```
# In Python, a list is a built-in dynamic sized array (automatically grows and shrinks).
# We can store all types of items (including another list) in a list.
# A list may contain mixed type of items, this is possible because a list mainly stores references at ****contiguous locations**

# Ordered - Lists maintain the order of elements as they are added.
# Mutable - Can be modified (add, remove, or change elements).
# Dynamic - Grows or shrinks as needed.
# Heterogeneous - Stores items of different data types.
# Supports Duplicates - Allows repeated elements.
# Iterable  - Can be looped through using loops or comprehensions.
# Variable-Length - Size is not fixed.
# Supports Nesting  - Can contain other lists or complex structures.
# Efficient Random Access - Accessing elements by index is fast.
```

```
#empty list
empty_list = []
print(empty_list)

empty_list1 = list()
print(empty_list1)
```

```
[]
[]
```

```
#integers list
integers_list = [10,20,30]
print(integers_list)
```

```
[10, 20, 30]
```

```
#string list
string_list = ["Govind","Sahil","Sanchit"]
print(string_list)
```

```
['Govind', 'Sahil', 'Sanchit']
```

```
#boolean list
boolean_list = [True,False,True,True]
print(boolean_list)
```

```
[True, False, True, True]
```

```
#nested list
nested_list = [[10,20],["Govind","Sahil"],['A','B'],[12.0,15.45]]
print(nested_list)
```

```
[[10, 20], ['Govind', 'Sahil'], ['A', 'B'], [12.0, 15.45]]
```

```
#list from tuple using list constructor
tuple1 = (10,"Hello",12.3,True)
list1 = list(tuple1)
print(list1)
```

```
[10, 'Hello', 12.3, True]
```

```
#list from iterable(range)
my_list = list(range(5))
print(my_list)
even_numbers = list(range(0, 20, 2))
print(even_numbers)  # [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```
[0, 1, 2, 3, 4]
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```
#list of mixed data types
mixed = [1, "Python", 3.14, True]
print(mixed)
```

```
[1, 'Python', 3.14, True]
```

```
#creating a list with repeated elements
# Create a list [2, 2, 2, 2, 2]
a = [2] * 5
print(a)
# Create a list [0, 0, 0, 0, 0, 0, 0]
b = [0] * 7
print(b)
```

```
[2, 2, 2, 2, 2]
[0, 0, 0, 0, 0, 0, 0]
```

```
#convert string to list of characters
chars = list("Govind Parab")
print(chars)
```

```
['G', 'o', 'v', 'i', 'n', 'd', ' ', 'P', 'a', 'r', 'a', 'b']
```

```
#convert string to list of words using string split function which converts string of words into list
split_list = "Python is a programming language".split()
print(split_list)
```

```
['Python', 'is', 'a', 'programming', 'language']
```

```
# accsessing the list elements
mixed_types = ["String",1,3.5,True,["Kite","White"],{"name":"Govind","age":24},(1,2,3,5)]
print(mixed_types)
print(mixed_types[0])
print(mixed_types[4][1])
print(mixed_types[5]["age"])
print(list(mixed_types[5].keys())[0])
print(mixed_types[6][3])
print(mixed_types[-1][0])
```

```
['String', 1, 3.5, True, ['Kite', 'White'], {'name': 'Govind', 'age': 24}, (1, 2, 3, 5)]
String
White
24
name
5
1
```

```
#append, insert, extend
# mixed_types.append("First")
# mixed_types.append((1,4,7))
# mixed_types.insert(3,False)
# mixed_types.extend((12,3,5,4))
# mixed_types.extend([123,54])
print(mixed_types)
```

```
['String', 1, 3.5, True, ['Kite', 'White'], {'name': 'Govind', 'age': 24}, (1, 2, 3, 5)]
```

```
# difference between append and extend
a = [1, 2]
b = [3, 4]

a.append(b)
print(a)  # [1, 2, [3, 4]]

a = [1, 2]
a.extend(b)
print(a)  # [1, 2, 3, 4]
```

```
[1, 2, [3, 4]]
[1, 2, 3, 4]
```

```
# update
mixed_types[2]="update"
print(mixed_types)
```

```
['String', 1, 'update', True, ['Kite', 'White'], {'name': 'Govind', 'age': 24}, (1, 2, 3, 5)]
```

```
#clear list
mixed_types.clear()
print(mixed_types)
```

```
[]
```

```
# index method
# list_name.index(element, start, end)
mixed_types=["Govind","Sahil",(1,2,3),[142,456,589],(1,2,3)]
mixed_types.index((1,2,3),3,5)
```

```
4
```

```
#reversing

#modified original list
# mixed_types.reverse()
# print(mixed_types)

# creates new list do not modifies original list
# new_list = mixed_types[::-1]
# print(new_list)
# print(mixed_types)

#reversed - this returns a iterator and need to be converted to list this creates now new list
numbers = [1, 2, 3, 4, 5]
reversed_iterator = reversed(numbers)
# print(next(reversed_iterator))
print(list(reversed_iterator))
print(numbers)
```

```
[5, 4, 3, 2, 1]
[1, 2, 3, 4, 5]
```

```
#slicing parameters(start:stop:step) all are optional
#all
print(numbers[::])
#from position to end
print(numbers[2:])
#from start to second position
print(numbers[:2])
#negative indexing to slice
print(numbers[-3:])
#between slicing
print(numbers[1:4])
#reversing using step as negative
print(numbers[::-1])
#accessing using steps
print(numbers[::2])
#from position to end with steps
print(numbers[1::2])
```

```
[1, 2, 3, 4, 5]
[3, 4, 5]
[1, 2]
[3, 4, 5]
[2, 3, 4]
[5, 4, 3, 2, 1]
[1, 3, 5]
[2, 4]
```

```
#list from list comphrension
#using lamda
lc1 = [(lambda x : x*x) (x) for x in numbers]
print(lc1)
#direct
lc2 = [x for x in range(5)]
print(lc2)
#direct filteration
lc3 = [x for x in range(10) if x%2==0]
print(lc3)
```

```
[1, 4, 9, 16, 25]
[0, 1, 2, 3, 4]
[0, 2, 4, 6, 8]
```

```
print(sum(numbers))
print(max(numbers))
print(min(numbers))
print(len(numbers))
print(5 in numbers)
```

```
15
5
1
5
True
```

```
sc = [1, 2, 3]
sc1 = sc.copy()
sc[0] = 10

print(sc)   # Output: [10, 2, 3]
print(sc1)  # Output: [1, 2, 3]  (original values remain unchanged)
# ✅ copy() creates a new list that is independent of the original.

import copy
sc = [1, 2, [3, 4]]  # List with a nested list
sc1 = copy.copy(sc)
sc[0] = 10
sc[2][0] = 99  # Modify nested list

print(sc)   # Output: [10, 2, [99, 4]]
print(sc1)  # Output: [1, 2, [99, 4]]  (Nested list changes!)
# ⚠ Shallow Copy Issue:
# It copies the outer list, but references inner objects.
# Changes to nested elements affect both lists.

import copy
sc = [1, 2, [3, 4]]
sc1 = copy.deepcopy(sc)
sc[0] = 10
sc[2][0] = 99  # Modify nested list

print(sc)   # Output: [10, 2, [99, 4]]
print(sc1)  # Output: [1, 2, [3, 4]]  (Completely independent copy!)
# ✅ Deep Copy:
# It creates a new copy of all objects, including nested lists.
# No link between original and copied list.
```

```
[10, 2, 3]
[1, 2, 3]
[10, 2, [99, 4]]
[1, 2, [99, 4]]
[10, 2, [99, 4]]
[1, 2, [3, 4]]
```

```
joined = " ".join(["Kite","Flying"])
print(joined)
```

```
Kite Flying
```

```
# The zip() function in Python combines multiple iterables such as lists, tuples, strings, dict etc, into a single iterator of
# zip returns an iterator so you need to convert it into list
# zip() pairs elements position-wise from multiple iterables.
# Each tuple contains elements from the input iterables that are at the same position.
# It stops when the shortest iterable ends.
lop = [4,5,6,4]
tup = ("Govind","Kirti")
# print(zip(lop,tup))
list_zipppid = list(zip(lop,tup))
print(list_zipppid)
```

```
[(4, 'Govind'), (5, 'Kirti')]
```

```
#unzip
p1, p2 = zip(*list_zipppid)
print(list(p1))
print(list(p2))
```

```
[4, 5]
['Govind', 'Kirti']
```

```
# It lets you loop over items and get their index at the same time.
for x in enumerate(tup):
  print(x)

for x,val in enumerate(tup,start=1):
  print(x," ",val)
```

```
(0, 'Govind')
(1, 'Kirti')
1   Govind
2   Kirti
```

```
nested_list = [[1, 2, 3], [4, 5], [6, 7, 8]]
flattened = [num for sublist in nested_list for num in sublist]
print(flattened)  # Output: [1, 2, 3, 4, 5, 6, 7, 8]
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

```
import numpy as np

a = np.array([[1, 2], [3, 4], [5, 6]])

# Flattening list using NumPy
result = a.flatten().tolist()

print(result)
```

```
[1, 2, 3, 4, 5, 6]
```

```
numbers = [5, 2, 9, 1, 5, 6]
numbers.sort()
print(numbers)  # Output: [1, 2, 5, 5, 6, 9]

numbers = [5, 2, 9, 1, 5, 6]
numbers.sort(reverse=True)
print(numbers)  # Output: [9, 6, 5, 5, 2, 1]
```

```
[1, 2, 5, 5, 6, 9]
[9, 6, 5, 5, 2, 1]
```

```
numbers = [5, 2, 9, 1, 5, 6]
new_sorted_list = sorted(numbers)
print(new_sorted_list)  # Output: [1, 2, 5, 5, 6, 9]
print(numbers)          # Output: [5, 2, 9, 1, 5, 6] (Original list unchanged)

new_sorted_list = sorted(numbers, reverse=True)
print(new_sorted_list)  # Output: [9, 6, 5, 5, 2, 1]
```

```
[1, 2, 5, 5, 6, 9]
[5, 2, 9, 1, 5, 6]
[9, 6, 5, 5, 2, 1]
```

```
words = ["banana", "apple", "cherry", "date"]
words.sort(key=len)
print(words)  # Output: ['date', 'apple', 'banana', 'cherry']
```

```
['date', 'apple', 'banana', 'cherry']
```

```
students = [("John", 25), ("Alice", 22), ("Bob", 24)]
students.sort(key=lambda x: x[1])  # Sort by age
print(students)  # Output: [('Alice', 22), ('Bob', 24), ('John', 25)]
```

```
[('Alice', 22), ('Bob', 24), ('John', 25)]
```

```
numbers = [1, 2, 3, 4, 2, 5]
numbers.remove(2)  # Removes the first occurrence of 2
print(numbers)  # Output: [1, 3, 4, 2, 5]
```

```
[1, 3, 4, 2, 5]
```

```
numbers = [1, 2, 3, 4, 5]
removed_value = numbers.pop(2)  # Removes element at index 2 (3rd element)
```

```
print(numbers)  # Output: [1, 2, 4, 5]
print(removed_value)  # Output: 3 (returns the removed element)
```

```
[1, 2, 4, 5]
3
```

```
print(numbers.pop())
print(numbers)  # Output: [1, 2, 4]
```

```
5
[1, 2, 4]
```

```
numbers = [1, 2, 3, 4, 5]
del numbers[1]  # Removes element at index 1 (2nd element)
print(numbers)  # Output: [1, 3, 4, 5]


numbers = [1, 2, 3, 4, 5]
del numbers[1:3]  # Removes elements at index 1 and 2
print(numbers)  # Output: [1, 4, 5]

del numbers
# print(numbers)  # NameError: name 'numbers' is not defined
```

```
[1, 3, 4, 5]
[1, 4, 5]
```

```
# Method              Removes by           Returns Removed Element?          Modifies Original List?          Errors if
# remove()          Value                ❌ No                            ✅ Yes                            ✅ Yes (Valu
# pop()             Index (Default: Last) ✅ Yes                          ✅ Yes                            ✅ Yes (Inde
# del                 Index or Slice         ❌ No                             ✅ Yes                            ✅ Yes (
```

```
l1 = [1,2,3]
l2 = [4,5,6]
l3 = l1+l2
print(l3)
```

```
[1, 2, 3, 4, 5, 6]
```

```
l1 = [1,2,3,[11,22]]
l2 = l1.copy()
print(l2)
l1[0] = 369
l1[3][0]=111
print(l2)
```

```
[1, 2, 3, [11, 22]]
[1, 2, 3, [111, 22]]
```

```
l1=[10,20,[10]]
l2=l1.copy()
l2[2][0]=99
l1
```

```
[10, 20, [99]]
```

```
l3 = list(map(lambda x, y: x + y, l1, l2))
print(l1)
print(l2)
print(l3)
```

```
[10, 20, [99]]
[10, 20, [99]]
[20, 40, [99, 99]]
```

```
l1 = [5,6,7]
l2 = [1,2,3]
l3 = [i+j for i,j in zip(l1,l2)]
print(list(zip(l1,l2)))
l3
```

```
[(5, 1), (6, 2), (7, 3)]
```

```
[6, 8, 10]
```