```
Type "help", "copyright", "credits" or "license()" for more information.
>>> # tuple unpacking , packing
>>> # list comprehension
>>>
>>> # tuple unpacking
>>> a = (1, 2, 3)
>>> b, c, d = a
>>> b
1
>>> c
2
>>> d
3
>>> q, w = a
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    q, w = a
ValueError: too many values to unpack (expected 2)
>>> type(a)
<class 'tuple'>
>>> type(b)
<class 'int'>
>>> q, w, e, r = a
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    q, w, e, r = a
ValueError: not enough values to unpack (expected 4, got 3)
>>> # tuple packing
>>> def fun(z, y):
     return z + y, z - y

>>> fun(3,2)
(5, 1)
>>> sum_num, diff_num = fun(3,2)
>>> sum_num
5
>>> diff_num
1
>>> # List comprehension
>>> # l a list with first 10 integers 0 to 9
>>> l = []  # empty list
>>> for i in range(10):
     l.append(i)
```

```
>>> l
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> # list comprehension
>>> l_2 = [i  for i in range(10)]
>>> l_2
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> for i in range(1,10):
      print(i)


1
2
3
4
5
6
7
8
9
>>> for i in range(2,10):
      print(i)


2
3
4
5
6
7
8
9
```

>>> `for i in range(10,2):` # Nothing is printed as stop index(10) > start index(2) and
`print(i)` # step is by default 1 (default value is assumed as step is not explicitly written)

```
>>> for i in range(10,2,-1):  # prints a sequence from 10 to 2 in
                              #reverse order as step is -1
      print(i)


10
9
8
```

```
7
6
5
4
3
>>> s = "123456789"
>>> list(s)
['1', '2', '3', '4', '5', '6', '7', '8', '9']
>>> s_list = [int(i) for i in s]
>>> s_list
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> p = [(i, 2 ** i) for i in range(11)]
>>> p
[(0, 1), (1, 2), (2, 4), (3, 8), (4, 16), (5, 32), (6, 64), (7, 128),
(8, 256), (9, 512), (10, 1024)]
>>>
>>> # list comprehension with conditional
>>>
>>> # [<expression containing iterating variable> <for loop> <if>]
>>> # 7, 14, 21,.... < 1000
>>> for i in range(0, 1000, 7):
      print(i)


0
7
14
21
28
35
42
49
56
63
70
77
.  # all values are not shown to save space. Assume the pattern continues
.
.
966
973
980
987
994
```

```
>>> 994/7  # calculating the number of values till 1000 divisible by
7
142.0
>>> 1000/7 #
142.85714285714286
>>> 143
143
>>> 35 % 7
0
>>> # 7 + 7+ ...7
>>> # If a num 'x' can be expressed as a repeated sum of another num 'y' then
>>> # 'x' is said to be divisible by 'y'
>>>
>>> # any num b/w 994 and 994+7 = 1001 can't be expressed as sum of 7s
>>> m_7 = [i for i in range(0, 1000, 7)]
>>> m_7
[0, 7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98, 105, 112,
119, 126, 133, 140, 147, 154, 161, 168, 175, 182, 189, 196, 203, 210,
217, 224, 231, 238, 245, 252, 259, 266, 273, 280, 287, 294, 301, 308,
315, 322, 329, 336, 343, 350, 357, 364, 371, 378, 385, 392, 399, 406,
413, 420, 427, 434, 441, 448, 455, 462, 469, 476, 483, 490, 497, 504,
511, 518, 525, 532, 539, 546, 553, 560, 567, 574, 581, 588, 595, 602,
609, 616, 623, 630, 637, 644, 651, 658, 665, 672, 679, 686, 693, 700,
707, 714, 721, 728, 735, 742, 749, 756, 763, 770, 777, 784, 791, 798,
805, 812, 819, 826, 833, 840, 847, 854, 861, 868, 875, 882, 889, 896,
903, 910, 917, 924, 931, 938, 945, 952, 959, 966, 973, 980, 987, 994]
>>> # Generate a list of num divisible by 7 from 0 till 1000
>>> n = []
>>> for i in range(1000):
     if i % 7 == 0:
          n.append(i)


>>> n
[0, 7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98, 105, 112,
119, 126, 133, 140, 147, 154, 161, 168, 175, 182, 189, 196, 203, 210,
217, 224, 231, 238, 245, 252, 259, 266, 273, 280, 287, 294, 301, 308,
315, 322, 329, 336, 343, 350, 357, 364, 371, 378, 385, 392, 399, 406,
413, 420, 427, 434, 441, 448, 455, 462, 469, 476, 483, 490, 497, 504,
511, 518, 525, 532, 539, 546, 553, 560, 567, 574, 581, 588, 595, 602,
609, 616, 623, 630, 637, 644, 651, 658, 665, 672, 679, 686, 693, 700,
707, 714, 721, 728, 735, 742, 749, 756, 763, 770, 777, 784, 791, 798,
805, 812, 819, 826, 833, 840, 847, 854, 861, 868, 875, 882, 889, 896,
903, 910, 917, 924, 931, 938, 945, 952, 959, 966, 973, 980, 987, 994]
```

```
>>> n_1 = [i for i in range(1000) if i % 7 == 0 ]  # using list
comprehension with conditional
>>> n_1
[0, 7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98, 105, 112,
119, 126, 133, 140, 147, 154, 161, 168, 175, 182, 189, 196, 203, 210,
217, 224, 231, 238, 245, 252, 259, 266, 273, 280, 287, 294, 301, 308,
315, 322, 329, 336, 343, 350, 357, 364, 371, 378, 385, 392, 399, 406,
413, 420, 427, 434, 441, 448, 455, 462, 469, 476, 483, 490, 497, 504,
511, 518, 525, 532, 539, 546, 553, 560, 567, 574, 581, 588, 595, 602,
609, 616, 623, 630, 637, 644, 651, 658, 665, 672, 679, 686, 693, 700,
707, 714, 721, 728, 735, 742, 749, 756, 763, 770, 777, 784, 791, 798,
805, 812, 819, 826, 833, 840, 847, 854, 861, 868, 875, 882, 889, 896,
903, 910, 917, 924, 931, 938, 945, 952, 959, 966, 973, 980, 987, 994]
>>> n == n_1
True
>>> len(n_1)
143

>>> # Assignment/ Exercise
>>> # Generate a list of tuple housing the multiples of 7 along with
its index
>>> # (0,0), (1,7),(2,14)
>>> # [(0,0), (1,7), (2,14),..... ,(142,994)]
>>>
```