INDIAN DATA CLUB

indiandataclub.com |

DAY 3

PYTHON CHALLENGE

TOPIC

- Lists (slicing, methods), tuples (immutability), dictionaries (key-value pairs), Sets

CHALLENGE

- Create an inventory system tracking items and quantities with a dictionary

#30DaysOfPython #IDC30DaysChallenge

Email
hchethankumar17@gmail.com
Bengaluru, Karnataka

30-Days Python Challenge

Indexing in Python

DAY -3

```
# Indexing :Indexing allows you to access individual elements in a list.
# EX:
indexing_list = [1, 2, 3, 4, 5]
#Types of indexing
# There are two types of indexing in Python:
#1. Positive indexing
"""Positive indexing starts from 0 and goes to the end of the list.
For example, in the list [1, 2, 3, 4, 5], the first element (1) is at index 0,
the second element (2) is at index 1, and so on."""
#  2.Negative indexing
"""Negative indexing starts from -1 and goes to the beginning of the list.
For example, in the list [1, 2, 3, 4, 5], the last element (5) is at index -1,
the second last element (4) is at index -2, and so on."""
```

```
indexing_list = [1, 2, 3, 4, 5]
# Accessing elements using positive indexing
print(indexing_list[0])  # Returns 1
print(indexing_list[1])  # Returns 2
print(indexing_list[2])  # Returns 3
```

Indexing can be done in:String,list,Tuple

Email
hchethankumar17@gmail.com
Bengaluru, Karnataka

#IDC30DaysChallenge
#30DaysOfPython

30-Days Python Challenge

LIST DATA TYPE

DAY -3

- The data or data type which has more than one value, then it is called as multi value or collection data type.
- EX: "STRING,LIST, TUPLE, SET, DICTIONARY"

```
# 1. List
""" A list is a collection of items which is ordered and changeable.
Which is Enclosed in [], Allows duplicate members."""
my_list = [1, 2, 3, 4, 5]
```

```
fruits = ["apple", "banana", "orange"]
# Print the list of fruits
print(fruits)
```

```
['apple', 'banana', 'orange']
```

Email
hchethankumar17@gmail.com
Bengaluru, Karnataka

#IDC30DaysChallenge
#30DaysOfPython

30-Days Python Challenge

SLICING

DAY -3

```python
# SLICING
"""Slicing is a way to access a subset of elements in a sequence.
Syntax:
var/val[start_index:end_index(+or-)1:step]
slicing can be done in lists, strings, and tuples."""
```

```python
# Example:
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(my_list[1:5])  # Output: [2, 3, 4, 5]
# Slicing can also be used with strings
my_string = "Hello, World!"
print(my_string[0:5])  # Output: Hello
# Slicing can be used to reverse a list or string
print(my_list[::-1])  # Output: [9, 8, 7, 6, 5, 4, 3, 2, 1]
# Slicing can also be used to get every second element
print(my_list[::2])  # Output: [1, 3, 5, 7, 9]
```

Email
hchethankumar17@gmail.com
Bengaluru, Karnataka

#IDC30DaysChallenge
#30DaysOfPython

30-Days Python Challenge

List methods with examples

DAY -3

```python
# 1. append()
my_list=["apple", "banana", "cherry"]
my_list.append("orange")
print("1. append():", my_list)  # Output: ['apple', 'banana', 'cherry', 'orange']
```

```python
# 2. extend()
my_list.extend(["kiwi", "mango"])
print("2. extend():", my_list)  # Output: ['apple', 'banana', 'cherry', 'orange', 'kiwi', 'mango']
```

```python
# 3. insert()
my_list.insert(1, "orange")  # Insert "orange" at index 1
print("After insert:", my_list)
# Output: After insert: ['apple', 'orange', 'banana', 'cherry']
```

```python
# 4. remove()
my_list.remove("banana")  # Remove the first occurrence of "banana"
print("After remove:", my_list)
# Output: After remove: ['apple', 'orange', 'cherry']
```

```python
# 5. pop()
my_list.pop(1)  # Remove and return the item at index 1
print("After pop:", my_list)
# Output: After pop: ['apple', 'cherry']
```

Email
hchethankumar17@gmail.com
Bengaluru, Karnataka

#IDC30DaysChallenge
#30DaysOfPython

30-Days Python Challenge

List methods with examples

DAY -3

```python
# 6. clear()
my_list.clear()  # Remove all items from the list
print("After clear:", my_list)
# Output: After clear: []
```

```python
# 7. index()
my_list = ["apple", "banana", "cherry"]
index_of_banana = my_list.index("banana")  # Get the index of "banana"
print("Index of banana:", index_of_banana)
# Output: Index of banana: 1
```

```python
# 8. count()
count_of_apple = my_list.count("apple")  # Count occurrences of "apple"
print("Count of apple:", count_of_apple)
```

```python
my_list = ["banana", "apple", "cherry"]
my_list.sort()  # Sort the list in ascending order
print("After sort:", my_list)
# Output: After sort: ['apple', 'banana', 'cherry']
```

Email
hchethankumar17@gmail.com
Bengaluru, Karnataka

#IDC30DaysChallenge
#30DaysOfPython

30-Days Python Challenge

Tuple Data Type

DAY -3

```
"Tuple is collection data type which is used to Store multiple items in a single variable."
# Tuple is ordered, unchangeable, and allows duplicate values.
# Tuple is defined by using parentheses ().
# Example of tuple
my_tuple = (1, 2, 3, 4, 5)
# Tuple can also contain different data types
my_tuple2 = (1, "Hello", 3.14, True)
# Tuple can be created using the tuple() constructor
my_tuple3 = tuple([1, 2, 3, 4, 5])
```

Tuple are immutable, meaning you cannot change their values after creation.
Because of immutablility tuples are used in secure data transmission

```
my_tuple = (1, 2, 3)
# Attempting to modify an element will raise an error
# my_tuple[0] = 10 # This will cause an error

# However, you can create a new tuple with the desired changes
new_tuple = my_tuple + (4,) # This creates a new tuple (1, 2, 3, 4)
print(new_tuple) # Output: (1, 2, 3, 4)
```

Email
hchethankumar17@gmail.com
Bengaluru, Karnataka

#IDC30DaysChallenge
#30DaysOfPython

30-Days Python Challenge

SET DATA TYPE

DAY -3

```
# set data type
"""set is collection datatype that is unordered,
unindexed,and does not allow duplicate elements
enclosed in {we can only pass mutable values in side a set}.
set is mutable, meaning you can add or remove elements from it."""
set1 = {1, 2, 3, 4, 5,(143, 123), "hello", "world"}
```


```
# set data type
"""set is collection datatype that is unordered,
unindexed,and does not allow duplicate elements
enclosed in {we can only pass mutable values in side a set}.
set is mutable, meaning you can add or remove elements from it."""
set1 = {1, 2, 3, 4, 5,(143, 123), "hello", "world"}
```


set is mutable collection data type because of inbuilt functions

Email
hchethankumar17@gmail.com
Bengaluru, Karnataka

#IDC30DaysChallenge
#30DaysOfPython

30-Days Python Challenge

Dictionary data type

DAY -3

```python
# Dictionary is a collection of key-value pairs.
"""var = {key1: value1, key2: value2, ...}
where key is immutable and value can be mutable or immutable.
A dictionary is unordered, changeable, and indexed.
A dictionary can be created using curly braces {} or the dict() constructor.
"""
# Example of a dictionary
var = {
    "name": "chethan",
    "age": 23,
    "city": "Bangalore"
    }
# o/p
print(var) # {'name': 'chethan', 'age': 23, 'city': 'Bangalore'}
```

if we store duplicate keys in a dictionary, previous value will be overwritten by the last value.

Email
hchethankumar17@gmail.com
Bengaluru, Karnataka

#IDC30DaysChallenge
#30DaysOfPython