|  |
| --- |
| DATA TYPE LIST |

|  |
| --- |
| WHAT IS LIST ? |
| * A list is a ordered collection of similar or different type of data items that is Iterable, mutable, indexed & can have duplicate items enclosed in [ ] separate by commas. |

|  |
| --- |
| FEATURES |
| * Items in a list are ordered. * List can store items of various types. * List items are enclosed in square brackets separated by commas.. * Items in a list are indexed. * Items in a list are mutable. * A list can have duplicate items. |

|  |
| --- |
| USE CASE |
| * When order matters or when need to store multiple values for an item. |

|  |  |  |
| --- | --- | --- |
| CREATE LIST | | |
| * List of 5 natural number | **:** | N0 = [1, 2, 3, 4, 5] |
| * An empty list | **:** | N1 = [ ] |
| * List of duplicate items | **:** | N2 = [1, 2, 2, 3, 3, 3] |
| * List with similar or different types o items | **:** | N3 = [1, 2, ‘Go’, [5, 6], ‘House’] |

|  |
| --- |
| SINGLE DIMENSIONAL LIST |
| * A list where elements are listed one after the other * Example : A list containing multiples of 5 **:** mylist01 = [5, 10, 15, 20] |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| L1 **:** | 5 | 10 | 15 | 20 |
| Positive Index **:** | 0 | 1 | 2 | 3 |

|  |
| --- |
| MULTI DIMENSIONAL LIST |
| * A list containing another list. * Example : mylist02 = [ [1, 2, 3], [‘G’, ‘o’, ‘o’, ‘d’], [4, 5, 6] ] |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 | 2 | 3 |  |  | G | o | o | d |  |  | 4 | 5 | 6 |  |
|  | 0 | 1 | 2 |  |  | 0 | 1 | 2 | 3 |  |  | 0 | 1 | 2 |  |
| 0 | | | | | 1 | | | | | | 2 | | | | |

|  |
| --- |
| ACCESS ELEMENT |

|  |
| --- |
| * Accessing using index of an item * **SYNTAX** : list\_name[index] |

|  |
| --- |
| * Example : mylist01 = [5, 10, 15, 20] |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| POSITIVE INDEX |  | NEGATIVE INDEX |  | NUMBERS |
| mylist01 [ 0 ] | = | mylist01 [ - 4 ] | = | 5 |
| mylist01 [ 1 ] | = | mylist01 [ - 3 ] | = | 10 |
| mylist01 [ 2 ] | = | mylist01 [ - 2 ] | = | 15 |
| mylist01 [ 3 ] | = | mylist01 [ - 1 ] | = | 20 |

|  |
| --- |
| * Example : mylist02 = [ [1, 2, 3], [‘G’, ‘o’, ‘o’, ‘d’], [4, 5, 6] ] |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| mylist01 [ 0 ] | = | [1, 2, 3] |  | mylist01 [ 2 ] | = | [4, 5, 6] |
| mylist01 [ 0 ][ 0 ] | = | 1 |  | mylist01 [ 2 ][ 0 ] | = | 4 |
| mylist01 [ 0 ][ 1 ] | = | 2 |  | mylist01 [ 2 ][ 1 ] | = | 5 |
| mylist01 [ 0 ][ 2 ] | = | 3 |  | mylist01 [ 2 ][ 2 ] | = | 6 |

|  |  |  |
| --- | --- | --- |
| mylist01[1] | = | [‘G’, ‘o’, ‘o’, ‘d’] |
| mylist01[1][0] | = | G |
| mylist01[1][1] | = | o |
| mylist01[1][2] | = | o |
| mylist01[1][3] | = | d |

|  |
| --- |
| * Example : mylist03 = [ [ 1 , 2 , 3 ] , [ [ ‘a’ , ‘b’ , ‘c’ ] , 4 , 5 ] , 6 ] * How to access item ‘b’ * Mylist03 [ 1 ][ 0 ][ 1 ] = ‘b’ |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | 6 |
|  | 1 | 2 | 3 |  |  |  |  |  |  |  |  |  |  |
|  | a | b | c |  | 4 | 5 |
|  | 0 | 1 | 2 |  |
|  | 0 | 1 | 2 |  |  |  | 0 | | |  | 1 | 2 |  |
| **0** | | | | | **1** | | | | | | | | | **2** |

|  |
| --- |
| ADD ELEMENT |

|  |  |
| --- | --- |
| * SINGLE ITEM * append ( ) * insert ( ) * extend ( ) | * MULTIPLE ITEMS * loop * split ( ) * split ( ) & loop |

|  |  |
| --- | --- |
| APPEND METHOD | |
| * append ( ) **:** add an item at the end of list . * Can add a list an an item. * SYNTAX : list\_name.append ( only 1 item ) |  |

|  |  |
| --- | --- |
| INSERT METHOD | |
| * Insert ( ) **:** add an item at the specific index. * SYNTAX : list\_name.insert ( index , item ) |  |

|  |  |
| --- | --- |
| EXTEND METHOD | |
| * extend ( ) **:** add an items of given list to existing list. * SYNTAX : list\_name.extend ( iterable item ) * Iterable Items : str, list, tuple, dict, set, f-set. |  |

|  |
| --- |
| SPLIT METHOD |
| * Splits a string into list of substrings * SYNTAX : string.split ( ) * SYNTAX : string.split ( separator , maxsplit )` |
|  |

|  |
| --- |
| FOR LOOP |
|  |

|  |
| --- |
| FOR LOOP & SPLIT METHOD |
|  |
|  |
| ADD TWO LISTS WITH ‘+’ OPERATOR |
|  |

|  |
| --- |
| UPDATE LIST |

|  |
| --- |
| SINGLE ITEM : BY REFER TO INDEX OF AN ITEM |
| * List\_name[index] = value |
|  |

|  |
| --- |
| MULTIPLE ITEM : BY PASSING RANGE OF INDEX |
| * List\_name [ index(start) **:** index(end) ] |
|  |

|  |
| --- |
| REMOVE ELEMENT |

|  |
| --- |
| REMOVE METHOD |
| * Remove the specified item from list * SYNTAX **:** List\_name.remove[item] |
|  |

|  |
| --- |
| POP METHOD |
| * Remove an item at specified index. * When index not specified then it remove last element. * SYNTAX **:** list\_name.pop(index) |
|  |

|  |
| --- |
| DEL KEYWORD |
| * del list\_name[index] : removes element at specified index * del list\_name[index(start):index(end)] * del list\_name : delete the list ( list no longer exists ) |
|  |

|  |
| --- |
| CLEAR METHOD |
| * It makes the list empty. * SYNTAX : list\_name.clear( ) |
|  |

|  |
| --- |
| LIST COMPREHENSION |

|  |  |
| --- | --- |
| * Provides a shorter syntax while creating a new list from the existing list. * SYNTAX : list\_name = [ expression for item in iterable if condition == True ] | |
|  | |
|  | |
| Condition : The condition is like a filter that only accepts the items that valuate to True. | Iterable : The iterable can be any iterable object, like a list, tuple, set etc. |
|  |  |
|  | |
| EXPRESSION : The expression is the current item in the iteration, but it is also the outcome, which you can manipulate before it ends up like a list item in the new list: | |
|  | |

|  |  |
| --- | --- |
| COPY METHOD | |
| * You cannot copy a list simply by typing list2 = list1, because: list2 will only be a reference to list1, and changes made in list1 will automatically also be made in list2. * SYNTAX : list\_name.copy() * SYNTAX : list\_name\_02 = list ( list\_name\_01 ) | |
|  |  |

|  |
| --- |
| LENGTH METHOD |
| * To Get the number of elements in List * SYNTAX : len ( iterable ) |
|  |

|  |
| --- |
| COUNT METHOD |
| * To know the number of appearances of 'x' element * SYNTAX : list\_name.count() |
|  |

|  |
| --- |
| INDEX METHOD |
| * SYNTAX : list\_name.index(item) |
|  |

|  |
| --- |
| REVERSE METHOD |
| * SYNTAX : list\_name.reverse() |
|  |

|  |  |
| --- | --- |
| FOR LOOP | |
| LOOP THROUGH LIST ITEMS | LOOP THROUGH INDEX NUMBERS |
|  |  |

|  |  |
| --- | --- |
| WHILE LOOP | |
| LOOP THROUGH LIST ITEMS |  |
|  | |

|  |  |
| --- | --- |
| SORT METHOD | |
| * SYNTAX : list\_name.sort() * SYNTAX : list\_name.sort(reverse=True) | |
| ALPHANUMEICALLY |  |
| DESCENDING ORDER |  |
| CASE SENSITIVE |  |
| CUSTOMIZE SORT |  |

|  |  |  |
| --- | --- | --- |
| SLICING | | |
| L [ start **:** stop ] | **:** | Items from start to ( stop - 1 ) |
| L [ start **:** ] | **:** | Items from start to rest of the list |
| L [ **:** stop ] | **:** | Items from 0 to ( stop - 1 ) |
| L [ **:** ] | **:** | Make a copy of existing list |
| L [ -1 ] | **:** | Last item in the list |
| L [ -2 **:** ] | **:** | Last two items in the list |
| L [ **: -**2 ] | **:** | Everything except last two items |
| L [ **: :** -1] | **:** | All items in the list, reversed |
| L [ -1 **: :** -1] | **:** | First two items, reversed |
| L [ **:** -3 **:** -1] | **:** | Last two items, reversed |
| L [ -3 **: :** -1] | **:** | Everything except the last two items, reversed |

|  |
| --- |
| CONCATENATION |
| * Add of two list. * Resulted in items of first list followed by items of second list. |
|  |

|  |
| --- |
| REPETITION |
|  |

|  |
| --- |
| MEMBERSHIP |
|  |