**Python Lists**

Accessing List Elements:

Lists are accessed using the index operator [], starting from index 0.

Attempting to access indices outside the list's range raises IndexError.

Nested lists are accessed using nested indexing.

Examples:

python

Copy code

my\_list = ['p', 'r', 'o', 'b', 'e']

print(my\_list[0]) # Output: p

print(my\_list[2]) # Output: o

print(my\_list[-1]) # Output: e

List Slicing:

Extracting subsets of elements from a list using the slicing operator [:].

Allows selection of specific elements or ranges within a list.

Examples:

python

Copy code

my\_list = ['p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z']

print(my\_list[2:5]) # Output: ['o', 'g', 'r']

print(my\_list[:-5]) # Output: ['p', 'r', 'o', 'g']

print(my\_list[5:]) # Output: ['a', 'm', 'i', 'z']

Changing List Elements:

Lists are mutable, and elements can be altered using the assignment operator (=).

Individual items or a range can be changed directly.

Methods like append(), extend(), and insert() add or modify list items.

Examples:

odd = [2, 4, 6, 8]

odd[0] = 1

odd[1:4] = [3, 5, 7]

Deleting List Elements:

del keyword deletes items or entire lists.

The remove(), pop(), clear() methods are used to delete specific items or clear the entire list.

Examples:

my\_list = ['p', 'r', 'o', 'b', 'l', 'e', 'm']

del my\_list[2]

del my\_list[1:5]

List Comprehensions:

Provides a concise way to create lists.

Consists of an expression followed by a for statement inside square brackets.

Examples:

pow2 = [2 \*\* x for x in range(10)]

Additional List Operations:

Membership testing using in keyword.

Iterating through lists using for loops.

Operations like reversing a list or concatenating lists using + or extend().