# **High Impact Skills Development Program** **AI & Data Science**

**Data Structure in Python**

**1. Lists:** Ordered collections of items that can be of any data type, including strings, integers, floats, and other lists.

**Working on Lists:**

1. append() - adds an element to the end of list

my\_list = [1, 2, 3]

my\_list.append(4)

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

2. insert() - adds an element at a specific position

my\_list = [1, 2, 3]

my\_list.insert(1, 4)

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

3. remove() - removes the first occurrence of an element

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

my\_list.remove(2)

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

4. pop() - removes and returns an element at a specific position

my\_list = [1, 2, 3]

popped\_element = my\_list.pop(1)

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

print(popped\_element) # Output: 2

5. sort() - sorts the list in ascending order

my\_list = [3, 2, 1]

my\_list.sort()

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

6. reverse() - reverses the list

my\_list = [1, 2, 3]

my\_list.reverse()

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

7. index() - returns the index of the first occurrence of an element

my\_list = [1, 2, 3]

index = my\_list.index(2)

print(index) # Output: 1

8. count() - returns the number of occurrences of an element

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

count = my\_list.count(2)

print(count) # Output: 2

9. extend() - adds multiple elements to the end

my\_list = [1, 2, 3]

my\_list.extend([4, 5, 6])

print(my\_list) # Output: [1, 2, 3, 4, 5, 6]

10. clear() - removes all elements

my\_list = [1, 2, 3]

my\_list.clear()

print(my\_list) # Output: []

11. del

1. Delete a single element by index:

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

del my\_list[2] # Delete the element at index 2

print(my\_list) # Output: [1, 2, 4, 5]

2. Delete a slice of elements:

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

del my\_list[1:3] # Delete elements at indices 1 and 2

print(my\_list) # Output: [1, 4, 5]

3. Delete all elements:

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

del my\_list[:] # Delete all elements

print(my\_list) # Output: [ ]

**Slicing of lists in Python**

Slicing allows you to extract a subset of elements from a list.

basic syntax:

my\_list[start:stop:step]

- start: The initial index (inclusive)

- stop: The final index (exclusive)

- step: The increment between indices (default is 1)

1. Get a subset of elements:

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

subset = my\_list[1:3]

print(subset) # Output: [2, 3]

2. Get every other element:

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

every\_other = my\_list[::2]

print(every\_other) # Output: [1, 3, 5]

3. Get the last 3 elements:

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

last\_three = my\_list[-3:]

print(last\_three) # Output: [3, 4, 5]

4. Get the first 3 elements:

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

first\_three = my\_list[:3]

print(first\_three) # Output: [1, 2, 3]

5. Reverse the list:

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

reversed\_list = my\_list[::-1]

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

**2. Tuples:** Ordered, immutable collections of items that can be of any data type.

**Working on Tuples**

1. index() - returns the index of the first occurrence of an element

2. count() - returns the number of occurrences of an element

1. index()

Find the index of the first occurrence of the element 3 in the list:

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

index = my\_list.index(3)

print(index) # Output: 2

2. count()

Count the number of occurrences of the element 3 in the list:

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

count = my\_list.count(3)

print(count) # Output: 2

**3. Sets:** Unordered collections of unique items that can be of any immutable data type.

1. add() - adds an element

1. `add()`

Add an element to the set:

my\_set = {1, 2, 3}

my\_set.add(4)

print(my\_set) # Output: {1, 2, 3, 4}

2. remove() - removes an element

2. `remove()`

Remove an element from the set:

my\_set = {1, 2, 3}

my\_set.remove(2)

print(my\_set) # Output: {1, 3}

3. discard() - removes an element if it exists

3. `discard()`

Remove an element from the set if it exists:

my\_set = {1, 2, 3}

my\_set.discard(2)

print(my\_set) # Output: {1, 3}

my\_set.discard(4) # No error if element doesn't exist

4. pop() - removes and returns an arbitrary element

4. `pop()`

Remove and return an arbitrary element from the set:

my\_set = {1, 2, 3}

popped = my\_set.pop()

print(popped) # Output: arbitrary element (e.g., 1)

print(my\_set) # Output: remaining elements (e.g., {2, 3})

5. union() - returns a new set with elements from two sets

5. `union()`

Return a new set with elements from two sets:

set1 = {1, 2, 3}

set2 = {3, 4, 5}

union\_set = set1.union(set2)

print(union\_set) # Output: {1, 2, 3, 4, 5}

6. intersection() - returns a new set with common elements from two sets

6. `intersection()`

Return a new set with common elements from two sets:

set1 = {1, 2, 3}

set2 = {3, 4, 5}

intersection\_set = set1.intersection(set2)

print(intersection\_set) # Output: {3}

7. difference() - returns a new set with elements unique to one set

7. `difference()`

Return a new set with elements unique to one set:

set1 = {1, 2, 3}

set2 = {3, 4, 5}

difference\_set = set1.difference(set2)

print(difference\_set) # Output: {1, 2}

8. clear() - removes all elements

8. `clear()`

Remove all elements from the set:

my\_set = {1, 2, 3}

my\_set.clear()

print(my\_set) # Output: set()

**4. Dictionaries:**

Unordered collections of key-value pairs where keys are unique and can be of any immutable data type, and values can be of any data type.

1. keys() - returns a view object with the dictionary's keys

1. `keys()`

Get the keys of the dictionary:

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

keys = my\_dict.keys()

print(keys) # Output: dict\_keys(['a', 'b', 'c'])

1. values() - returns a view object with the dictionary's values

2. `values()`

Get the values of the dictionary:

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

values = my\_dict.values()

print(values) # Output: dict\_values([1, 2, 3])

1. items() - returns a view object with the dictionary's key-value pairs

3. `items()`

Get the key-value pairs of the dictionary:

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

items = my\_dict.items()

print(items) # Output: dict\_items([('a', 1), ('b', 2), ('c', 3)])

1. get() - returns the value for a given key

4. `get()`

Get the value for a given key:

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

value = my\_dict.get('b')

print(value) # Output: 2

1. setdefault() - sets a value for a key if it doesn't exist

5. `setdefault()`

Set a value for a key if it doesn't exist:

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

my\_dict.setdefault('c', 3)

print(my\_dict) # Output: {'a': 1, 'b': 2, 'c': 3}

1. update() - updates the dictionary with key-value pairs

6. `update()`

Update the dictionary with key-value pairs:

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

my\_dict.update({'c': 3, 'd': 4})

print(my\_dict) # Output: {'a': 1, 'b': 2, 'c': 3, 'd': 4}

1. pop() - removes and returns a key-value pair

7. `pop()`

Remove and return a key-value pair:

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

popped = my\_dict.pop('b')

print(popped) # Output: 2

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

1. clear() - removes all key-value pairs

8. `clear()`

Remove all key-value pairs:

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

my\_dict.clear()

print(my\_dict) # Output: { }