Sets

In [1]:
```
a = [ 10, 20, 30, 10, 50]
print(a)
print(type(a))
```

```
[10, 20, 30, 10, 50]
<class 'list'>
```

In [2]:
```
b = { 10, 20, 30, 10, 50}
print(b)
print(type(b))
```

```
{10, 20, 50, 30}
<class 'set'>
```

In [3]:
```
c = {10,10,10,10,10}
print(c)
```

```
{10}
```

In [4]:
```
d = { 10,20,30,40,}
print(d)
```

```
{40, 10, 20, 30}
```

In [5]:
```
d = { 10,20,30,40}
print(d[2])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_4140\2678402605.py in <module>
      1 d = { 10,20,30,40}
----> 2 print(d[2])

TypeError: 'set' object is not subscriptable
```

In [6]:
```
s1 = {20,30,60}
s2 = {20,40,70}
#intersection of s1 and s2
print(s1.intersection(s2))
```

```
{20}
```

In [7]:
```
s1 = {20,30,60}
s2 = {25,40,70}
#intersection of s1 and s2
print(s1.intersection(s2))
```

```
set()
```

In [8]:
```
s1 = {20,30,60}
s2 = {25,40,70}
#Union of s1 and s2
print(s1.union(s2))
```

```
{20, 70, 40, 25, 60, 30}
```

In [9]:
```
s1 = {20,35,60}
s2 = {20,45,70}
#Union of s1 and s2
print(s1.union(s2))
```

```
{35, 20, 70, 60, 45}
```

In [10]:
```
s = 'aaaabbbbbccccczzzzlllllhhhhhhiiiiirrrrrraaaaaakkkkkkkk'
print(set(s))
```

```
{'r', 'k', 'c', 'h', 'i', 'l', 'b', 'a', 'z'}
```

In [11]:
```
s = 'aaaabbbbbccccczzzzlllllhhhhhhiiiiirrrrrraaaaaakkkkkkkk'
print(set(s))
```

```
{'r', 'k', 'c', 'h', 'i', 'l', 'b', 'a', 'z'}
```

In [12]:
```
s = 'aaaabbbbbccccczzzzlllllhhhhhhiiiiirrrrrraaaaaakkkkkkkk'
print(set(s))
```

```
{'r', 'k', 'c', 'h', 'i', 'l', 'b', 'a', 'z'}
```

In [14]:
```
lst = [10,20,30,40,20,30,20,10,30,10,50]
print(Set(lst))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_4140\1928208982.py in <module>
      1 lst = [10,20,30,40,20,30,20,10,30,10,50]
----> 2 print(Set(lst))

NameError: name 'Set' is not defined
```

In [15]:
```
lst = [10,20,30,40,20,30,20,10,30,10,50]
print(set(lst))
```

```
{40, 10, 50, 20, 30}
```

In [16]:
```
s = {13,26}
s.add(39)
print(s)
```

```
{26, 13, 39}
```

In [18]:
```
s = (16,32,48,64,80,96,112,128,144,160)
s.remove(160)
print(s)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_4140\3436382619.py in <module>
      1 s = (16,32,48,64,80,96,112,128,144,160)
----> 2 s.remove(160)
      3 print(s)

AttributeError: 'tuple' object has no attribute 'remove'
```

In [19]:
```
s = {16,32,48,64,80,96,112,128,144,160}
s.remove(160)
print(s)
```

```
{64, 32, 96, 128, 16, 48, 80, 112, 144}
```

Lists

- Using square brackets[]
- [10,20,30] --> List of integers
- [10.2,30,4] --> List of floating point values
- ['a','b','c'] --> List of strings
- [[10,20],[30,40],[50,60]] --> List of lists

In [9]:
```
lst = [10,20,30,40]
#ind   0   1  2  3
print(lst[2])
```

```
print(lst[3])
print(lst[4])
```

```
30
40
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_4148\2190355527.py in <module>
      3 print(lst[2])
      4 print(lst[3])
----> 5 print(lst[4])

IndexError: list index out of range
```

In [10]:
```
s = "HELLO WORLD"
#ind 012345678910
print(s[6])
print(s[0])
print(s[12])
```

```
W
H
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_4148\226015912.py in <module>
      3 print(s[6])
      4 print(s[0])
----> 5 print(s[12])

IndexError: string index out of range
```

In [11]:
```
r=range(10,100,10)
print(r[7])
```

```
80
```

In [12]:
```
#ind   -4 -3 -2 -1
lst = [10,20,30,40]
#ind   0   1  2  3
print(lst[2])
print(lst[3])
print(lst[-2])
print(lst[-3])
```

```
30
40
30
20
```

In [13]:
```
lst = [10,20,'hello',[10,20]]
print(len(lst))
```

```
4
```

In [14]:
```
s='hello world'
print(len(s))
```

```
11
```

In [15]:
```
help(len)
```

```
Help on built-in function len in module builtins:

len(obj, /)
    Return the number of items in a container.
```

Reading a list of integers in a single line

In [ ]:
```
x=list(map(int, input().split()))
print(x)
print(type(x))
print(len(x))
print(sum(x))
```

In [1]:
```
#Harry Hermoine Wesely Malfoy Dumbledore
names = list(map(str, input().split))
```

```
Harry Hermoine Wesely Malfoy Dumbledore
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11324\1648912828.py in <module>
      1 #Harry Hermoine Wesely Malfoy Dumbledore
----> 2 names = list(map(str, input().split))

TypeError: 'builtin_function_or_method' object is not iterable
```

In [2]:
```
#Harry Hermoine Wesely Malfoy Dumbledore
names = list(map(str, input().split))
```

```
print(names)
```

```
Harry Hermoine Wesely Malfoy Dumbledore
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11324\3632002098.py in <module>
      1 #Harry Hermoine Wesely Malfoy Dumbledore
----> 2 names = list(map(str, input().split))
      3 print(names)

TypeError: 'builtin_function_or_method' object is not iterable
```

In [3]:
```
#Harry Hermoine Wesely Malfoy Dumbledore
names = list(map(str, input().split()))
print(names)
```

```
Harry Hermoine Wesely Malfoy Dumbledore
['Harry', 'Hermoine', 'Wesely', 'Malfoy', 'Dumbledore']
```

Reading a list of integers in different lines

In [ ]:
```
n =int(input())
lst = []
for i in range(n):
    x = int(input())
    lst.append(x)
print(lst)
print(type(lst))
print(sum(lst))
```

In [ ]:
```
names = []
n = int(input())  #Length of a list
for _ in range(n):
    s = int(input)   #reading each element
    names.append(s)  #appending it to the list
print(names)
```

In [ ]:
```
"""Harry
Hermoine
Wesely
Malfoy
Dumbledore"""
names = []
n = int(input())  #Length of a list
```

```
for _ in range(n):
    s = input()   #reading each element
    names.append(s)  #appending it to the list
print(names)
```

Traversal of Lists

- Movement from one element to another is known as traversing a list
- It is of two types.
  - Element based traversal
  - Index based traversal

Element based traversal

In [ ]:
```
l = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
#ind   0  1   2   3   4   5   6   7   8    9
for element in l:
    print(element, end = " ")
```

In [ ]:
```
e = [11, 22, 36, 47, 56, 66, 79, 83, 94, 100]
#ind   0  1   2   3   4   5   6   7   8    9
cnt = 0
for _ in e:
    if _%2==0:
        cnt+=1
print(cnt)
```

In place operations to the list cannot be done using element based transversal

In [5]:
```
x = [10, 20, 30, 40, 50]
#[11,21,31,41,51]  --->Expected output
for i in x:
    i+=1
print(x)
```

```
[10, 20, 30, 40, 50]
```

This happened because here we take copy of value in element

Index based traversal

In [2]:
```
l = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
#ind   0  1   2   3   4   5   6   7   8    9
for i in range(len(s)):
    print(s[i],end = " ")
```

```
10 20 30 40 50 60 70 80 90 100
```

Count even numbers that are present at even indices in the list

In [3]:
```
e = [11, 22, 36, 47, 56, 66, 79, 83, 94, 100]
#ind   0  1   2   3   4   5   6   7   8    9
cnt=0
for i in range(len(e)):
    if i%2==0:
        if e[i]%2==0:
            cnt+=1
print(cnt)
```

```
3
```

In place operations to the list cannot be done using element based transversal

In [10]:
```
x = [10, 20, 30, 40, 50]
#[11,21,31,41,51]  --->Expected output
for i in range(len(x)):
    x[i]+=1
    print(x[i], end = " ")
```

```
11
21
31
41
51
```

List Methods

- Lists are mutable
- Mutability : Ability to change after creating.
- Mutable Objects: Lists, Sets, Dictionaries

- Immutable Objects : Strings,Tuples, Integers, Floating Point values

Insertion Methods

- append()
- Insert()
- extend()

Deletion Methods

- pop()
- remove()
- clear()

Searching Methods

- index()
- count()

In place Modifications

- reverse()
- sort()

-copy()

In [37]:
```
# Muttability
marks = [67,98,89,95,99]
print(marks)
marks[0] = 91
print(marks)
```

```
[67, 98, 89, 95, 99]
[91, 98, 89, 95, 99]
```

In [38]:
```
#Immutability
s = 'hallo'
print(s)
s[1] = 'e'
print(s)
```

```
hallo
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_24912\123622334.py in <module>
      2 s = 'hallo'
      3 print(s)
----> 4 s[1] = 'e'
      5 print(s)

TypeError: 'str' object does not support item assignment
```

List Append

- used to append an object at the end of the list.
- can append one object at a time.

In [11]:
```
lst=[]
lst.append(10)
lst.append('Hello')
lst.append([10,20,30])
print(lst)
```

```
[10, 'Hello', [10, 20, 30]]
```

for ages [14,7,41,65,44,79,6,2,21,22,43,68,1,90] print the following lists based on the given condition:

- child = < 13
- teen = >=13 and <=19
- adult = >=20 and <50
- old = >=50

In [14]:
```
ages = [14 ,7, 41, 65, 44, 79, 6, 2, 21, 22, 43, 68, 1, 90]
child = []
teen = []
adult = []
old = []
for i in ages:
    if i < 13:
        child.append(i)
    elif 13 <= i <= 19:
        teen.append(i)
    elif 20 <= i <50:
        adult.append(i)
    else:
        old.append(i)
print(child,teen,adult,old,sep = "\n")
```

```
[7, 6, 2, 1]
[14]
[41, 44, 21, 22, 43]
[65, 79, 68, 90]
```

List Extend

- Used to extend an existing list with given iterable.
- Takes every every element of the iterable and appends it to the existing list.
- We should only pass iterables as an arguments to extend() method

In [15]:
```
marks = [45,65,87] #creted list
new_marks = [98,67,43]
marks.append(new_marks)
print(marks)
```

```
[45, 65, 87, [98, 67, 43]]
```

In [16]:
```
marks = [45,65,87] #creted list
new_marks = [98,67,43]
marks.extend(new_marks)
print(marks)
```

```
[45, 65, 87, 98, 67, 43]
```

In [17]:
```
ch = ['A','B','C','D','E']
ch.append('FGHIJ')
print(ch)
```

```
['A', 'B', 'C', 'D', 'E', 'FGHIJ']
```

In [18]:
```
ch = ['A','B','C','D','E']
ch.extend('FGHIJ')
print(ch)
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
```

List insert

insert(index,object)

- Inserts the given object at specified index.
- Moves all the existing elements one step towards right to insert the given object at specified index.

In [19]:
```
x = [10, 20, 30, 40]
#ind   0  1   2   3
print(x)
x.insert(1,50)
print(x)
```

```
[10, 20, 30, 40]
[10, 50, 20, 30, 40]
```

Print list of individual characters of all strings in a list ['Harry', 'Hermoine', 'Wesely', 'Malfoy', 'Dumbledore']

In [14]:
```
output=[]
output.extend('Harry')
print(output)
```

```
['H', 'a', 'r', 'r', 'y']
```

In [9]:
```
output=[]
output.extend('Harry')
print(output)
output.extend('Hermoine')
```

```
print(output)
output.extend('Wesely')
print(output)
output.extend('Malfoy')
print(output)
output.extend('Dumbledore')
print(output)
```

```
['H', 'a', 'r', 'r', 'y']
['H', 'a', 'r', 'r', 'y', 'H', 'e', 'r', 'm', 'o', 'i', 'n', 'e']
['H', 'a', 'r', 'r', 'y', 'H', 'e', 'r', 'm', 'o', 'i', 'n', 'e', 'W', 'e', 's', 'e', 'l', 'y']
['H', 'a', 'r', 'r', 'y', 'H', 'e', 'r', 'm', 'o', 'i', 'n', 'e', 'W', 'e', 's', 'e', 'l', 'y', 'M', 'a', 'l', 'f',
'o', 'y']
['H', 'a', 'r', 'r', 'y', 'H', 'e', 'r', 'm', 'o', 'i', 'n', 'e', 'W', 'e', 's', 'e', 'l', 'y', 'M', 'a', 'l', 'f',
'o', 'y', 'D', 'u', 'm', 'b', 'l', 'e', 'd', 'o', 'r', 'e']
```

In [10]:
```
output=[]
output.extend('Harry')
output.extend('Hermoine')
output.extend('Wesely')
output.extend('Malfoy')
output.extend('Dumbledore')
print(output)
```

```
['H', 'a', 'r', 'r', 'y', 'H', 'e', 'r', 'm', 'o', 'i', 'n', 'e', 'W', 'e', 's', 'e', 'l', 'y', 'M', 'a', 'l', 'f',
'o', 'y', 'D', 'u', 'm', 'b', 'l', 'e', 'd', 'o', 'r', 'e']
```

In [11]:
```
names = ['Harry', 'Hermoine', 'Wesely', 'Malfoy', 'Dumbledore']
for i in names:
    output.extend(i)
print(output)
```

```
['H', 'a', 'r', 'r', 'y', 'H', 'e', 'r', 'm', 'o', 'i', 'n', 'e', 'W', 'e', 's', 'e', 'l', 'y', 'M', 'a', 'l', 'f',
'o', 'y', 'D', 'u', 'm', 'b', 'l', 'e', 'd', 'o', 'r', 'e', 'H', 'a', 'r', 'r', 'y', 'H', 'e', 'r', 'm', 'o', 'i', 'n',
'e', 'W', 'e', 's', 'e', 'l', 'y', 'M', 'a', 'l', 'f', 'o', 'y', 'D', 'u', 'm', 'b', 'l', 'e', 'd', 'o', 'r', 'e']
```

List Pop

- used to remove and return the element at specified index
- Index is defaulted to -1 (Removes last elements in the list)
- Returns the element after removing
- Throws an index out of bound error, if specified index is not present in the list

In [12]:
```
x = [10, 20, 30, 40, 50]
x.pop()
print(x)
```

```
[10, 20, 30, 40]
```

In [13]:
```
x = [10, 20, 30, 40, 50]
popped_element = x.pop()
print(popped_element)
print(x)
```

```
50
[10, 20, 30, 40]
```

In [1]:
```
x = [10, 20, 30, 40, 50]
popped_element = x.pop(2)
print(popped_element)
print(x)
```

```
30
[10, 20, 40, 50]
```

In [2]:
```
x = [10, 20, 30, 40, 50]
popped_element = x.pop(6)
print(popped_element)
print(x)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_24912\1956261163.py in <module>
      1 x = [10, 20, 30, 40, 50]
----> 2 popped_element = x.pop(6)
      3 print(popped_element)
      4 print(x)

IndexError: pop index out of range
```

List Remove

- removes the given elements in a list.
- Throws an error if the specified member is not present in the list.
- removes the element at lowest index if multiple elements are present.

In [39]:
```
lst = [10,20,30,40,50,60]
lst.remove()
print(lst)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_24912\4137727785.py in <module>
      1 lst = [10,20,30,40,50,60]
----> 2 lst.remove()
      3 print(lst)

TypeError: list.remove() takes exactly one argument (0 given)
```

In [40]:
```
lst = [10,20,30,40,50,60]
lst.remove(50)
print(lst)
```

```
[10, 20, 30, 40, 60]
```

In [42]:
```
lst = [10,20,30,40,50,60]
lst.remove(100)
print(lst)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_24912\2253391458.py in <module>
      1 lst = [10,20,30,40,50,60]
----> 2 lst.remove(100)
      3 print(lst)

ValueError: list.remove(x): x not in list
```

In [43]:
```
lst = [10,20,30,40,30,30,50,60,30]
lst.remove(30)
print(lst)
```

```
[10, 20, 40, 30, 30, 50, 60, 30]
```

List Clear

- Clears the contents of the list leaving an empty list

In [30]:
```
lst = [10,20,30]
```

```
lst.clear()
print(lst)
```

```
[]
```

In [31]:
```
lst=[10,20,30,40,50]
lst.clear()
print(lst)
```

```
[]
```

List Index

- Used to find the index of an element if it's a member in the list.
- Throws an error if the specified member is not present in the list.
- Gives the first occurence index if multiple occurences are present.

In [6]:
```
a = [10,20,30,40,50]
print(a.index(40))
```

```
3
```

In [7]:
```
a = [10,20,30,40,50]
print(a.index(45))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_24912\1678947136.py in <module>
      1 a = [10,20,30,40,50]
----> 2 print(a.index(45))

ValueError: 45 is not in list
```

In [9]:
```
a = [10,15,20,15,30,40,15]
print(a.index(15))
```

```
1
```

List Count

- Returns the count of a specified element in the list.

- Example: lst = [10,10,10,20]
- Applying lst.count(10) gives 3 as 10 is present for 3 times in the list.
- Returns 0 if the specified element is not in the list.

In [3]:
```
lst = [10,10,10,20,30,40,50,10]
print(lst.count(10))
```

```
4
```

In [4]:
```
lst = [10,10,10,20,30,40,50,10]
print(lst.count(100))
```

```
0
```

List Reverse

- Reverses the given list in-place.
- You'll lose your original list.
- The original list itself will modified to store the reversed version.

In [10]:
```
lst = [5,10,15,20,25,30]
print(f'List before reversal : {lst}')
lst.reverse()
print(f'List after reversal : {lst}')
```

```
List before reversal : [5, 10, 15, 20, 25, 30]
List after reversal : [30, 25, 20, 15, 10, 5]
```

List Sort

- Sorts the given list (in ascending order) in-place.
- It can only be applied on homogeneous lists.
- Takes two optional keyword arguments
  - reverse
  - key
- reverse will be set to false by default.
- setting reverse = true will sort the list in descending order.

- You can specify as a function as key, if the sort has to take place based on a function.

In [11]:
```
lst = [2,17,9,-1,-45,23,-59,99,100]
lst.sort() # ascending
print(lst)
```

```
[-59, -45, -1, 2, 9, 17, 23, 99, 100]
```

In [12]:
```
names = ['Harry', 'Hermoine', 'Wesely', 'Malfoy', 'Dumbledore']
names.sort()
print(names)
```

```
['Dumbledore', 'Harry', 'Hermoine', 'Malfoy', 'Wesely']
```

In [16]:
```
a = [45,56,10,10.5,99,11,23,34.9,'Hello']
a.sort()
print(a)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_24912\1251581164.py in <module>
      1 a = [45,56,10,10.5,99,11,23,34.9,'Hello']
----> 2 a.sort()
      3 print(a)

TypeError: '<' not supported between instances of 'str' and 'float'
```

In [24]:
```
a = [45,56,10,10.5,99,11,23,34.9]
a.sort()
print(a)
```

```
[10, 10.5, 11, 23, 34.9, 45, 56, 99]
```

In [17]:
```
lst = [2,17,9,-1,-45,23,-59,99,100]
lst.sort(reverse = True) # descending
print(lst)
```

```
[100, 99, 23, 17, 9, 2, -1, -45, -59]
```

In [18]:
```
names = ['Harry', 'Hermoine', 'Wesely', 'Malfoy', 'Dumbledore']
names.sort(reverse = True)
print(names)
```

```
['Wesely', 'Malfoy', 'Hermoine', 'Harry', 'Dumbledore']
```

In [23]:
```
names = ['Harry', 'Hermoine', 'Weasely', 'Malfoy', 'Dumbledore']
```

```
#  Lengths   5          8         7          6            10
# 5 6 7 8 9 10
# ['Harry', 'Malfoy', 'Weasely', 'Hermoine', 'Dumbledore']
names.sort(key = len)
print(names)
```

```
['Harry', 'Malfoy', 'Weasely', 'Hermoine', 'Dumbledore']
```

In [25]:
```
def fac_cnt(n):
    fc = 0
    for i in range(1,n+1):
        if n%i==0:
            fc += 1
    return fc
# Sorting based on number of factors
nums = [10,4,5,24,12]
#        4 3 2 8 6
# 5  4 10  12 24
nums.sort(key = fac_cnt)
print(nums)
```

```
[5, 4, 10, 12, 24]
```

In [32]:
```
#Functions that returns vowel count of a string
def vow_cnt(s):
    cnt = 0
    for i in s:
        if i in 'aeiou':
            cnt += 1
    return cnt
strings = ['zac', 'ruuiosq', 'paeqiz', 'apiq']
strings.sort(key = vow_cnt)
print(strings)
```

```
['zac', 'apiq', 'paeqiz', 'ruuiosq']
```

In [33]:
```
#Functions that returns vowel count of a string
def vow_cnt(s):
    cnt = 0
    for i in s:
        if i in 'aeiou':
            cnt += 1
    return cnt
strings = ['zac', 'ruuiosq', 'paeqiz', 'apiq']
strings.sort(key = vow_cnt,reverse = True)
print(strings)
```

```
['ruuiosq', 'paeqiz', 'apiq', 'zac']
```

In [34]:
```
#Functions that returns vowel count of a string
def vow_cnt(s):
    cnt = 0
    for i in s:
        if i in 'aeiou':
            cnt += 1
    return cnt
strings = ['zac', 'ruuiosq', 'paeqiz', 'apiq']
strings.sort(reverse = True,key = vow_cnt)
print(strings)
```

```
['ruuiosq', 'paeqiz', 'apiq', 'zac']
```

List Copy

- Used to copy from one list to the another.
- Deep Copy
  - If two objects are deep copied, the changes we make to one object will be reflected on the other.
- Shallow Copy
  - If two objects are shallow copied, the changes we make to one object will not be reflected on the other.
- In Python, copying 2 lists in the following manner performs a deep copy
  - lst2 = lst1
- So every change we make to lst2 will be reflected on lst1 and every change we make to lst1 will be reflected on lst2.
- To avoid this we use list.copy() method which performs shallow copy.
- These conditions are same for strings and dictionaries as well.

In [35]:
```
lst1 = [10,20,30]
lst2 = lst1       # deep copy
# modifing list1
lst1.append(40)   # 40 will be added to list2 too.
print(lst1)
print(lst2)
```

```
[10, 20, 30, 40]
[10, 20, 30, 40]
```

In [36]:
```
lst1 = [10,20,30]
lst2 = lst1.copy()      #shallow copy
```

```
# modifing list1
lst1.append(40)   # 40 will not be added to list2.
print(lst1)
print(lst2)
```

```
[10, 20, 30, 40]
[10, 20, 30]
```

Slicing

In [ ]:

List Comprehensions

In [44]:
```
# Print squares of numbers in a list
lst = [1,2,3,4,5,6]
squares = []
for i in lst:
    squares.append(i*i)
print(squares)
```

```
[1, 4, 9, 16, 25, 36]
```

In [45]:
```
# Print squares of numbers in a list using list comprehension
lst = [1,2,3,4,5,6]
squares = [i*i for i in lst]
print(squares)
```

```
[1, 4, 9, 16, 25, 36]
```

Generate a list that contains the factors of a given number

In [46]:
```
n = int(input())
fac = []
for i in range(1,n+1):
    if n%i==0:
        fac.append(i)
print(fac)
```

```
10
[1, 2, 5, 10]
```

In [47]:
```
n = int(input())
fac = [i for i in range(1,n+1) if n%i==0]
print(fac)
```

```
10
[1, 2, 5, 10]
```

In [1]:
```
names = ['Harry', 'Hermoine', 'Wesely', 'Malfoy', 'Dumbledore']
new_list = [i for i in names]
print(new_list)
```

```
['Harry', 'Hermoine', 'Wesely', 'Malfoy', 'Dumbledore']
```

In [2]:
```
names = ['Harry', 'Hermoine', 'Wesely', 'Malfoy', 'Dumbledore']
new_list = [len(i) for i in names]
print(new_list)
```

```
[5, 8, 6, 6, 10]
```

In [3]:
```
names = ['Harry', 'Hermoine', 'Wesely', 'Malfoy', 'Dumbledore']
new_list = [max(i) for i in names]
print(new_list)
```

```
['y', 'r', 'y', 'y', 'u']
```

In [ ]:
```

```