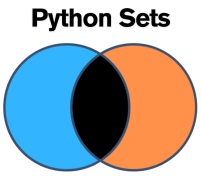
@learnwithbhawana

Python Set Exercises for Beginners – Basic to Advanced

- **Welcome to our Python Set Exercises!**
- **Sets are powerful for handling unique data, like IDs .**



- **These 10 questions, range from basic to advanced, perfect for beginners learning Python.**
- **Practice these to master sets for data analysis.**

Basic Set Exercises

Question: Write a Python program to create a set of unique IDs from a list:

[101, 102, 101, 103, 102].

Solution:

```
In [8]: ids = [101, 102, 101, 103, 102]
        unique_ids = set(ids)
        print("Unique IDs:", unique_ids)

        Unique IDs: {101, 102, 103}
```

Explanation: Sets automatically remove duplicates, ideal for unique identifiers in the data.

Question: Create a set of transaction types {"Stocks", "Bonds"} and add "MutualFunds".

STOCKS
BONDS

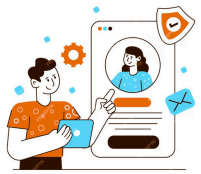
Solution:

```
In [9]: trans_types = {"Stocks", "Bonds"}
        trans_types.add("MutualFunds")
        print("Updated Transaction Types:", trans_types)

        Updated Transaction Types: {'MutualFunds', 'Bonds', 'Stocks'}
```

Explanation: The add() method inserts a single element.

Question: Remove User ID 102 from the set {101, 102, 103}.



User Id

Solution:

```
In [10]: userid = {101, 102, 103}
         userid.remove(102)
         print("Updated User ID:", userid)

         Updated User ID: {101, 103}
```

Explanation: remove() deletes an element; raises KeyError if not found.

Question: Find common music types between {"Pop", "Rock", "Classical"} and {"Pop", "Folk","Jazz"}.

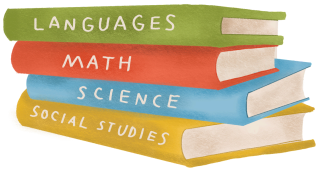
Solution:

```
In [11]: set1 = {"Pop", "Rock", "Classical"}
         set2 = {"Pop", "Folk","Jazz"}
         common = set1.intersection(set2)
         print("Common Music Types:", common)

         Common Music Types: {'Pop'}
```

Explanation: intersection() finds shared elements, useful for comparing two data sets.

Question: Combine Subjects : {"Science", "Maths"} and {"English", "Hindi"} into one set.



Solution:

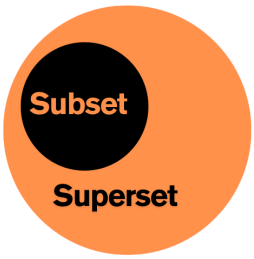
```
In [12]: set1 = {"Science", "Maths"}
         set2 = {"English", "Hindi"}
         combined = set1.union(set2)
         print("All Subjects:", combined)

         All Subjects: {'English', 'Science', 'Hindi', 'Maths'}
```

Explanation: union() merges sets, eliminating duplicates, ideal for consolidating data.

Advanced Set Exercises

Question: Check if one set of clients is a subset of another.



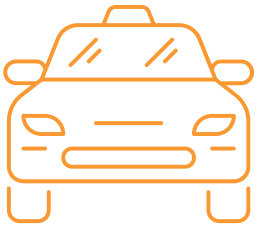
Solution:

```
In [13]: subset = {101, 102}
         superset = {101, 102, 103, 104}
         is_subset = subset.issubset(superset)
         print("Is Subset:", is_subset)

         Is Subset: True
```

Explanation: issubset() verifies if all elements are in another set.

Question: Find car brands in {"TATA", "Mahindra", "Maruti"} but not in {"Hyundai", "Mahindra", "TOYOTA"}.



Solution:

```
In [15]: set1 = {"TATA", "Mahindra", "Maruti"}
         set2 = {"Hyundai", "Mahindra", "TOYOTA"}
         unique = set1.difference(set2)
         print("Unique Car Brands:", unique)

         Unique Car Brands: {'TATA', 'Maruti'}
```

Explanation: difference() identifies exclusive elements, key for isolating specific financial data.

Question: Create a **frozenset** from [101, 102, 101] and try adding 103.

Solution:

```
In [16]: set1 = [101, 102, 101]
         frozen_set1 = frozenset(set1)
         print("Frozen Set:", frozen_set1)

         Frozen Set: frozenset({101, 102})
```

Explanation: **Frozensets** are immutable, useful for fixed account ids in high security systems like Banking.

Question: Write a program to check if "**listen**" and "**silent**" are **anagrams** using sets.

Solution:

```
In [17]: str1 = "listen"
         str2 = "silent"
         are_anagrams = set(str1) == set(str2)
         print("Are Anagrams:", are_anagrams)

         Are Anagrams: True
```

Explanation: Converting strings to sets compares unique characters, ignoring order, applicable for

data validation.

Question: From a list [1000, 2000, 1000, 3000], find unique combinations of two amounts using sets.

Unique Combinations

Solution:

```
In [18]: from itertools import combinations
         amounts = [1000, 2000, 1000, 3000]
         unique_amounts = set(amounts)
         combos = set(combinations(unique_amounts, 2))
         print("Unique Amount Combinations:", combos)

         Unique Amount Combinations: {(1000, 3000), (3000, 2000), (1000, 2000)}
```

Explanation: Sets with combinations ensure unique pairs, useful for scenario analysis.

Question: Find Maximum and Minimum Values in a set of numbers.

Solution:

```
In [20]: set1 = {5,9,5,10.2,15.5,15}
         print("Max of set1:",max(set1))
         print("Min of set1:",min(set1))

         Max of set1: 15.5
         Min of set1: 5
```

Explanation: Use the 'max()' function and the 'min()' function to get maximum and minimum value in 'set'.

Question: Write a Python program to iterate over a set and create a new set that contains double of each number.

Solution:

```
In [21]: original_set = {1, 2, 3, 4}
         doubled_set = {x * 2 for x in original_set}
         print(doubled_set)

         {8, 2, 4, 6}
```

Explanation:

We're going through each number in the set and creating a new set with each number multiplied by 2 using set comprehension.

Question: Write a Python program to show that a frozenset cannot be changed, unlike a regular set.

Change/Modify:
Regular Set ✅
Frozen Set ❌

Solution:

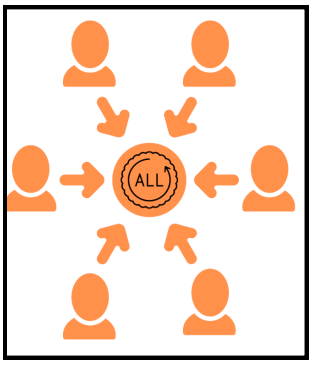
```
In [23]: normal_set = {1, 2, 3}
         normal_set.add(4)  # ✅

         frozen = frozenset([1, 2, 3])
         frozen.add(4)  # ❌

         ---------------------------------------------------------------------------
         AttributeError                            Traceback (most recent call last)
         Cell In[23], line 5
               2 normal_set.add(4)  # ✅
               4 frozen = frozenset([1, 2, 3])
         ----> 5 frozen.add(4)

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

Question: Find all clients or common clients across two advisors.



Solution:

```
In [28]: advisor1 = {101, 102, 103}
         advisor2 = {102, 103, 104}
         all_clients = advisor1 | advisor2
         common_clients = advisor1 & advisor2
         print(all_clients, common_clients)

         {101, 102, 103, 104} {102, 103}
```

Explanation : Union grabs everyone; intersection finds the VIP overlap!

Question:

Write a Python program to extract the IDs of all transactions with an amount **greater than ₹10,000** and store them in a **set**.

Iteration X

Solution:

```
In [30]: transactions = [{"id": 1, "amount": 5000}, {"id": 2, "amount": 15000}]
         high_value = {t["id"] for t in transactions if t["amount"] > 10000}
         print(high_value)

         {2}
```

Explanation : Use a **set comprehension** and a conditional if to filter only high-value transaction.

THE END