100 DAYS 100 PYTHON PROBLEMS

[Day 8 : Tuples and sets]

TUPLE :-

- A tuple is a collection data type in Python that is similar to a list but is immutable, meaning that once it is created, its elements cannot be changed or modified. Tuples are defined using parentheses '**()**'.

```
my_tuple = (1, 2, 3, 'hello', 3.14)
```

- Key characteristics of tuples:
  - **Immutable :** Once a tuple is created, you cannot add, remove, or modify elements.
  - **Ordered :** Elements in a tuple maintain their order
  - Can contain different data types.
- **Accessing Elements** : Elements in a tuple are accessed using indexing, similar to lists. Indexing starts from 0 for the first element.

```
first_element = my_tuple[0]
```

- **Slicing :**

```
sliced_tuple = my_tuple[1:3]  # Returns (2, 3)
```

- **Length of a Tuple:**

```
tuple_length = len(my_tuple)
```

- **Tuple Concatenation:**

```
new_tuple = my_tuple + ('world', 42)
```

100 DAYS 100 PYTHON PROBLEMS

[Day 8 : Tuples and sets]

- **Tuple Repetition:**

```
repeated_tuple = my_tuple * 2
```

- **Tuple Unpacking:**

```
a, b, c, *rest = my_tuple
```

- **Methods:** Tuples have limited methods compared to lists due to their immutability.
  count(x): Returns the number of occurrences of element x in the tuple.
  index(x): Returns the index of the first occurrence of element x.

```
count_3 = my_tuple.count(3)
index_hello = my_tuple.index('hello')
```

- **Nested Tuples:**

```
nested_tuple = ((1, 2), (3, 4))
```

- **Named Tuples:**

```
from collections import namedtuple
Point = namedtuple('Point', ['x', 'y'])
p = Point(1, 2)
```

- Use tuples when you have a collection of items that should remain constant throughout the program.
- Tuples are often used for returning multiple values from a function.

100 DAYS 100 PYTHON PROBLEMS

[Day 8 : Tuples and sets]

SET :-

- A set is an unordered collection of unique elements in Python. Sets are defined using curly braces '**{}**'.

```
my_set = {1, 2, 3, 3, 'hello', 3.14}
```

- Key characteristics of sets:
  - **Unordered :** Elements have no specific order.
  - **Unique elements :** Sets cannot contain duplicate elements. It removes them automatically.
  - **Mutable :** You can add/ remove elements in sets.
- **Accessing Elements:** Can't access using indexing.

```
for element in set1:
    print(element)
```

- **Set Operations:**
  - **UNION :** Returns a set containing all unique elements from both sets.

    ```
    union_set = set1 | set2
    ```
  - **INTERSECTION :** Returns a set containing common elements between two sets.

    ```
    intersection_set = set1 & set2
    ```
  - **DIFFERENCE :** Returns a set containing elements that are in the first set but not in the second.

    ```
    difference_set = set1 - set2
    ```

100 DAYS 100 PYTHON PROBLEMS

[Day 8 : Tuples and sets]

- **Modifying Sets:**

```
# Adding elements
set1.add(4)

# Removing elements
set1.remove(2)
```

- **Set Comprehensions:** Like lists and dictionaries, sets support comprehensions for concise creation**.**

```
squares = {x**2 for x in range(5)}
```

- **Frozen Sets:** Immutable version of sets is called 'frozenset'**.**

```
frozen_set = frozenset([1, 2, 3])
```

- **Subset and Superset:**

```
is_subset = set1.issubset(set2)
is_superset = set1.issuperset(set2)
```

100 DAYS 100 PYTHON PROBLEMS

[Day 8 : Tuples and sets]

PRACTICE QUESTIONS :-

❖ TUPLE

1. Create a tuple with elements 5, 'apple', and 2.5.
2. Access the third element of the tuple created in question 1.
3. Concatenate two tuples: (1, 2, 3) and (4, 5, 6).
4. Create a tuple with five elements, and then unpack it into five variables.
5. Check if the element 'banana' exists in the tuple (1, 'apple', 3.14, 'banana')

❖ SET

1. Create two sets: {1, 2, 3, 4} and {3, 4, 5, 6}.
2. Find the union of the sets created in question 1.
3. Find the intersection of the sets created in question 1.
4. Add the element 'orange' to the set {'apple', 'banana'}.
5. Remove the element 3 from the set {1, 2, 3, 4, 5}.

❖ MIXED

1. Create a tuple with elements 'apple', 3, and a set with elements 1, 2, 3.
2. Convert the tuple (7, 8, 9) to a set.
3. Create a set with the common elements between the tuple (1, 2, 3) and the set {3, 4, 5}.