem involving unique combinations or duplicates?