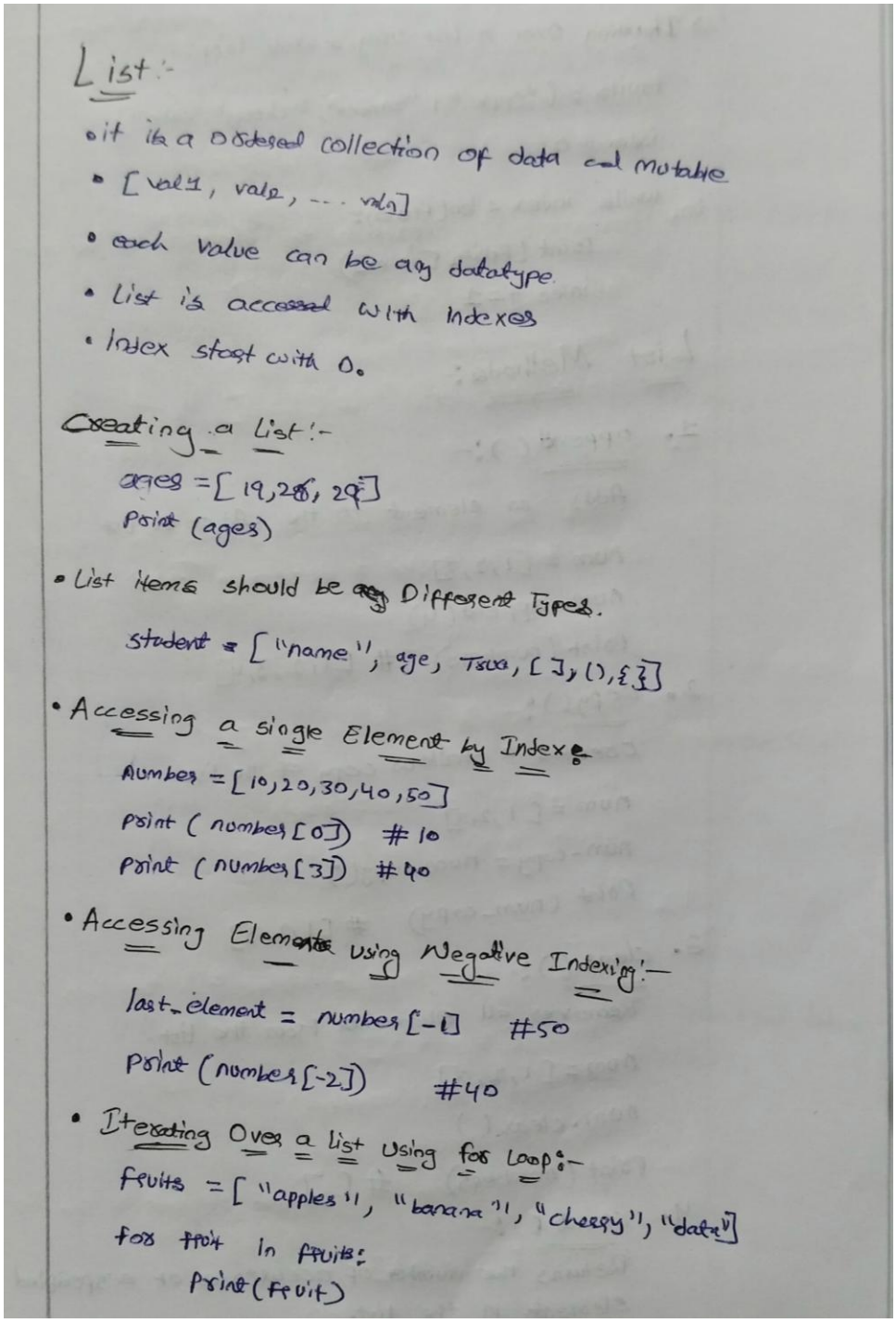
List Methods



List :-

- it is a Ordered collection of data and mutable
- [val1, val2, ... valn]
- each value can be any datatype
- List is accessed with indexes
- Index start with 0.

Creating a List :-

```
ages = [19, 28, 29]
print (ages)
```

- List items should be Different Types.

```
student = ["name", age, True, [], (), {}]
```

- Accessing a single Element by Index :-

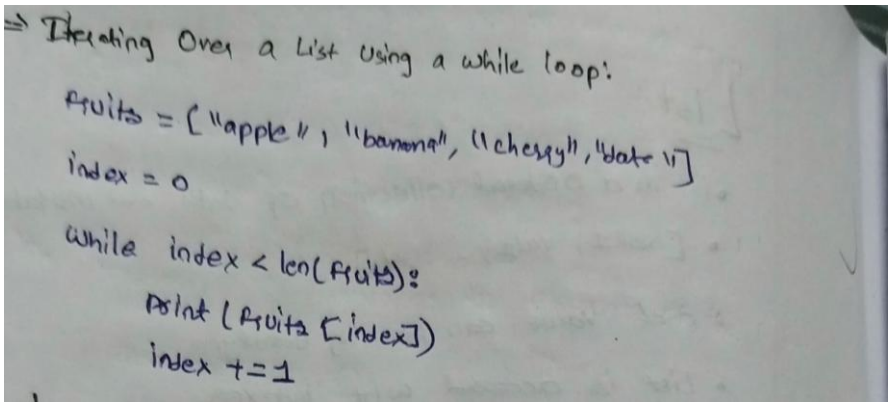
```
numbers = [10, 20, 30, 40, 50]
print (numbers[0])   # 10
print (numbers[3])   # 40
```

- Accessing Elements using Negative Indexing :-

```
last_element = numbers[-1]   #50
print (numbers[-2])          #40
```

- Iterating Over a list using for loop :-

```
fruits = ["apples", "banana", "cherry", "date"]
for fruit in fruits:
    print(fruit)
```



⇒ Iterating Over a List using a while loop :

```
fruits = ["apple", "banana", "cherry", "date"]
index = 0

while index < len(fruits):
    print (fruits[index])
    index += 1
```

List Methods:-

1. append():-

Adds an element to the end of the list

```
num = [1,2,3]
num.append(4)
print(numbers)  # [1,2,3,4]
```

2. copy():-

Creates a shallow copy of the list.

```
num = [1,2,3]
num_copy = num.copy()
print(num_copy)  # [1,2,3]
```

3. clear():-

Removes all elements from the list.

```
num = [1,2,3]
num.clear()
print(numbers)  # []
```

4. count():-

Returns the number of occurences of a specified element in the list.

```
num = [1,2,2,3]
count_of_two = num.count(2)
print(count_of_two)  #2
```

5. Extend() :-

Adds elements of an iterable (like another list) to the end of the list.

```
num = [1,2,3]
num.extend([4,5])
print(numbers)  # [1,2,3,4,5]
```

6. index() :-

Returns the index of the first occurrence of a specified element.

```
num = [1,2,3,2]
index_of_two = numbers.index(2)
print(index_of_two)  #1
```

7. Insert() :-

Inserts an element at a specific position in the list

```
num = [1,2,3]
num.insert(2,3)
print(num)  # [1,2,3,4]
```

8. pop() :-

Removes and returns the element at the specified index (or the last element if no index is specified).

```
num = [1,2,3]
last_ele = num.pop()
Print(last_ele)  #3
Print(num)  # [1,2]
```

9. remove( ) :- removes the first occurrence of a specified element.

```
Num = [1,2,3,2]
num.remove(2)
Print (num)        # [1,3,2]
```

10. reverse( ) :- Reverses the elements of the list.

```
num = [1,2,3]
num.reverse( )
Print (numbers)   # [3,2,1]
```

11. Sort( ) :-

sorts the list in ascending order by default (can also be sorted in descending order)

```
num = [3,1,4,2]
num = sort( )
Print (num)        # [1,2,3,4]

num.sort (reverse=True)
Print (num)        # [4,3,2,1]
```

12. min( ) :-

Returns the smallest element in the list.

```
num = [3,1,4,2]
minimum = min (numbers)
Print (minimum)   #1
```

13. Max( ) :- Returns the largest element in the list

```
num = [3,1,4,2]
maximum = max (numbers)
Print (maximum)   #4
```

List Comprehension:-

List comprehension provides a concise way to create lists in python. It's a shorthand for looping through an iterable and applying an expression to each item.

Example:- Creates a list of squares.

```
squares = [x**2 for x in range(5)]
print(squares)  #[0,1,4,9,16]
```

Example:- Create a list of even numbers.

```
even = [x for x in range(10) if x%2==0]
print(evens)  #[0,2,4,6,8]
```

Nested Lists:-

A nested list is a list that contains other lists as its elements. You can access element of a nested list using indexing.

Example:- simple nested list

```
matrix = [
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]
    ]
print(matrix[1][2])  #6
```

Example:- Iterates through a nested list

```
for row in matrix:
    for element in row:
        Print(element, end=" ")
    Print()
```

O/P:-

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$