Day 4 : Data Structures - Lists

Working with Lists in Python

In [22]:
```
## creating a list
Name  = ["Kim", "Cheryl", "John", "Padma", "Berlin"]

Name.append("Krishna")  ## add item to end of the list
Name
```

Out[22]:
```
['Kim', 'Cheryl', 'John', 'Padma', 'Berlin', 'Krishna']
```

In [7]:
```
## add item to the end of the list
Name.append("Krishna")
print(Name)

Name.append("Padma")
print(Name)
```

```
['Kim', 'Cheryl', 'John', 'Padma', 'Berlin', 'Krishna']
['Kim', 'Cheryl', 'John', 'Padma', 'Berlin', 'Krishna', 'Padma']
```

In [14]:
```
## add each element of an iterable individually to the end of your list
Name = ["Kim", "Cheryl", "John"]

# Extend with another list
Name.extend(["Padma", "Berlin"])
print(Name)

Name = ["Kim", "Cheryl", "John"]
Name.append(["Padma", "Berlin"])
print(Name)
```

```
['Kim', 'Cheryl', 'John', 'Padma', 'Berlin']
['Kim', 'Cheryl', 'John', ['Padma', 'Berlin']]
```

In [17]:
```
## Insert an item at a given position
Name = ["Kim", "Cheryl", "John"]
Name.insert(1,"Sree")      ## at which index you want to put your name, (say), I wa
Name
```

Out[17]:
```
['Kim', 'Sree', 'Cheryl', 'John']
```

In [20]:
```
## Remove the first item from the list
Name = ["Kim", "Cheryl", "John"]

Name.extend(["Padma", "Berlin"])
print(Name)

Name.remove("Berlin")
Name
```

```
['Kim', 'Cheryl', 'John', 'Padma', 'Berlin']
```

Out[20]:
```
['Kim', 'Cheryl', 'John', 'Padma']
```

In [73]:
```
## Change List Items
Name = ["Kim", "Cheryl", "John"]
Name[0] = "Vishnu"
Name
```

Out[73]:
```
['Vishnu', 'Cheryl', 'John']
```

In [29]:
```
## Remove the item at the given position in the list, and return it
subjects = ["biology", "maths", "statistics", "chemistry", "physiscs"]

remove = subjects.pop(2)
print(subjects)

subjects.pop() ##removes the last item
subjects
```

```
['biology', 'maths', 'chemistry', 'physiscs']
```

Out[29]:
```
['biology', 'maths', 'chemistry']
```

In [32]:
```
## Remove all items from the list
subjects = ["biology", "maths", "statistics", "chemistry", "physiscs"]
subjects.clear()
subjects
```

Out[32]:
```
[]
```

In [39]:
```
## Return zero-based index in the list of the first item
numbers = [22,22,34,65,65,76,78,78,76]
number = numbers.index(78)
print(number)

numbers = ["22","22","34","65","65","76","78","78","76"]
number = numbers.index("78")
print(number)

numbers = ["22","22","34","65","78","65","76","78","76"]
number = numbers.index("78",5)   ### Find next 78 starting at position 5
print(number)
```

```
6
6
7
```

In [40]:
```
## Return the number of times items appear in the list.
subjects = ["biology", "maths", "statistics", "chemistry", "physiscs","maths", "sta
subjects.count("statistics")
```

Out[40]:
```
3
```

In [42]:
```
## Sort the items of the list
numbers = ["22","22","34","65","65","76","78","78","76"]
```

```
numbers.sort()
numbers
```

```
Out[42]: ['22', '22', '34', '65', '65', '76', '76', '78', '78']
```

```
In [45]: ## Reverse the elements of the list
fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
fruits.reverse()
fruits
```

```
Out[45]: ['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']
```

```
In [46]: ## Return a shallow copy of the list
fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
fruit = fruits.copy()
fruit
```

```
Out[46]: ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
```

List Comprehensions

```
In [56]: def squares():
    for x in range(1,11):
        print(x**2)

squares()

"""
you want the output to be in list
"""

def squares():
    return[x**2 for x in range(1,11)]

print(squares())

## or
squares = []
for x in range(1,11):
    squares.append(x**2)

print(squares)
```

```
1
4
9
16
25
36
49
64
81
100
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

In [78]:
```
## create a new list with the values doubled
num = [5,6,3,2,-4,-7,8,-1]
double = [x*2 for x in num]
print(double)

## filter the list to exclude negative numbers
remove = [ x for x in num if x > 0]
print(remove)

## apply a function to all the elements
fun = [float(x) for x in num]
print(fun)

## create a list of 2-tuples like (number, square)
tup = [(x , x**2) for x in num]
print(tup)

## filtering odd numbers from a list
odd_numbers = [num for num in range(1, 10) if num % 2 != 0 ]

print(odd_numbers)
```

```
[10, 12, 6, 4, -8, -14, 16, -2]
[5, 6, 3, 2, 8]
[5.0, 6.0, 3.0, 2.0, -4.0, -7.0, 8.0, -1.0]
[(5, 25), (6, 36), (3, 9), (2, 4), (-4, 16), (-7, 49), (8, 64), (-1, 1)]
[1, 3, 5, 7, 9]
```

In [72]:
```
## del statement
a = [-1, 1, 66.25, 333, 333, 1234.5]
del a[4]
print(a)

b = [-1, 1, 66.25, 333, 333, 1234.5]
del b[-4]
print(b)

b = [-1, 1, 66.25, 333, 333, 1234.5]
del b[0:3]
print(b)
```

```
[-1, 1, 66.25, 333, 1234.5]
[-1, 1, 333, 333, 1234.5]
[333, 333, 1234.5]
```

In [83]:
```
##  nested list comprehension
for i in range(1,6):
    for j in range (3,7):
        for k in range(2,5):
            result = i * j * k
            print(result)



"""
instead, we can use nested comprehension
"""
```

```
multiply = [[[i * j * k for k in range(2,5)] for j in range(3,7)] for i in range(1,
print(multiply)
```

```
6
9
12
8
12
16
10
15
20
12
18
24
12
18
24
16
24
32
20
30
40
24
36
48
18
27
36
24
36
48
30
45
60
36
54
72
24
36
48
32
48
64
40
60
80
48
72
96
30
45
60
40
60
80
50
75
```

```
100
60
90
120
[[[6, 9, 12], [8, 12, 16], [10, 15, 20], [12, 18, 24]], [[12, 18, 24], [16, 24, 32],
[20, 30, 40], [24, 36, 48]], [[18, 27, 36], [24, 36, 48], [30, 45, 60], [36, 54, 7
2]], [[24, 36, 48], [32, 48, 64], [40, 60, 80], [48, 72, 96]], [[30, 45, 60], [40, 6
0, 80], [50, 75, 100], [60, 90, 120]]]
```

Day 5 : Data Structures - Tuples

In [2]:
```
## creating tuples
t = "hello" , 23645 , 6785
t
```

Out[2]:
```
('hello', 23645, 6785)
```

In [4]:
```
## Tuples may be nested:
u = t, (1, 2, 3, 4, 5)
print(u)

v = t,u,(1, 2, 3, 4, 5)
v
```

```
(('hello', 23645, 6785), (1, 2, 3, 4, 5))
```

Out[4]:
```
(('hello', 23645, 6785),
 (('hello', 23645, 6785), (1, 2, 3, 4, 5)),
 (1, 2, 3, 4, 5))
```

In [5]:
```
## Tuples are immutable:
t[0] =99
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[5], line 2
      1 ## Tuples are immutable:
----> 2 t[0] =99

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

In [9]:
```
## Tuples can contain mutable objects
numbers = ([1, 2, 3], [3, 2, 1])
print(type(numbers))

numbers[1][0] = 99
numbers
```

```
<class 'tuple'>
```

Out[9]:
```
([1, 2, 3], [99, 2, 1])
```

In [13]:
```
## construct empty tuple and get the length
empty = ()
print(len(empty))
```

```
numbers = ([1, 2, 3], [3, 2, 1])
print(len(numbers))
```

```
0
2
```

In [18]:
```
## Accessing and Slicing
Name  = ("Kim", "Cheryl", "John", "Padma", "Berlin")
print(Name[3])
print(Name[-2])
print(Name[-4])
print(Name[2])

print(Name[1:4])
print(Name[0:3])
```

```
Padma
Padma
Cheryl
John
('Cheryl', 'John', 'Padma')
('Kim', 'Cheryl', 'John')
```

In [20]:
```
## Finding minimum & maximum from tuple
maths = (78,56,43,78,90)
print(max(maths))
print(min(maths))
```

```
90
43
```

In [22]:
```
## Combining two tuples
maths = (78,56,43,78,90)
stats = (98,45,32,78,89)
comb = maths+stats
comb
```

Out[22]:
```
(78, 56, 43, 78, 90, 98, 45, 32, 78, 89)
```

CODING EXERCISE

Working with Lists in Python

(a) Create a list and display its elements.

(b) Access elements using both positive and negative indexing.

(c) Apply slicing to extract specific portions of a list.

In [10]:
```
places = ["Hongasandra" , "Begur" , "BTM" , "HSR Layout" , "Bommanahalli"]
print(f"The list of places is: {places} \n")

print(f" The third place is: {places[2]} \n")
print(f" The fourth place is: {places[-1]} \n")
print(f"Slicing from index 1 to 3: {places[1:4]}\n")
```

```
The list of places is: ['Hongasandra', 'Begur', 'BTM', 'HSR Layout', 'Bommanahalli']

 The third place is: BTM

 The fourth place is: Bommanahalli

Slicing from index 1 to 3: ['Begur', 'BTM', 'HSR Layout']
```

(d) Iterate over a list using a loop and demonstrate list concatenation.

In [11]:
```
for place in places :
    print(f" -{place}")

add_more = ["Jayanagar" , "Kormangala"]
combined = places + add_more
print(f"\n{combined}\n")
```

```
 -Hongasandra
 -Begur
 -BTM
 -HSR Layout
 -Bommanahalli

['Hongasandra', 'Begur', 'BTM', 'HSR Layout', 'Bommanahalli', 'Jayanagar', 'Kormanga
la']
```

(e) Add elements to a list using the `append()`, `extend()`, and `insert()` methods.

(f) Remove elements from a list using `del`, `remove()`, and `pop()` functions.

In [12]:
```
app = places.append("Madiwala")
print(f"After append: {places}")

ext = places.extend(["Majestic","Shivajinagar"])
print(f"After extend: {places}")

ins = places.insert(0 , "Electronicity")
print(f"After insert: {places}")

rem = places.remove("BTM")
print(f"After removing: {places}")

pop = places.pop()
print(f"After poping last element: {places}")

pop = places.pop(1)
print(f"After poping first element: {places}")

del places[3]
print(f"After deleting 3rd place: {places}")
```

```
After append: ['Hongasandra', 'Begur', 'BTM', 'HSR Layout', 'Bommanahalli', 'Madiwal
a']
After extend: ['Hongasandra', 'Begur', 'BTM', 'HSR Layout', 'Bommanahalli', 'Madiwal
a', 'Majestic', 'Shivajinagar']
After insert: ['Electronicity', 'Hongasandra', 'Begur', 'BTM', 'HSR Layout', 'Bomman
ahalli', 'Madiwala', 'Majestic', 'Shivajinagar']
After removing: ['Electronicity', 'Hongasandra', 'Begur', 'HSR Layout', 'Bommanahall
i', 'Madiwala', 'Majestic', 'Shivajinagar']
After poping last element: ['Electronicity', 'Hongasandra', 'Begur', 'HSR Layout',
'Bommanahalli', 'Madiwala', 'Majestic']
After poping first element: ['Electronicity', 'Begur', 'HSR Layout', 'Bommanahalli',
'Madiwala', 'Majestic']
After deleting 3rd place: ['Electronicity', 'Begur', 'HSR Layout', 'Madiwala', 'Maje
stic']
```

Manipulating Tuples in Python

(a) Create and print tuples containing various types of data.

(b) Access tuple elements using both positive and negative indices.

(c) Slice tuples to extract subsets of elements.

In [17]:
```python
tuples = ("Hello" , "Cheryl" , False , 3.14 ,200)
print(f"tuples containing various types of data {tuples} \n")

print(f" The third element is: {tuples[2]} \n")
print(f" The fourth element is: {tuples[-1]} \n")

print(f"Sliced tuple: {tuples[1:4]}\n")
```

```
tuples containing various types of data ('Hello', 'Cheryl', False, 3.14, 200)

 The third element is: False

 The fourth element is: 200

Sliced tuple: ('Cheryl', False, 3.14)
```

(d) Iterate through a tuple and perform membership testing using the `in` operator.

In [21]:
```python
for element in tuples:
    print(f"- {element}")

is_present = "Cheryl" in tuples
is_absent = "Vishnu" in tuples

print(f"Is 'Cheryl' in the tuple? {is_present}")
print(f"Is 'Vishnu' in the tuple? {is_absent}")
```

```
- Hello
- Cheryl
- False
- 3.14
- 200
Is 'Cheryl' in the tuple? True
Is 'Vishnu' in the tuple? False
```

(e) Concatenate two or more tuples and display the result.

```
In [22]: tuple1=(1, 2, 3)
         tuple2=('a', 'b', 'c')
         print(f"Concatenated tuple: {tuple1+tuple2}")
```

```
Concatenated tuple: (1, 2, 3, 'a', 'b', 'c')
```

(f) Convert a list into a tuple.

```
In [23]: lists = ['HI', 'how', 'are', 'you', 'minto']
         print(f"The list {lists}")

         tuples = tuple(lists)
         print(f"The tuple {tuples}")
```

```
The list ['HI', 'how', 'are', 'you', 'minto']
The tuple ('HI', 'how', 'are', 'you', 'minto')
```

(g) Utilize the `sorted()`, `count()`, and `index()` functions to manipulate and analyze tuples.

```
In [28]: tuples = (144, 953, 322, 455, 769, 769, 445, 679, 500)
         print(f"The sorted tuple : {sorted(tuples)}")
         print(f"The count of the tuple : {tuples.count(769)}")
         print(f"The number 322 is first found at index: {tuples.index(322)}")
```

```
The sorted tuple : [144, 322, 445, 455, 500, 679, 769, 769, 953]
The count of the tuple : 2
The number 322 is first found at index: 2
```