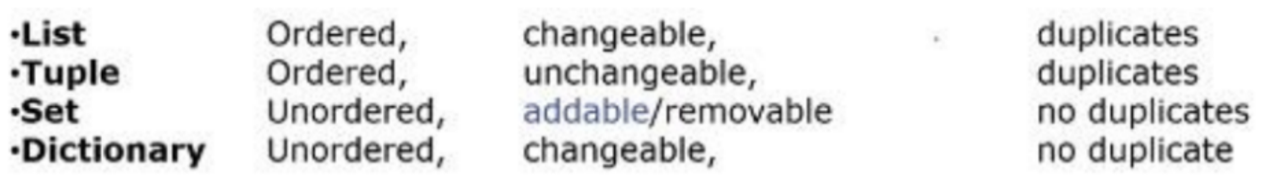
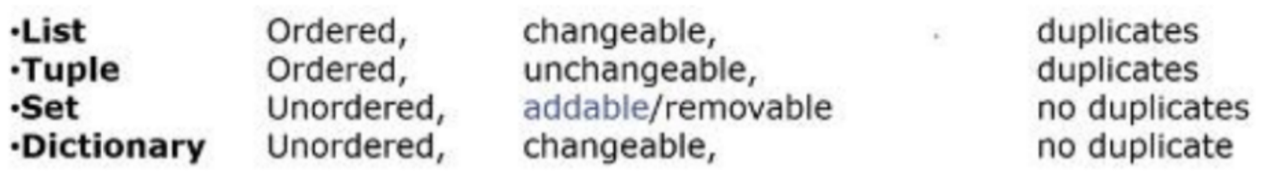
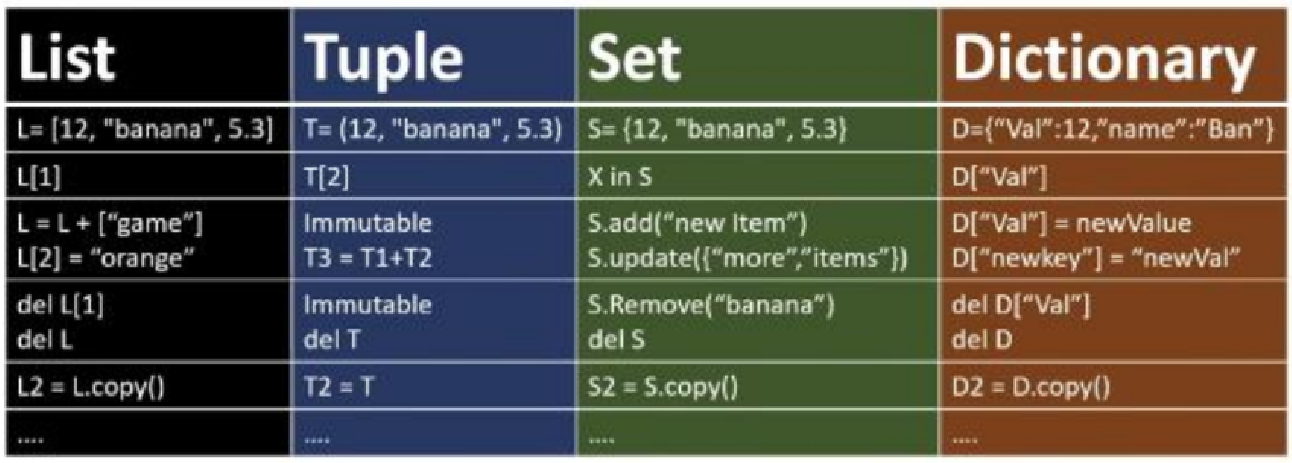


·**List**	Ordered,	changeable,	duplicates
·**Tuple**	Ordered,	unchangeable,	duplicates
·**Set**	Unordered,	addable/removable	no duplicates
·**Dictionary**	Unordered,	changeable,	no duplicate



·**List**	Ordered,	changeable,	duplicates
·**Tuple**	Ordered,	unchangeable,	duplicates
·**Set**	Unordered,	addable/removable	no duplicates
·**Dictionary**	Unordered,	changeable,	no duplicate



List	Tuple	Set	Dictionary
L= [12, "banana", 5.3]	T= (12, "banana", 5.3)	S= {12, "banana", 5.3}	D={"Val":12,"name":"Ban"}
L[1]	T[2]	X in S	D["Val"]

L = L + ["game"]
L[2] = "orange"
	Immutable
	T3 = T1+T2
		S.add("new Item")
		S.update({"more","items"})
			D["Val"] = newValue
			D["newkey"] = "newVal"

del L[1] del L	Immutable del T	S.Remove("banana") del S	del D["Val"] del D
L2 = L.copy()	T2 = T	S2 = S.copy()	D2 = D.copy()
....	....	....	....

List in Python

In [12]:

```
List1
```

Out[12]:

```
[10, 20, 30, 'apple', 40, 'mango']
```

In [60]:

```
List1=[10,20,30,"apple",40,"mango"]
```

In [61]:

```
Numbers = [1,3,7,5,2,4,6]
```

In [62]:

```
Animals = ["cat","dog","ant","lion"]
```

In [14]:

```
print(List1[0])
print(List1[1])
```

```
10
20
```

Slicing the list:

In [32]:

```
print(List1[2:6]) #slicing from the 2nd index to 6th index
print(List1[:6]) #from the starting to given index
print(List1[1:]) #from first index to last
print(List1[:]) #complete list
```

```
[30, 'apple', 40, 'mango']
[10, 20, 30, 'apple', 40, 'mango']
[20, 30, 'apple', 40, 'mango']
[10, 20, 30, 'apple', 40, 'mango']
```

Modifying the List:

In [46]:

```
List1[2]="banana" #Change the element's data
print(List1)

#Add an element to the end
List1.append("50")
print(List1)

#Add an element to any index
List1.insert(1,"grapes")
print(List1)

#Removing an Element by value
List1.remove("50")
print(List1)


#Removing an Element by index and get its value
print(List1.pop(2))
print(List1)

#Extend the list by adding elements
List1.extend(["pear","lemon"])
print(List1)
```

```
[10, 20, 'banana', 'apple', 40, 'mango']
[10, 20, 'banana', 'apple', 40, 'mango', '50']
[10, 'grapes', 20, 'banana', 'apple', 40, 'mango', '50']
[10, 'grapes', 20, 'banana', 'apple', 40, 'mango']
20
[10, 'grapes', 'banana', 'apple', 40, 'mango']
[10, 'grapes', 'banana', 'apple', 40, 'mango', 'pear', 'lemon']
```

Operation In List

In [63]:

```
#Clear all items from the List
Animals.clear()
print(Animals)

#count the given paramter from the List
List1.count("apple")

#sort the list
print(Numbers)
```

```
Numbers.sort()
print(Numbers)

#reverse the List
Numbers.reverse()

newNums = Numbers.copy()
print(newNums)

#check if an item exist
print(3 in Numbers)

#concatenate List
secondList = List1 + ["item1","item2"]

print(secondList)
```

```
[]
[1, 3, 7, 5, 2, 4, 6]
[1, 2, 3, 4, 5, 6, 7]
[7, 6, 5, 4, 3, 2, 1]
True
[10, 20, 30, 'apple', 40, 'mango', 'item1', 'item2']
```

Tuple In Python

In [66]:

```
#Creating Tuple:

# 1. Directly with parentheses (most common)
tuple1 = (1, 2, 3)

# 2. For a single-item tuple, you MUST include a trailing comma
single_item_tuple = ("hello",)  # This is a tuple
not_a_tuple = ("hello")         # This is just a string

# 3. Using the tuple() constructor
tuple2 = tuple([4, 5, 6])  # Convert from a list
tuple3 = tuple("abc")      # Creates ('a', 'b', 'c')

# 4. Without parentheses (tuple packing)
my_tuple = 1, 2, "three"
print(my_tuple)  # Output: (1, 2, 'three')

# 5. An empty tuple
empty_tuple = ()
```

```
(1, 2, 'three')
```

In [67]:

```
#Accessing Elements (Indexing & Slicing)

fruits = ("apple", "banana", "cherry", "date")

print(fruits[0])   # Output: apple (first element)
print(fruits[2])   # Output: cherry (third element)
print(fruits[-1])  # Output: date (last element)
```

```
numbers = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

print(numbers[2:5])    # Output: (2, 3, 4)
print(numbers[:4])     # Output: (0, 1, 2, 3)
print(numbers[::2])    # Output: (0, 2, 4, 6, 8)
print(numbers[::-1])   # Output: (9, 8, 7, 6, 5, 4, 3, 2, 1, 0)
```

```
apple
cherry
date
(2, 3, 4)
(0, 1, 2, 3)
(0, 2, 4, 6, 8)
(9, 8, 7, 6, 5, 4, 3, 2, 1, 0)
```

In [68]:

```
#Tuple Unpacking

# Basic unpacking
fruits = ("apple", "banana", "cherry")
fruit1, fruit2, fruit3 = fruits

print(fruit1)  # Output: apple
print(fruit2)  # Output: banana
print(fruit3)  # Output: cherry

# Useful for swapping variables (no temp variable needed!)
a = 5
b = 10
a, b = b, a  # This works because (b, a) is packed into a tuple, then unpacked
print(a)     # Output: 10
print(b)     # Output: 5

# Using asterisk * to capture multiple items
numbers = (1, 2, 3, 4, 5)
first, *middle, last = numbers

print(first)   # Output: 1
print(middle)  # Output: [2, 3, 4] (Note: this becomes a list)
print(last)    # Output: 5
```

```
apple
banana
cherry
10
5
1
[2, 3, 4]
5
```

Set In Python

In [71]:

```
#Creating Set

# 1. Directly with curly braces (for non-empty sets)
set1 = {1, 2, 3}
```

```
# 2. Using the set() constructor (most common for empty sets and from other iterables)
empty_set = set()   # Correct way to make an empty set
# empty_set = {}    # INCORRECT! This creates an empty dictionary.

set_from_list = set([1, 2, 2, 3, 4])  # Converts list to set, duplicates removed
print(set_from_list)  # Output: {1, 2, 3, 4} (order may vary)

set_from_string = set("hello")         # Creates {'h', 'e', 'l', 'o'}
print(set_from_string)                 # Output: {'o', 'e', 'h', 'l'} (notice only one '
```

```
{1, 2, 3, 4}
{'o', 'e', 'h', 'l'}
```

In [72]:

```
#Properties

# Duplicates are automatically removed
numbers = {1, 2, 2, 3, 3, 3, 4}
print(numbers)  # Output: {1, 2, 3, 4} (order may vary)

# You cannot access elements by index
fruits = {"apple", "banana", "cherry"}
# print(fruits[0]) # This will raise a TypeError

# You can check for membership very efficiently (this is a key strength of sets)
print("banana" in fruits)  # Output: True (Very fast operation)
print("mango" in fruits)   # Output: False
```

```
{1, 2, 3, 4}
True
False
```

In [73]:

```
#Modifying Sets

my_set = {1, 2, 3}

# 1. Add a single element
my_set.add(4)
print(my_set)  # Output: {1, 2, 3, 4}

# 2. Add multiple elements from an iterable (list, tuple, another set)
my_set.update([5, 6, 7])
print(my_set)  # Output: {1, 2, 3, 4, 5, 6, 7}

# 3. Remove an element (raises error if the element is not found)
my_set.remove(3)
print(my_set)  # Output: {1, 2, 4, 5, 6, 7}

# 4. Discard an element (does NOT raise an error if the element is not found - safer)
my_set.discard(10) # No error, even though 10 isn't in the set
my_set.discard(2)

# 5. Remove and return an arbitrary element (since sets are unordered)
popped_item = my_set.pop()
print(f"Popped: {popped_item}, Set is now: {my_set}")

# 6. Remove all elements
my_set.clear()
print(my_set)  # Output: set()
```

```
{1, 2, 3, 4}
{1, 2, 3, 4, 5, 6, 7}
{1, 2, 4, 5, 6, 7}
Popped: 1, Set is now: {4, 5, 6, 7}
set()
```

In [74]:

```
#Set Operations

A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

print(A | B)           # Output: {1, 2, 3, 4, 5, 6, 7, 8}
print(A.union(B))      # Output: {1, 2, 3, 4, 5, 6, 7, 8}

print(A & B)                # Output: {4, 5}
print(A.intersection(B))    # Output: {4, 5}


print(A - B)             # Output: {1, 2, 3} (in A, not in B)
print(B - A)             # Output: {6, 7, 8} (in B, not in A)
print(A.difference(B))   # Output: {1, 2, 3}


print(A ^ B)                         # Output: {1, 2, 3, 6, 7, 8}
print(A.symmetric_difference(B))     # Output: {1, 2, 3, 6, 7, 8}

X = {1, 2}
Y = {1, 2, 3}

print(X.issubset(Y))     # Output: True (X is a subset of Y)
print(Y.issuperset(X))   # Output: True (Y is a superset of X)

# Disjoint check: Do the sets have no elements in common?
print(X.isdisjoint({9, 10})) # Output: True
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
{4, 5}
{4, 5}
{1, 2, 3}
{8, 6, 7}
{1, 2, 3}
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
True
True
True
```

Dictionaries In Python

In [86]:

```
# A simple dictionary
dict1 = {
    "name": "Waleed",
    "age": 20,
    "city": "FSD"
}

# A dictionary with different key types
mixed_keys = {
    "name": "Imran",        # String key
    42: "Freshie",     # Integer key
    (1, 2): "tuple key"  # Tuple key (immutable)
}

# A dictionary with a list as a value
complex_value = {
    "scores": [85, 92, 78],
    "attributes": {"height": 170, "weight": 65}
}
```

In [77]:

```
dict1
```

Out[77]:

```
{'name': 'Waleed', 'age': 20, 'city': 'New York'}
```

In [78]:

```
mixed_keys
```

Out[78]:

```
{'name': 'Imran', 42: 'Freshie', (1, 2): 'tuple key'}
```

In [82]:

```
mixed_keys[42]
```

Out[82]:

```
'Freshie'
```

In [92]:

```
print(dict1["city"])
print(dict1.get("name"))
```

```
FSD
Waleed
```

In [94]:

```
#Modifying Dictionary

person = {"name": "Jamshed", "age": 25}

# 1. Change an existing value
person["age"] = 26
print(person)

# 2. Add a new key-value pair
person["city"] = "Karachi"
print(person)

# 3. Update multiple values at once using .update()
person.update({"age": 27, "country": "Pakistan"})
print(person)
```

```
{'name': 'Jamshed', 'age': 26}
{'name': 'Jamshed', 'age': 26, 'city': 'Karachi'}
{'name': 'Jamshed', 'age': 27, 'city': 'Karachi', 'country': 'Pakistan'}
```

In [97]:

```python
#Operation in Dictionary

person = {"name": "Jamshed", "age": 25, "city": "Karachi"}

# Get all keys
print(list(person.keys()))

# Get all values
print(list(person.values()))

# Loop through key-value pairs
for key, value in person.items():
    print(f"{key}: {value}")

# Remove a specific key
removed_city = person.pop("city")
print(removed_city)
print(person)
```

```
['name', 'age', 'city']
['Jamshed', 25, 'Karachi']
name: Jamshed
age: 25
city: Karachi
Karachi
{'name': 'Jamshed', 'age': 25}
```

In [106]:

```python
#Nested Dictionary

# A nested dictionary
users = {
    "user1": {
        "name": "Ali",
        "age": 30,
        "contacts": {"email": "ali@example.com", "phone": "123-4567"}
    },
    "user2": {
        "name": "Ahmed",
        "age": 25,
        "contacts": {"email": "ahmed@example.com", "phone": "987-6543"}
    }
}

# Accessing nested values
print(users["user1"]["name"])
print(users["user2"]["contacts"]["email"])   #

# Modifying nested values
users["user1"]["age"] = 31
users["user1"]["contacts"]["phone"] = "555-1234"
print(users["user1"]["contacts"])
print(users["user1"]["contacts"]["phone"])
```

```
Ali
ahmed@example.com
{'email': 'ali@example.com', 'phone': '555-1234'}
555-1234
```

In [ ]: