```
"""
Lists:
List is a collection of items under a common variable name.
All the elements in a list are enclosed in [] and are separated by comma.

Characteristics of Lists:
1. Lists are ordered.
2. Lists are mutable (changeable / modifiable).
3. Lists allow duplicates.
4. Lists allow all the data types.
5. Lists allow mutiple data types.


Creation of Lists:
Creating an empty list:
Syntax:
<variable_name> = []
 (or)
<variable_name> = list()

Example:
my_list = []
print(type(my_list))
print(my_list)

my_list = list()
print(type(my_list))
print(my_list)

Creation of list with elements:
# Syntax:
<variable_name> = [<item1>, <item2>, <item3>]
Example:
my_list = [1, 2, 3, 4, 5]
print(type(my_list))
print(my_list)


Number of elements in a list:
len() - returns the count of elements present in a list.
# Syntax:
<variable_name> = len(<list_variable_name>)
Example:
my_list = [1, 2, 3, 4, 5]
n_elements = len(my_list)
print(n_elements) # 5


my_list = [1, 2, 3, 4, 5]
print(my_list)

my_list = [1, 2, 3, 4, 5, 1, 2, 3, 1, 2, 1]
print(my_list)

my_list = ["value1", "value2", "value3"]
print(my_list)

my_list = [1, "Two", 3.0, True, 4 + 5j]
print(my_list)
```

```
# Accessing the elements in a list:
Indexing:
To access the individual elements from a list, we use indices.
1. Index in a list is from 0 to (length of list - 1).
2. We can also access the list elements from the end
   using negative indexing.
3. Negative indices range is from 1 to (length of list).
Syntax:
<list_variable_name>[<index>]
Example
l1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
# Positive Indexing
print(l1[0]) # 10
print(l1[9]) # 100
print(l1[4]) # 50

# Negative Indexing
print(l1[-1]) # 100
print(l1[-10]) # 10
print(l1[-4]) # 70

# print(l1[23]) # Index Error
# print(l1[-45]) # Index Error


List Slicing:
Extracting a part of the list.
Syntax:
<list_variable_name>[<start>:<stop>:<step>]
1. start value is included.
   Start value is optional.
   Default value for start is 0.
2. stop value is excluded.
   Stop value is optional.
   Default value for stop is (length of list).
3. step value is optional.
   Default value for step is 1.
Example
l1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
print(l1[3: 7]) # prints the values from 3 to 6.

# First 5 values:
print(l1[0: 5]) #
print(l1[: 5])

# Last 5 values:
print(l1[5: 10]) #
print(l1[5: ])

print(l1[:]) # [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

# Step value
print(l1[0: 10: 2])
print(l1[0: 10: 3])

print(l1[3: 7]) # [40, 50, 60, 70]
print(l1[5: 8]) # [60, 70, 80]
```

```
print(l1[2: 8: 3]) # [30, 60]
print(l1[5: 10: 2]) # [60, 80, 100]

# Slicing with Negative indices:
print(l1[-1: -6: -2])
print(l1[-3: -9: -3])


print(l1[1: 5: 2]) # [20, 40]
print(l1[-7: -1]) # [40, 50, 60, 70, 80, 90]
print(l1[7: 1: -2]) # [80, 60, 40]
print(l1[-2: -8]) # []
print(l1[-8: -2]) # [30, 40, 50, 60, 70, 80]
print(l1[2: 7: -2]) # []
print(l1[5: 5]) # []
print(l1[-8: -1: 2]) # [30, 50, 70, 90]
print(l1[-1: -8: -2]) # [100, 80, 60, 40]
"""


"""
Updating the values in the list
Syntax:
Modifying a single value
<list_variable_name>[<index>] = <new_value>

Example:
l1 = [10, 20, 30, 40, 50]
l1[0] = 0
print(l1)

# Modifying multiple elements
l1[2: 5] = [3, 4, 5]
print(l1) # [0, 20, 3, 4, 5]


l1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
l1[2: 5] = [3, 4]
print(l1) # [10, 20, 3, 4, 60, 70, 80, 90, 100]

l1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
# l1[2: 9: 2] = [1, 2, 3]
l1[2: 9: 2] = [1, 2, 3, 4]
print(l1)
"""

"""
Adding elements to a list:
1. append(<item>): Adds the element at the end of the list.
Syntax:
<list_variable_name>.append(<item>)
2. insert(<index>, <item>): Adds the element at the specified position.
Syntax:
<list_variable_name>.insert(<index>, <element>)
3. extend(<list>): Adds all the elements of the second list to the first list.
Syntax:
<list1_variable_name>.extend(<list2_variable_name>)
```

```
l1 = [10, 20, 30, 40, 50]
l1.append(60)
l1.append(70)
print(l1) # [10, 20, 30, 40, 50, 60, 70]

l1.insert(0, 100)
print(l1)
l1.insert(3, 80)
print(l1) # [100, 10, 20, 80, 30, 40, 50, 60, 70]
l1.insert(9, 90)
print(l1) # [100, 10, 20, 80, 30, 40, 50, 60, 70, 90]
l1.insert(11, 110)
print(l1) # [100, 10, 20, 80, 30, 40, 50, 60, 70, 90, 110]
l1.insert(20, 120)
print(l1) # [100, 10, 20, 80, 30, 40, 50, 60, 70, 90, 110, 120]

l1 = [10, 20, 30, 40, 50]
l2 = [1, 2, 3, 4, 5]
l1.append(l2)
print(l1) # [10, 20, 30, 40, 50, [1, 2, 3, 4, 5]]
print(len(l1)) # 6
# [10, 20, 30, 40, 50, 1, 2, 3, 4, 5]
l1 = [10, 20, 30, 40, 50]
l1.extend(l2) # [10, 20, 30, 40, 50, 1, 2, 3, 4, 5]
print(l1)
"""


"""
Removing elements from the list
1. pop(<index>) : Removes the element at the specified index
                  and returns the element.
                  If the index is not specified,
                  then it will remove the element at the last index.
                  raises IndexError if the list is empty or if the given index
doesn't exist.
Syntax:
<list_variable_name>.pop(<index>)
2. remove(<item>): Removes the first matching element from the list.
                   if the element is not present, it will raise error.
Syntax:
<list_variable_name>.remove(<item>)
3. clear(): Removes all the elements from the list.
Syntax:
<list_variable_name>.clear()
4. del : Removes the element at the specified index.
         If the index is not provided, raises syntax error.
         If the index is out of range, raises IndexError.
Syntax:
del <list_variable_name>[<index>]


l1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
removed_element = l1.pop(0)
print(removed_element)
print(l1)

removed_element = l1.pop()
print(removed_element)
```

```
print(l1)

# l1.clear()
# print(l1)

# removed_element = l1.pop()
# l1.pop(20)
print(removed_element)
print(l1)

del l1[0]
print(l1)

# del l1[]
# del l1[20]
print(l1)

l1 = [10, 20, 30, 40, 50]
print(l1.remove(10))
print(l1)
l1 = [10, 20, 30, 40, 50, 10]
l1.remove(10)
print(l1)
# l1.remove(100)

# Membership Operators:
# in, not in
l1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
print(10 in l1) # True
print(1000 in l1) # False

print(500 not in l1) # True
print(50 not in l1) # False


l1.remove(100)
print(l1)
x = 1000
if x in l1:
    l1.remove(x)
print(l1)

del l1[2: 5]
print(l1)
"""


# Iterating over a list
l = [10, 20, 30, 40, 50]
for item in l:
    print(item, end=' ')
print()

for index in range(len(l)):
    print(l[index], end=' ')


"""
Enumerate:
```

```
Enumerate will return an enumerate object.
"""
print(enumerate(l))
print(list(enumerate(l)))


for item in enumerate(l):
    print(item, end=' ')
print()

for index, value in enumerate(l):
    print(f"Index = {index} and value is {value}")


"""
l1 = [10, 20, 30, 40, 50]
l2 = [60, 70, 80, 90, 100]
for index in range(len(l1)):
    print(l1[index], l2[index])

l1 = [10, 20, 30, 40, 50]
l2 = [60, 70, 80, 90, 100]
l3 = [110, 120, 130, 140, 150]
for index in range(len(l1)):
    print(l1[index], l2[index], l3[index])


l1 = [10, 20, 30, 40, 50]
l2 = [60, 70, 80, 90]
l3 = [110, 120, 130]
for index in range(len(l1)):
    print(l1[index], l2[index], l3[index])
"""


"""
zip:
To combine two or more lists
zip returns a zip object
"""
l1 = [10, 20, 30, 40, 50]
l2 = [60, 70, 80, 90, 100]
print(zip(l1, l2))
print(list(zip(l1, l2)))

l1 = [10, 20, 30, 40, 50]
l2 = [60, 70, 80, 90, 100]
l3 = [110, 120, 130, 140, 150]
print(list(zip(l1, l2, l3)))

l1 = [10, 20, 30, 40, 50]
l2 = [60, 70, 80, 90]
l3 = [110, 120, 130]
print(list(zip(l1, l2, l3)))


"""
List Methods:
1. append()
```

```
2. insert()
3. extend()
4. pop()
5. remove()
6. clear()
7. count() - returns the number of occurences of an element.
8. copy() - used to create a replica of the original list.
9. reverse()
10. sort(reverse=False) - sorts the list in either ascending or descending order.
                          if reverse is False / not specified, sorts the list in
ascending order.
                          if reverse is True, sorts the list in descending order.
11. index() - returns the first occurence of the element if it is present.
              if the element is not present, it will raise a ValueError.

Method:
<list_variable_name>.<method_name>()

Class / Function:
<function_name>(<list_variable_name>)
"""
l = [10, 20, 30, 40, 50, 10, 20, 30, 40, 10, 20, 30, 10, 20, 10]
print(l.count(10)) # 5
print(l.count(20)) # 4
print(l.count(100)) # 0


print(l.index(50)) # 4
print(l.index(10)) # 0
print(l.index(20)) # 1
# print(l.index(100)) # ValueError: 100 is not in list
x = 100
if x in l:
    print(l.index(x))


l = [30, 10, 50, 20, 40]
l.reverse()
print(l) # [40, 20, 50, 10, 30]

l.sort()
print(l) # [10, 20, 30, 40, 50]
l = [30, 10, 50, 20, 40]
l.sort(reverse=True)
print(l) # [50, 40, 30, 20, 10]


"""
Identity Operators:
is - True if the objects refer the same memory location
     False if the objects refer different memory locations.
is not - True if the objects refer different memory locations.
         False if the objects refer the same memory location
"""
l1 = [10, 20, 30, 40, 50]
l2 = l1
l2[0] = 0
print("l2 = ", l2) # [0, 20, 30, 40, 50]
print("l1 = ", l1) # [0, 20, 30, 40, 50]
```

```
print("Equality of l1 and l2", l1 == l2)


print(id(l1))
print(id(l2))
print(l1 is l2) # True
print(l1 is not l2) # False



l1 = [10, 20, 30, 40, 50]
l2 = [10, 20, 30, 40, 50]
print("Equality of l1 and l2", l1 == l2)
print(l1 is l2) # False
print(l1 is not l2) # True

l2[0] = 0
print("l2 = ", l2) # [0, 20, 30, 40, 50]
print("l1 = ", l1) # [0, 20, 30, 40, 50]
print("Equality of l1 and l2", l1 == l2)


print(id(l1))
print(id(l2))


l1 = [10, 20, 30, 40, 50]
l2 = l1.copy()
print(id(l1))
print(id(l2))
l2[0] = 0
print("l2 = ", l2)
print("l1 = ", l1)


"""
List Comprehension:
An easy and elegant way to create a list.
It makes the code more readable and concise.

Syntax:
<list_variable_name> = [<item> for <loop_variable> in range([start,]stop[,step]) /
iterable]
<list_variable_name> = [<item> for <loop_variable> in range([start,]stop[,step]) /
iterable if <condition>]
<list_variable_name> = [<true_value> if <condition> else <false_value> for
<loop_variable> in range([start,]stop[,step]) / iterable]

Example
"""
l1 = []
for i in range(10):
    l1.append(i)

print(l1)

# Using List Comprehension
l1 = [i for i in range(10)]
print(l1)
```

```
# Create a list with numbers between n1 and n2
"""
n1 = int(input("Enter N1: "))
n2 = int(input("Enter N2: "))
l1 = []
for i in range(n1, n2 + 1):
    l1.append(i)
print(l1)

# Using List Comprehension
l1 = [i for i in range(n1, n2 + 1)]
print(l1)
"""


# 0 1 4 9 16 25 36 49 64 81 100
l1 = []
for i in range(11):
    l1.append(i * i)
print(l1)

l1 = [i * i for i in range(11)]
print(l1)


l1 = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
l2 = []
for i in l1:
    if i % 2 == 0:
        l2.append(i)
print(l2)

# Using List Comprehension
l2 = [i for i in l1 if i % 2 == 0]
print(l2)

l2 = [i for i in l1 if i % 2 == 1]
print(l2)


# [0, 0, 4, 0, 16, 0, 36, 0, 64, 0, 100]
l2 = [i if i % 2 == 0 else 0 for i in l1]
print(l2)

# Odd Numbers
l2 = [1 if i % 2 == 0 else 0 for i in l1]
print(l2)


# Nested Lists:
l1 = [1, 2, 3, [4, 5, 6], 7, 8, 9]
a = l1[3]
print(a)
print(a[1])

print(l1[3][1])
```

```
l1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(l1[2][1])


l1 = [1, [2, 3, [4, 5, 6], 7], 8, 9]
print(l1[1][2][1]) # 5


# append vs extend:
l1 = [1, 2, 3, 4, 5]
l2 = [6, 7, 8, 9, 10]
l1.append(l2)
print("List is ", l1) # [1, 2, 3, 4, 5, [6, 7, 8, 9, 10]]
print("List length is: ", len(l1)) # 6

l1 = [1, 2, 3, 4, 5]
l2 = [6, 7, 8, 9, 10]
l1.extend(l2)
print("List is ", l1) # [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print("List length is: ", len(l1)) # 10

# Operators with Lists:
# + => Concatenation
l1 = [1, 2, 3, 4, 5]
l2 = [6, 7, 8, 9, 10]
l3 = l1 + l2
print("l3 = ", l3)

# * => Repetition
l1 = [1, 2, 3, 4, 5]
l2 = l1 * 3
print(l2) # [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```