Date 3/12/21

CHAPTER → 5

Sequence Data Type : LIST

Introduction :-

एक वेरिएबल में एक से अधिक value स्टोर करने के लिए का प्रयोग किया जाता है। Sequence data type

यह तीन प्रकार का होता है -

1. List
2. Tupple
3. Dictionary

(1.) **LIST :-** बड़ी मात्रा में डेटा स्टोर करने के लिए इसका प्रयोग होता है।
इसमें किसी भी प्रकार का डेटा स्टोर किया जा सकता है।
LIST को Square bracket के अंदर लिखा जाता है तथा डेटा को comma के द्वारा अलग किया जाता है।

```
[ ]                  → Empty List
[1, 2, 3, 4]         → List of Integers
[1.2, 2.3, 3.4]      → "  "  floating point
['x', 'y', 'z']      → "  "  characters

["Delhi", "Meerut", "Baraut"] → List of string
[1, "Rajesh", 12.5, 2, "Rohan", 15.6] → "  "  Mixed Value
```

NOTE : लिस्ट mutable Sequence data type है। यानी लिस्ट के डेटा को बदला जा सकता है। (edit)

Creating a List :-

पायथन में तीन प्रकार से लिस्ट बनायी जाती है।

- By Assigning Value
- By other Sequence
- Dynamically via keyboard

(A.) By Assigning Value :-

असाइनमेंट ऑपरेटर (=) का प्रयोग करके लिस्ट बनायी जाती है।

Syntax : <list name> = [<comma Separated Value>]
Example : marks = [28, 58, 92, 29, 55]

(B.) Creating List by other Sequence :-

किसी अन्य Sequence जैसे- String या Tuple का प्रयोग करके list बनायी जा सकती है।

इसमे list बनाने के लिए list function का प्रयोग किया जाता है।

Syntax : <list name> = list (<any Sequence>)
Example : grade = list ("ABCD")
print (grade)
['A', 'B', 'C', 'D']

(C.) Creating list dynamically via keyboard :-

eval फंक्शन के अंदर input Statement का प्रयोग करके dynamic list बनायी जाती है।

इनपुट फंक्शन इनपुट के रूप में String प्राप्त करता है तथा

eval function इस String को list element के रूप में convert कर देता है।

Syntax : <variable> = eval (input ("< any message >"))

Example : n = eval (input ("Enter three number"))
mark = list (n)
print (mark)

Accessing the List :—

दो तरीको से लिस्ट की जाती है access -

1. Whole list at once
2. Individual element one by one

1. Whole List at once :—

लिस्ट के नाम का प्रयोग करके पूरी लिस्ट एक बार में एक्सेस की जाती है।

Syntax : print (<list name>)
Example : marks = [48, 34, 27, 31, 22]
print (marks)
Output : [48, 34, 27, 31, 22]

print (* marks)
Output : 48 34 27 31 22

2. Individual element one by one :—

लिस्ट के प्रत्येक element को Index no. का प्रयोग करके access किया जा सकता है।

जब लिस्ट बनायी जाती है तब लिस्ट के प्रत्येक element को एक unique no. दिया जाता है जो Index no. कहलाता है। यह 0 से Start होता है।

Forward Index → 0 1 2 3

[28 , 29 , 31 , 51]

-4 -3 -2 -1 ← Backward Index

Syntax : <list name> [< index no. of element >]

Example:
```
mark = [ 28 , 29 , 31 , 51 ]
print (Mark [1])
```

Traversing the list :

लिस्ट के प्रत्येक element को access करने के लिए for loop में list के नाम का प्रयोग किया जाता है। यह process traversing है।

Syntax :
```
for < control variable > in < list name > :
    body of loop
```

Example:
```
l = [ 10, 20 , 30 , 40 , 50 ]

for a in l :
    print ( a, end = " " )
```

NOTE : उसमें Index no. का प्रयोग नही होता।

List Operations :— लिस्ट ऑपरेशन कई प्रकार के होते है —

1. **Combining the list :-**

Concatenation Operator ('+') का प्रयोग करके दो या अधिक लिस्ट को जोड़ा जा सकता है।

Syntax : <list 1> + <list 2>
Example : print (mark1 + mark 2)

2. **Replicating the list :-**

'*' Operator का प्रयोग करके दी गई लिस्ट को दोहराया जा सकता है।

Syntax : <list name> * <integer>
Example : print (list 1 * 3)

3. **Slicing the list :-**

लिस्ट को कई भागो में बाँटने की प्रक्रिया है। इसके लिए Index no. का प्रयोग किया जाता है। Slicing

Syntax : <list name> [Start : Stop : Step]

Start → Starting Value (default Value 0)
Stop → ending Value (Slicing Stop)
Step → difference Value (default Value 1)

Example : num = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
print (num [1, 7, 2])
Output → [20, 40, 60]

Program: निम्न प्रोग्राम दी गई लिस्ट को Slicing करने की क्रिया को प्रदर्शित करेगा।

```
num = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
print (num [1, 7, 2]
```

Output → [20, 40, 60]

num [2:6] → 30, 40, 50, 60
num [4:] → 50, 60, 70, 80, 90, 100
num [:6] → 10, 20, 30, 40, 50, 60
num [::] → 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

4. Adding elements to list :-

पहले से बनी हुई लिस्ट में append function के द्वारा नये element जोड़े जा सकते हैं।

Syntax <List name>.append <element>
Example
```
mark.append (75)
print (mark)
```

Program: 10 संख्याओं की एक लिस्ट बनाने का प्रोग्राम बनाएं जिसमें नम्बर यूजर एंटर करे।

```
num = [ ]
for a in range (10):
    n = int (input ("Enter a no.", a+1)
        num.append (n)
print ("List is ", num)
```

5. **Updating the Element :—**

लिस्ट एक mutable data type है जिस कारण उसे अपडेट किया जा सकता है. delete किया जा सकता है या बदला जा सकता है।

Syntax : <list name>[<index no.>] = <new element>

Example : num[2] = 55

Program : 5 नामों की लिस्ट बनाने के लिए एक प्रोग्राम लिखें। प्रोग्राम यूजर को नए नाम के साथ एलिमेंट को अपडेट करने के लिए एक नया नाम और सूचकांक स्थिति इनपुट करने के लिए कहना चाहिए

```
name = ['Aditi', 'Bhawna', 'chetna', 'Dipika', 'kanak']
print (name)
n = int (input ("Enter the name position that will be
nm = input ("Enter new name")
num [n-1] = nm
print ("List after Update:", num)
```

6. **Deleting Elements :—**

'del' स्टेटमेंट का प्रयोग करके individual element delete किया जा सकता है।

Syntax-1 : del <list name>[<index>]
Example : del mark[5]

Syntax-2 : del <list name>[<start>:<stop>]
Example : del mark[3:6]

7. Copying the List :-

तीन प्रकार से एक लिस्ट को अन्य लिस्ट में कॉपी किया जा सकता है।

1. Assignment method
2. copy() function
3. list() function

(1.) Assignment Method -

Assignment operator (=) का प्रयोग करके लिस्ट कॉपी की जाती है।

Syntax: <second list name> = <first list name>

Example:
```
A = [12, 13, 14, 15]
B = A
print(A)
print(B)
```
Output → [12, 13, 14, 15]
[12, 13, 14, 15]

NOTE → Assignment Operator के द्वारा वास्तव में लिस्ट कॉपी नही की जाती। बल्कि इस लिस्ट को दूसरा नाम दिया जाता है।

(2.) Copy() function - इस function के द्वारा वास्तव में एक लिस्ट अन्य लिस्ट में कॉपी की जाती है। कॉपी करने के बाद दोनो लिस्ट अलग-अलग होती है।

Syntax: <second list name> = <first list name>.copy()

Example:
```
A = [11, 22, 33, 44]
B = A.Copy()
print(A)
print(B)
```

3. **list() function** —

यह function भी एक लिस्ट को अन्य लिस्ट में कॉपी करता है तथा दोनो लिस्ट अलग-2 होती है।

Syntax <second list name> = list(<first list name>)

Functions of List :-

1. **लेन (len())** —

लिस्ट की लम्बाई ज्ञात करने के लिए इसका प्रयोग होता है। लिस्ट की ल० लिस्ट में element की संख्या है।

Syntax: len(<list name>)

Example: A = [1, 2, 3, 4, 9, 10]
या print(a.len())
print(len(a))

2. ~~अपेंड~~ **एक्सटेंड (extend())** —

यह फंक्शन दो या अधिक लिस्ट को जोड़ने के लिए प्रयोग किया जाता है। दूसरी लिस्ट पहली लिस्ट के अंत में जोड़ी जाती है।

Syntax: <list 1>.extend(<list 2>)

Example: A = [11, 22, 33]
B = [44, 55, 66]
A.extend(B)
print A

Output: [11, 22, 33, 44, 55, 66]

3. इन्सर्ट (insert()) –

इस फंक्शन का प्रयोग एक element को लिस्ट में जोड़ने के लिए इसका प्रयोग होता है।
यह फंक्शन लिस्ट में किसी भी स्थान पर element को जोड़ सकता है।

Syntax: <list name>.insert(<position>,<element>)
Example: A = [11, 22, 44, 55]
A.insert(2, 33)
Output: [11, 22, 33, 44, 55]

4. पॉप (pop()) –

इस फंक्शन के द्वारा elements delete किये जा सकते हैं।
Element delete करने के लिए index no. का प्रयोग होता है।
यदि index no लिस्ट में नही है तब यह अंतिम element delete करता है।
Delete करने के बाद यह लिस्ट को प्रिन्ट भी कर देता है।

Syntax: <list name>.pop(<index>)
Example: a = [11, 12, 10, 9, 8]
a.pop(2)
Output [11, 12, 9, 8]

5. रिमूव (remove()) –

लिस्ट से element delete करने के लिए इसका प्रयोग होता है।
Element delete करने के लिए यह index no. की जगह element का ही प्रयोग करता है।

Syntax: <list name>.remove(<element>)
Example: A = [9, 10, 11, 10, 8, 12]
A.remove(10)
print(A)
Output: [9, 11, 10, 8, 12]

NOTE → यह element की पहली उपस्थिति को delete करता है।

6. रिवर्स (reverse()) —

यह फंक्शन लिस्ट के element को उल्टे क्रम में व्यवस्थित कर देता है।

Syntax: <list name>.reverse()
Example: Vowel = ['a', 'i', 'e', 'u', 'o']
print(vowel.reverse())
Output: ['o', 'u', 'e', 'i', 'a']

7. सॉर्ट (Sort()) —

इस फंक्शन के द्वारा लिस्ट के element को ascending order में व्यवस्थित किया जाता है।

Syntax: <list name>.sort()
Example: a = [2, 9, 8, 1, 5, 4]
a.sort()
print(a)
Output [1, 2, 4, 5, 8, 9]

For descending —

```
a.sort(reverse=True)
print(a)
```

Output : [9, 8, 5, 4, 2, 1]