**Data Types in Python**

**1. Difference Between Mutable and Immutable Data Types in Python**

* **Mutable Data Types**:
  + Mutable data types can be changed after they are created. This means that you can alter, add, or remove elements within these data structures without creating a new object.
  + **Examples**:
    - **List**: A list can have elements added, removed, or changed.

my\_list = [1, 2, 3]

my\_list [0] = 10 # Now my\_list is [10, 2, 3]

my\_list. append (4) # Now my\_list is [10, 2, 3, 4]

* + - **Dictionary**: You can add, remove, or change key-value pairs.

my\_dict = {'a': 1, 'b': 2}

my\_dict['a'] = 3 # Now my\_dict is {'a': 3, 'b': 2}

my\_dict['c'] = 4 # Now my\_dict is {'a': 3, 'b': 2, 'c': 4}

* **Immutable Data Types**:
  + Immutable data types cannot be changed after they are created. Any modification will result in a new object being created.
  + **Examples**:
    - **Tuple**: Once a tuple is created, you cannot modify its contents.

my\_tuple = (1, 2, 3)

# My\_tuple [0] = 10 # This would raise an error

* + - **String**: Strings are immutable, meaning any operation on a string that alters its content will result in a new string.

my\_str = "hello"

my\_str = my\_str + " world" # Now my\_str is "hello world", a new string

**2. Type Conversion Between int, float, and str**

* **int ()**: Converts a number or a string to an integer. Non-numeric strings will cause an error.

int ('123') # Output: 123

int (12.34) # Output: 12

* **float ()**: Converts a number or a string to a floating-point number.

float ('12.34’) # Output: 12.34

float (12) # Output: 12.0

* **str ()**: Converts any value to a string.

Str (123) # Output: '123'

str (12.34) # Output: '12.34'

**3. Comparison of List and Tuple in Python**

* **List**:
  + Mutable: Elements can be changed.
  + Syntax: Created using square brackets [].
  + Example:

my\_list = [1, 2, 3]

my\_list. Append (4) # Now my\_list is [1, 2, 3, 4]

* **Tuple**:
  + Immutable: Elements cannot be changed after creation.
  + Syntax: Created using parentheses ().
  + Example:

my\_tuple = (1, 2, 3)

# My\_tuple [0] = 10# This would raise an error

* **When to Use**:
  + **List**: When you need a sequence of elements that can be modified (e.g., adding, removing, or changing elements).
  + **Tuple**: When you need a sequence of elements that should not change throughout the program, or when you want to ensure the data's integrity.

**4. Python’s Set Data Structure**

* **Uniqueness**: Python’s set ensures that all elements are unique. If you attempt to add a duplicate element, it will be ignored.

my\_set = {1, 2, 3}

my\_set.add (2) # my\_set is still {1, 2, 3}

* **Operations on Sets**:
  + **Union**: Combines elements from two sets.

set1 = {1, 2, 3}

set2 = {3, 4, 5}

set1.union(set2) # Output: {1, 2, 3, 4, 5}

* + **Intersection**: Returns elements common to both sets.

set1.intersection(set2) # Output: {3}

* + **Difference**: Returns elements in one set but not in the other.

set1.difference(set2) # Output: {1, 2}

**5. Boolean Values in Conditional Statements and Loops**

* **Boolean Values**: True and False are used to control the flow of a program.
* **Conditional Statements**:
  + If a condition evaluates to True, the associated block of code is executed.

if condition:

# Execute this block if condition is True

* **Loops**:
  + Loops like while and for rely on Boolean values to determine whether to continue or break the loop.

while condition:

# Loop as long as condition is True

**6. Common Operations on Strings in Python**

* **Concatenation**: Combining strings using +.

"Hello" + " " + "World" # Output: "Hello World"

* **Slicing**: Extracting a part of a string.

my\_str = "Hello World"

my\_str [0:5] # Output: "Hello"

* **Upper/Lower Case**: Changing the case of the string.

"hello”. Upper () # Output: "HELLO"

"HELLO”. Lower () # Output: "hello"

* **Replace**: Replacing parts of a string.

"Hello World”. replace ("World", "Python") # Output: "Hello Python"

**7. Concept of Dictionary Keys in Python**

* **Keys Must Be Immutable**: Dictionary keys must be immutable types like strings, numbers, or tuples because they are hashed to access values quickly.
* **Why Immutable**: If keys were mutable, their hash values could change, leading to inconsistencies and making the dictionary unreliable.

**8. Arithmetic Operations Between int and float in Python**

* When you perform an arithmetic operation between an int and a float, the result is of type float.

result = 5 + 2.0 # Output: 7.0 (float)

**Practical Examples**

**9. Convert a List of Strings to a Dictionary**

string\_list = ["apple", "banana", "cherry"]

string\_dict = {s: len(s) for s in string\_list}

print(string\_dict) # Output: {'apple': 5, 'banana': 6, 'cherry': 6}

**10. Remove Duplicates from a List of Integers**

int\_list = [4, 5, 2, 4, 3, 2, 1]

unique\_sorted\_list = sorted(list(set(int\_list)))

print(unique\_sorted\_list) # Output: [1, 2, 3, 4, 5]

**11. Populate a Dictionary with User Input**

user\_dict = {}

for \_ in range (5):

key = input ("Enter key: ")

value = input ("Enter value: ")

user\_dict[key] = value

print(user\_dict)

**12. Convert a List of Tuples to a Dictionary**

name\_age\_list = [("Alice", 30), ("Bob", 25), ("Charlie", 35)]

name\_age\_dict = dict(name\_age\_list)

print(name\_age\_dict) # Output: {'Alice': 30, 'Bob': 25, 'Charlie': 35}

**13. Merge Two Dictionaries**

dict1 = {'a': 1, 'b': 2}

dict2 = {'b': 3, 'c': 4}

merged\_dict = {\*\*dict1, \*\*dict2} # dict2 overwrites dict1

print(merged\_dict) # Output: {'a': 1, 'b': 3, 'c': 4}

**14. Convert a List to a Tuple and Vice Versa**

my\_list = [1, 2, 3]

my\_tuple = tuple(my\_list)

print(my\_tuple) # Output: (1, 2, 3)

my\_list\_again = list(my\_tuple)

print(my\_list\_again) # Output: [1, 2, 3]

**15. Slicing in Python**

* **List Slicing**:

my\_list = [1, 2, 3, 4, 5]

print (my\_list [1:4]) # Output: [2, 3, 4]

* **String Slicing**:

my\_str = "Hello World"

print (my\_str [0:5]) # Output: "Hello"

**16. Examples of List, Tuple, Dictionary, and Set**

* **List**:

my\_list = [1, 2, 3]

* **Tuple**:

my\_tuple = (1, 2, 3)

* **Dictionary**:

my\_dict = {'a': 1, 'b': 2, 'c': 3}

* **Set**:

my\_set = {1, 2, 3, 2} # Duplicate '2' is ignored, my\_set = {1, 2, 3}