```
#! /usr/bin/python3

# Creation of List Method 1

times = 80

print ("-" * times)
print ("[1]List creation by expression")
L = [1, 2, 3]
print (L)
print ("-" * times)

print ("[2] List can contain objects of any type")
L = [1, 3.14, "Hello"]
print (L)
print ("-" * times)

print ("[3] Container types like lists, dictionary can nest within list")
L = [1, 3.14, "Hello", ["Nested", "List", "Example"], "end"]
print (L)
print ("-" * times)

print ("[4] Traversing list Method 1")
for x in L:
    print (type (x), ":", x)
print ("-" * times)

print ("[5] Traversing list Method 2 (Indexing)")
for x in range (len (L)):
    print ("L[", x, "]:", L[x])
print ("-" * times)

print ("[6] Length operation")
print ("len(L):", len (L));
print ("-" * times)

print ("[7] Concatenation of two lists")
L1 = [1, 2, 3]
L2 = [4, 5, 6]
L3 = L1 + L2
print ("L1=", L1, "| L2=", L2, "| L3=L1+L2=", L3)
print ("-" * times)

print ("[8] Multiplying a list")
L1_3 = L1 * 3
print ("L1=", L1, "| L1_3=L1*3=", L1_3)
print ("-" * times)

print ("[9] Testing for membership")
print ("L1=", L1, "| 3 in L1=", 3 in L1, "| 10 in L1=", 10 in L1)
print ("-" * times)
```

```
print ("[10] List creation Method 2 : List comprehensions")
L1 = [ c for c in 'SPAM' ]
print ("[c for c in 'SPAM']:", L1)
L2 = [ x for x in range (2,10) ]
print ("[x for x in range (2,10)]:", L2)
res = [c*4 for c in 'SPAM']
print ("[c*4 for c in 'SPAM']:", res)
print ("-" * times)

print ("[11] Indexing")
L1 = [1, 2, 3, 4, "Hello", "Python", "World"]
print ("L[0]:", L1[0])
print ("L[-1]:", L1[-1])
print ("L[-len(L)]:", L1[-len(L1)])
print ("-" * times)

print ("[12] Slicing")
print ("L1[0:4]:", L1[0:4])
print ("L1[5:len (L)]:", L1[4:len(L1)])
print ("L1[::]:", L1[::])
print ("L1[-2:-6:-2]:", L1[-2:-6:-2])
print ("L1[2:6:2]:", L1[2:6:2])
print ("-" * times)

print ("[13] Matrix:Using list of lists")
M = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print ("Printing rows and invidual element using indexing")
print ("M[0]:", M[0])
print ("M[1]:", M[1])
print ("M[2]:", M[2])
print ("M[0][0]:", M[0][0], "| M[0][1]:", M[0][1], "| M[0][2]:", M[0][2])
print ("M[1][0]:", M[1][0], "| M[1][1]:", M[1][1], "| M[1][2]:", M[0][2])
print ("M[2][0]:", M[2][0], "| M[2][1]:", M[2][1], "| M[2][2]:", M[0][2])
print ("Printing Matrix by iteration")
for row in range (len(M)):
    for col in range (len (M[row])):
        print ("M[", row, "][", col, "]:", M[row][col])
print ("-" * times)

print ("[14]:Indexing and slice assignment")
print ("Lists can be changed in place. A range within a list can be")
print ("assigned simultaneously by slice assignment")
L = [x for x in range (1,10)]
print ("Original list:L:", L)
L[5] = 100
print ("After L[5]=100:", L)
L[2:4] = [200, 300, 400]
print ("After L[2:4]=[200, 300, 400]:", L)
L[6:len(L):2] = [600, 800]
print ("After L[6:len(L):2]=[600, 800]:", L)
print ("Before L[7:8]=[], L:", L, "| L[7]:", L[7])
L[7:8] = []
```

```
print ("After L[7:8]=[]:", L)
print ("-" * times)

print ("[15]:Appending to list")
L = [1, 2, 3]
print ('Before L.append ("Hello") L:', L)
L.append ("Hello")
print ('After L.append ("Hello") L:', L)
L.append ([4,5,6])
print ("After L.append ([4,5,6]) L:", L)
print ("-" * times)

print ("[16]:Sorting a list")
L = ["Hello", "python", "excellent", "World", "language"]
print ("Original L:", L)
L.sort ()
print ("After L.sort (), L:", L)
L.sort (key=str.lower)
print ("After L.sort (key=str.lower):", L)
L.sort (reverse=True)
print ("After L.sort (reverse=True):", L)
L.sort (key=str.lower, reverse=True)
print ("After L.sort (key=str.lower, reverse=True):", L)
print ("-" * times)

print ("[17]:Extending a list")
L = [1,2,3]
print ("Original L:", L)
L.extend ([4,5,6])
# Notice the different between append and extend
print ("After L.extend ([4,5,6]) L:", L)
L1 = ["Hello", "Python", "World"]
res = []
for x in L1:
    res.extend (list (x))
print ("res:", res)
print ("-" * times)

print ("[18]:Poping an element from list")
print ("Original L1:", L1)
res = L1.pop ()
print ("Poped element res:", res)
print ("After L1.pop (), L1:", L1)
print ("-" * times)

print ("[19]:In place reversal of the list")
L = [x for x in range (0,10)]
print ("Original L:", L)
L.reverse ()
print ("After L.reverse (), L:", L)
print ("-" * times)
```

```
print ("[20]:del operation of list")
print ("Original L:", L)
del (L[0])
print ("After del (L[0]), L:", L)
del (L[2:len (L):2])
print ("After del (L[2:len (L):2), L:", L)
print ("-" * times)
```