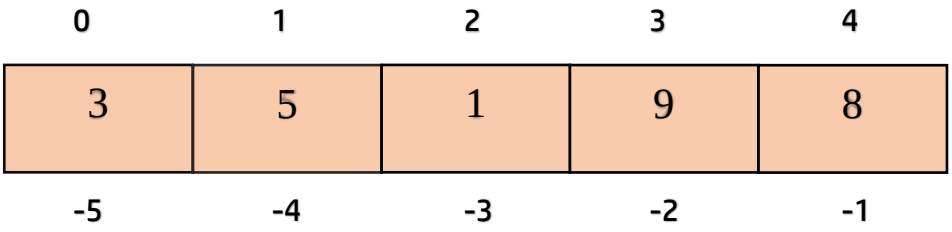
Lesson-6: Sequence Types (List & Tuple)

- List is a linear mutable data structure
- Tuple is a linear immutable data structure
- Sequences follow zero-based indexing



Program: 1

```
ints        = [1, 2, 3]
strings     = ['abc','def','ghi']
floats      = [1.2, 2.3, 3.4]
mixed       = [1, 2, 'abc', 1.2]
empty       = [ ]

print( ints )
print( strings )
print( floats )
print( mixed )
print(empty)
```

Output:

Draw The Diagram:

Original: l = [1, 2, 3]

1. l[2] = 5
2. del l[1]
3. l.insert(2,5)
4. l.append(6)
5. l.sort()
6. l.reverse()
7. l.sort(reverse=True)

Program: 2

```
ints     = (1, 2, 3)
strings  = ('abc','def','ghi')
floats   = (1.2, 2.3, 3.4)
mixed    = (1, 2, 'abc', 1.2)
empty    = ( )

print( ints )
print( strings )
print( floats )
print( mixed )
print(empty)
```

Output:

Guess The Ouput:

Original: 1) l = (5, 2, 3) 2) s = "Subhash"

```
>>> l.sort( )          >>> l.reverse( )          >>> l.append(6)

>>> l.insert(2, 'Hello')   >>> del l[1]          >>> s.reverse()

>>>  s.sort()
```

Guess The Output:

s = (1, 2, 3, 4) w = (5, 6) t = (1,3.5,3) u = (1, '3', 4)	s = [1, 2, 3, 4] w = [5,6] t = [1,3.5,3] u = [1, '3', 4]	s = 'hello' w = '!'

print(len(s))
print(s[0])
print(s[1:4])
print(s[1:])
print(s.count('e'))
print(s.count(4))
print(s.index('e')) //Error
print(s.index(3))
print('h' in s)
print(s + w)
print(min(s))
print(max(s))
print(sum(s))
print(max(t)) //3.5
print(max(u)) //Error
	print(len(s))
	print(s[0])
	print(s[1:4])
	print(s[1:])
	print(s.count('e'))
	print(s.count(4))
	print(s.index('e')) //Error
	print(s.index(3))
	print('h' in s)
	print(s + w)
	print(min(s))
	print(max(s))
	print(sum(s))
	print(max(t)) //3.5
	print(max(u)) //Error
		print(len(s))
		print(s[0])
		print(s[1:4])
		print(s[1:])
		print(s.count('e'))
		print(s.count(4)) //Error
		print(s.count('4')) //0
		print(s.index('e'))
		print(s.index(3)) //Error
		print('h' in s)
		print(s + w)
		print(min(s))
		print(max(s))
		print(sum(s)) //Error

Guess The Output:

```
>>> [1, 2, 3] + ( 4, 5, 6 )
```

```
>>> (1)
```

```
>>>(1,)
```

```
>>> 'coco' + 'nut'
```

```
>>> s = [10, 30, 20, 10]
>>> s[1:3]
```

```
>>> s = "Subhash"
>>> s[4:]
```

```
>>> s = (20,30,50,40,90,100)
>>> s[1:5]
```

Draw The Diagram:

```
l = [ [2,3,4], [5,4,7], [2,5,6] ]
```

```
t = ( (2,3,4), (5,6,7), (2,5,6) )
```

Program: 3 **Output:**

```
l = [ [2,3,4], [5,4,7], [2,5,6] ]
t = ( (2,3,4), (5,6,7), (2,5,6) )


print( l[1][1] )
print( t[1][1] )

print( l[2][0] )
print( t[2][0] )

print( l[0][2] )
print( t[0][2] )
```

Program: 4 **Output:**

```
#finding average score of the first subject of the class

class_grades = [ [85,91,89], [78,81,86], [62,75,77] ]

k = 0
sum = 0

while (k < len(class_grades)):
    sum = sum + class_grades[k][0]
    k = k + 1

average = sum / len(class_grades)

print(average)
```

Program: 5 **Output:**

```
#finding average exam score for each student in class
```

```
#finding average exam score for each student in class

class_grades = [ [85,91,89], [78,81,86], [62,75,77] ]

i = 0
j = 0
avg = []
sum = 0

while ( i < len(class_grades)):
    while( j < len(class_grades[i])):
        sum = sum + class_grades[i][j]
        j = j + 1

    sum = sum / len(class_grades[i])
    print( "The average of {0}th student is {1}".format(i,sum))
    i = i + 1
    j = 0
    sum = 0
```

Program: 6 **Output:**

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

for i in nums:
    print(i)

for i in (1, 2, 3, 4, 5):
    print(i)

for i in "hello":
    print(i)

for i in range(5):
    print(i)

for i in range(3,5):
    print(i)

for i in range(0,10,2):
    print(i)

for i in range(10,0,-2):
    print(i)
```

Guess The Output:

```
>>> for i in range [0, 11]:
        print(i)
```

```
>>> for i in range(10, -1, -2):
        print(i)
```

Write the 'while' equivalent of the following:

Original Code:

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

for i in nums:
    print(i)
```

Equivalent 'while' code:

Program: 7

```
#This is a self-observation program to analyse and appreciate python constructs

password_out = ''
case_changer = ord('a') - ord('A')
encryption_key = (('a','m'), ('b','h'), ('c','t'), ('d','f'), ('e','g'), ('f','k'), ('g','b'), ('h','p'), ('i','j'), ('j','w'), ('k','e'), ('l','r'),
('m','q'), ('n','s'), ('o','l'), ('p','n'), ('q','i'), ('r','u'), ('s','o'), ('t','x'), ('u','z'), ('v','y'), ('w','v'), ('x','d'), ('y','c'), ('z','a'))

print( "This program will encrypt and decrypt user passwords\n" )

which = input("Enter (e) to encrypt a password, and (d) to decrypt a password\n")

while which != 'e' and which != 'd':
    which = input("INVALID - Enter 'e' to encrypt, 'd' to decrypt\n")

encrypting = (which == 'e')

password_in = input("Enter password:")

if encrypting:
    from_index = 0
    to_index   = 1
else:
    from_index = 1
    to_index   = 0

case_changer = ord('a') - ord('A')

for ch in password_in:
    letter_found = False

    for t in encryption_key:
        if ('a' <= ch and ch <= 'z') and ch == t[from_index]:
            password_out = password_out + t[to_index]
            letter_found = True
        elif ('A' <= ch and ch <= 'Z') and chr(ord(ch) + 32) == t[from_index]:
            password_out = password_out + chr(ord(t[to_index]) - case_changer)
            letter_found = True

    if not letter_found:
```

```
        password_out = password_out + ch

if encrypting:
    print( "Your encrypted password is: ", password_out)
else:
    print( "Your decrypted password is: ", password_out)
```

Output:

```
$ python3 encrpyt_decrypt_program.py

This program will encrypt and decrypt user passwords

Enter (e) to encrypt a password, and (d) to decrypt a password
e
Enter password:subhash
Your encrypted password is:  ozhpmop

$ python3 encrpyt_decrypt_program.py
This program will encrypt and decrypt user passwords

Enter (e) to encrypt a password, and (d) to decrypt a password
d
Enter password:ozhpmop
Your decrypted password is:  subhash

$ python3 encrpyt_decrypt_program.py
This program will encrypt and decrypt user passwords

Enter (e) to encrypt a password, and (d) to decrypt a password
e
Enter password:8889
Your encrypted password is:  8889
```

Program: 8 **Output:**

```
nums_one = [ 1, 3, 4, 5 ]
num_two  = nums_one

nums_two[2] = 8
print( nums_one )
print( nums_two )

print( id(nums_one) )
print( id(nums_two) )


l_one = [1,2,3]
l_two = list(l_one)
print(l_one)
print(l_two)

l_two[0] = 0
```

```
print(l_one)
print(l_two)

print( id(l_one) )
print( id(l_two) )
```

Guess The Output:

Note: These are called 'List Comprehensions'

```
>>> [ x for x in range(10) if (x % 2 == 0) ]


>>> [ x**2 for x in range(5) ]


>>>[ x for x in [-1, 1, -2, 2, -3, 3, -4, 4] if( x >= 0) ]


>>> [ord(x) for x in 'Hello']


>>> vowels = ('a', 'e', 'i', 'o', 'u' )
>>> w = 'Hello'
>>> [ch for ch in w if ch in vowels]


>>> temperatures = [88,94,97,89,101,98,102,95,100]
        [ (t - 32) * 5/9 for t in temperatures ]

>>> a, b = [int(x) for x in input("Enter 2 numbers").split()]


>>>  eval("[1,2,3] + [1,2,3]")
```

Program 9:

```
l = range(0,9)
print(l)

l = list(range(0,9))
print(l)

lone = [1,2,3,4,5]
lone[1:2] = 10,11
print(lone)

print(lone * 2)

ltwo = lone
ltwo[0] = 23
```

Output:

```
lone[3] = 35
print(lone)
print(ltwo)

lthree = ltwo[:]
lthree[1] = 89
print(lthree)
print(ltwo)

lfour = lthree.copy()
lfour[1] = 98
print(lthree)
print(lfour)

nl = [9,7,6]
lfive = [1,2,3,nl]
print(lfive)

for i in lfive[3]:
    print(i, end = " ")
print()

print(lfive[3][0])
print(lfive[3][1])
print(lfive[3][2])

lsix = [[1,2,3],[4,5,6],[7,8,9]]

for lseven in lsix:
    print(lseven)

for lseven in lsix:
    for num in lseven:
        print(num, end = " ")
    print()

leight = [1,2,3]
lnine  = [1,1,1]
lten = [i + j for i in leight for j in lnine ]
print(lten)

leleven = [ 4 * [0] for i in range(3) ]
print(leleven)


leleven = [ 4 * [i] for i in range(3) ]
print(leleven)
```

Program 10:

```
tone = 1,2,3
print(tone)

l = [11,22,33]
ttwo = tuple(l)
print(ttwo)

tthree = tuple(range(11,20))
print(tthree)

print(tthree[::2])

num1, num2 = tone[0:2]
print(num1)
print(num2)

tfour = tthree * 2
print(tfour)

tfive = (1,2,3,(4,5,6))

for t in tfive:
    print(t)

for t in tfive[3]:
    print(t)
```

Output:

Few More Methods Of List – REMOVE, POP and EXTEND:

```
>>> l = [1,2,3]
>>> l.remove(1)
>>> l
[2,3]
```

```
>>> l = [1,2,3]
>>> l.pop( )
3
>>> l
[1,2]
>>>l.pop(0)
1
>>> l
[2]
```

```
>>> l = [1,2,3]
>>> l.extend([5,6])
>>> l
[1,2,3,5,6]
```

Program 11:

Given two different sequences, write a program to find out how many elements are common in those sequences. **Hint**: Use **zip( )** method.

```
count = 0

myname = ['s','u','b','h','a','s','h']
yourname = ('S','u','B','H','a','S','h')

for m, y in zip(myname, yourname):
   if m == y:
      count = count + 1;

print(count, "values are equal")
```

Output:

```
(1, 3)
(2, 4)
```

NOTE: How **zip( )** function works?

```
>>> l = [1,2]
>>> t = (3, 4)
>>> zip(l,t)
>>> for pair in zip(l,t):
        print(pair)
```

Program 12 (Same Program, Broken To One More Step):

Given two different sequences, write a program to find out how many elements are common in those sequences. **Hint**: Use **zip( )** method.

Output:

```
count = 0

myname = ['s','u','b','h','a','s','h']
yourname = ('S','u','B','H','a','S','h')

for pair in zip(myname, yourname):
   print(pair)
   m,y = pair
   if m == y:
      count = count + 1;

print (count, "values are equal")
```

```
('s', 'S')
('u', 'u')
('b', 'B')
('h', 'H')
('a', 'a')
('s', 'S')
('h', 'h')
3 values are equal
```

Format Operator:

```
print( 'sum of %d and %d is %d' % (1, 2,3))
print( 'My name is %s and I work as %s since %d years' % ('Subhash', 'Programmer', 15))
print( 'My name is %s and I work as %s since %f years' % ('Subhash', 'Programmer', 10))
```

Output:

```
sum of 1 and 2 is 3
My name is Subhash and I work as Programmer since 15 years
My name is Subhash and I work as Programmer since 10.000000 years
```

Programming Assignments:

1. WAP to determine whether given number occurs in list 'nums'
2. WAP that prompts the user for a list of integers, stores in another list only those values between 1-100, and displays the resulting list
3. WAP that prompts the user for a list of integers, stores in another list only those values that are in tuple 'valid_values', and displays the resulting list.
4. WAP that prompts the user for a list of integers and stores them in a list. For all values that are greater than 100, the string 'josh' should be stored instead. The program should display the resulting list.
5. WAP that prompts the user to enter a list of first names and stores them in a list. The program should display how many times the letter 'a' appears within the list.
6. WAP that prompts the user to enter integer values for each of two lists. It then should display the lists are of the same length, whether the elements in each list sum to the same value and whether there are any values that occur in both lists.
7. WAP to modify or replace an existing element of a tuple with a new element
8. WAP to find the first occurrence of an element in a tuple
9. WAP to accept elements in the form of a tuple and display their sum and average
10. WAP to add two matrices and display the sum matrix using lists
11. WAP to sort the list elements using bubble sort technique