List Manipulation

सीबीएसई पाठ्यक्रम पर आधारित

कक्षा -11

अध्याय - 7

द्वारा:

संजीव भदौरिया

स्नातकोत्तर शिक्षक (संगणक विज्ञान)

के० वि० बाराबंकी (लखनऊ संभाग)

परिचय

- Python में List एक प्रकार का पात्र (container) होता है जिसमे किसी भी प्रकार का मानो (values) की सूची रखी जा सकती है |
- List एक का परिवर्तनीय(mutable) data type होता है अर्थात आप list के किसी भी value को बदल सकते हैं, बदलाव के कारण python दूसरी list नहीं बनाता है |
- List एक प्रकार से string और tuple जैसी ही sequence होती है बस सिर्फ अंतर इतना होता है की list की values बदली जा सकती हैं (mutable) और tuple या string की नहीं (immutable)|
- इस अध्याय में हम list के manipulation के बारे में पढेंगे, जिसमे हम list बनाना, उसको प्रयोग करना और list पर कुछ क्रियाएं (operation) built in functions के द्वारा सीखेंगे|

List को बनाना

- List पाइथन का एक मानक (standard) data type है | यह एक ऐसा sequence है जो किसी भी प्रकार के डाटा की सूची store कर सकता है |
- List को पाइथन में व्यक्त करने के लिए square brackets " [ ] " का प्रयोग करते हैं | उदहारण के तौर पर -
  - [ ] यह रिक्त list है (Empty list)
  - [1, 2, 3] यह integers की list है
  - [1, 2.5, 5.6, 9] यह numbers की list है (integer और float)
  - ['a', 'b', 'c'] यह characters की list है |
  - ['a', 1, 'b', 3.5, 'zero'] यह mixed values की list है |
  - ['one', 'two', 'three'] यह string की list है
- एक बार फिर से यह दोहरा लेते हैं कि python में list और dictionary mutable और बाकी समस्त data type immutable होते हैं |

List को बनाना

- List को बनाने के लिए निम्न तरीके हैं -
  - Empty list बनाने के लिए -

    L = [ ]
  - list बनाने के लिए आप निम्न statement का भी रयोग कर सकते हैं -

    L = list( )

```
>>> Mylist = list('hello')
>>> Mylist
['h', 'e', 'l', 'l', 'o']
>>>
```

```
>>> L = ('p','a','n','k','j')
>>> NewList = list(L)
>>> NewList
['p', 'a', 'n', 'k', 'j']
>>>
```

ये tuple है

  - Long lists बनाने के लिए -

    even = [0, 2, 4, 6, 8, 10 ,12 ,14 ,16 ,18 ,20]
  - Nested list बनाने के लिए -

    L = [3, 4, [5, 6], 7]

एक तरीका यह भी हो सकता है

```
>>> L1 = list(input("Enter List Elements"))
Enter List Elements12345
>>> L1
['1', '2', '3', '4', '5']
```

List को बनाना

– **जैसा की हमने इस उदहारण में देखा कि list में भले ही हमने नंबर में मान दिया हो परन्तु list में मान string के रूप में ही गया है**

```
>>> L1 = list(input("Enter List Elements"))
Enter List Elements12345
>>> L1
['1', '2', '3', '4', '5']
```

– यदि हमें numeric रूप में ही value को input करना है तो list के साथ निम्न function लगाना पड़ेगा -

eval(input())

L=eval(input("Enter list to be added "))

eval ( ) function किसी string को पास करने पर उसके type को identify करके return करता है |

```
>>> a="15"
>>> b="25"
>>> print(a+b)
1525
>>> print(eval(a)+eval(b))
40
```

String Values

एक अन्य उदहारण देखें

```
>>> a="15"
>>> b=eval(a)
>>> type(a)
<class 'str'>
>>> type(b)
<class 'int'>
>>>
```

List को Access करना

- List को access करना सीखने से पहले हम list और string में समानता देख लें|
- List एक प्रकार से string के जैसी ही sequence होती है |
- List भी अपने प्रत्येक अवयव (Element) का index बनाती है |
- String के जैसे इसमें भी 2 index होती है एक forward index (0, 1, 2, 3, ….n-1 तक) और एक backward index (-n से -1 तक) |

Forward index	0	1	2	3	4	5	6	7	8	9	10	11	12	13
List	R	E	S	P	O	N	S	I	B	I	L	I	T	Y
Backward index	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- इसमें भी आप string के जैसे ही values को access कर सकते हैं |

```
>>> vowels=['a','e','i','o','u']
>>> vowels[4]
'u'
>>> vowels[-5]
'a'
>>> vowels[-1]
'u'
```

List को Access करना

- List की लम्बाई (Length) को पता करने के लिए len( ) function का प्रयोग करते हैं |

```
>>> name=list("Pankaj")
>>> name
['P', 'a', 'n', 'k', 'a', 'j']
>>> len(name)
6
>>>
```

Important 1: List के साथ membership operator (*in, not in*) दोनों कार्य ठीक वैसे ही करते हैं जैसे की अन्य sequences में|

- L[i] यह i index पर जो भी वैल्यू है उसको return करता है
- L [i : j] एक नयी list return करता है जिसमे L के i index और j index के मध्य की समस्त values होती हैं इसमें j index की वैल्यू शामिल है |

```
>>> name=list("Pankaj")
>>> name[3]
'k'
>>> nm=name[2:4]
>>> nm
['n', 'k']
```

Important 2: + operator किसी list को अन्य list के आखिरी में जोडती है जबकि * किसी भी list को repeat करता है |

List और String में अंतर

- List और string में बस यही एक basic difference है की string immutable है और list mutable|
- String की individual values को बदला नहीं जा सकता जबकि list में ऐसा

```
>>> string="aeiou"
>>> string[2]
'i'
>>> string[2]='I'
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    string[2]='I'
TypeError: 'str' object does not support item assignment
```

String में value नहीं बदली | error आगई |

List में value बदल गयी | मतलब साफ़ है की list mutable होती है |

```
>>> st=list("aeiou")
>>> st
['a', 'e', 'i', 'o', 'u']
>>> st[2]='I'
>>> st
['a', 'e', 'I', 'o', 'u']
>>>
```

List को travers करना

- List को travers करने का तात्पर्य उसके प्रत्येक अवयव(element) को access करना और उनको process करना है |
- List का traversal को अत्यंत आसान बनाता है for loop –

```
for <item> in <list>:
    # प्रत्येक item को यहाँ process करें
```

```
L=['P','Y','T','H','O','N']
for a in L:
    print(a)
```

```
 RESTART: C:/
P
Y
T
H
O
N
```

```
L=['P','Y','T','H','O','N']
length=len(L)
for a in range(length):
    print("Index ",a," और ",(a-length)," पर आइटम है : ",L[a])
```

```
 RESTART: C:/Users/KVBBKServer/
Index  0  और  -6  पर आइटम है :  P
Index  1  और  -5  पर आइटम है :  Y
Index  2  और  -4  पर आइटम है :  T
Index  3  और  -3  पर आइटम है :  H
Index  4  और  -2  पर आइटम है :  O
Index  5  और  -1  पर आइटम है :  N
```

Python UNICODE को समर्थन करता है अतः हिंदी में भी output संभव है |

List की तुलना (Compare) करना

- दो अलग अलग list की तुलना relational operator के द्वारा की जा सकती है|
- Python आतंरिक रूप से शब्दकोशीय क्रम (lexicographical order) में list या tuple की तुलना करता है |
- इसका मतलब यह है की तुलना किये जाने वाले sequence समान प्रकार के हों तथा उनके प्रत्येक अवयव(element) क्रमशः एक दूसरे के सामान हों |

```
>>> L1,L2=[1,2,3],[1,2,3]
>>> L3=[1,[2,3]]
>>> L1==L2
True
>>> L1==L3
False
```

कुछ उदाहरण

```
>>> [1,2,8,9]<[9,1]
True
>>> [1,2,8,9]<[1,2,9,1]
True
>>> [1,2,18,9]<[1,2,9,10]
False
```

```
>>> L1,L2=[1,2,3],[1,2,3]
>>> L3=[1,[2,3]]
>>> L1<L2
False
>>> L1<L3
Traceback (most recent call last):
  File "<pyshell#17>", line 1, in <module>
    L1<L3
TypeError: '<' not supported between instances of 'int' and 'list'
```

- पहली तुलना में पाइथन ने error नहीं दी क्योंकि दोनों list समान हैं |
- जबकि दूसरी तुलना में दोनों list के तुलनात्मक मान नहीं हैं | इसलिए python ने error दे दी|

List Operations (+, *)

- List पर होने वाले प्रमुख operation हैं joining list, replicating list और list की slicing(टुकड़े) |
- List को ज्वाइन करने के लिए ***+ operator*** का प्रयोग किया जाता है जी को पहली list के आखिरी में दूसरी list को जोड़ देता है | + operator के साथ दोनों operands को list type का होना चाहिए, अन्यथा error आयेगी |

```
>>> L1=[1,2,3]
>>> L2=[4,5,6,7]
>>> L3=L1+L2
>>> L3
[1, 2, 3, 4, 5, 6, 7]
```

- List को replicate करने के लिए **** operator*** का प्रयोग किया जाता है |

```
>>> L1=[1,2,3]
>>> L2=L1*3
>>> L2
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

List Operations (Slicing)

- List को slice करने के लिए syntax है - seq = list [start : stop]

```
>>> LST=[10,12,14,20,22,24,30,32,34]
>>> SEQ=LST[3:-3]
>>> SEQ
[20, 22, 24]
>>> SEQ=LST[2:4]
>>> SEQ
[14, 20]
```

- List को slice करने के लिए एक और syntax है - seq=list[start:stop:step]

```
>>> LST=[10,12,14,20,22,24,30,32,34]
>>> SEQ=LST[0:10:2]
>>> SEQ
[10, 14, 22, 30, 34]
>>> LST[2:10:3]
[14, 24, 34]
>>> LST[::3]
[10, 20, 30]
```

```
>>> LST[::-1]
[34, 32, 30, 24, 22, 20, 14, 12, 10]
```

Slice का प्रयोग list Modification के लिए

- निम्नलिखित उदाहरणों पर ध्यान केन्द्रित करिए |

```
>>> L=["one","two","three"]
>>> L
['one', 'two', 'three']
>>> L[0:2]=[0,1]
>>> L
[0, 1, 'three']
>>> L=["one","two","three"]
>>> L[0:2]="a"
>>> L
['a', 'three']
```

यहाँ नयी values assign की जा रही हैं|

यहाँ भी नयी values assign की जा रही हैं|

दोनों के परिणामों में अंतर को परखिये |

```
>>> l=[1,2,3]
>>> l[2:]="604"
>>> l
[1, 2, '6', '0', '4']
>>> l[2:]=144
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    l[2:]=144
TypeError: can only assign an iterable
```

144 एक संख्या है न की sequence

List के साथ कार्य करना

- List में elements को जोड़ना (Appending) – list.append(item)

```
>>> L=[10,12,14]
>>> L.append(16)
>>> L
[10, 12, 14, 16]
```

- List में elements को सुधारना (updating) – list[index]=<new value>

```
>>> L=[10,12,14,30]
>>> L[2]=24
>>> L
[10, 12, 24, 30]
```

- List में elements को delete करना – del list[index]

Important: del कमांड से हम एक element अथवा एक slice अथवा सम्पूर्ण list को डिलीट कर सकते हैं|

```
>>> L=[10,12,14,30]
>>> del L[2]
>>> L
[10, 12, 30]
```

Important: यदि आप del list करते हैं तो सम्पूर्ण list ही Delete हो जाएगी|

List के साथ कार्य करना

- List से pop ( ) करने पर सिर्फ एक element ही डिलीट होगा |
- pop ( ) फंक्शन से slice delete नहीं हो सकती |
- pop ( ) फंक्शन delete किये जाने वाली value कोप return भी करता है |

list.pop(<index>)

```
>>> L=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]
>>> L.pop()
17
>>> L.pop(5)
6
>>> L.pop(0)
1
```

Last item

6th item

1st item

List Functions और Methods

– List manipulation के लिए पाइथन कुछ built-in फंक्शन भी प्रदान करता है | list के फंक्शन को कॉल करने का प्रारूप निम्न है - <list-object>.<method-name>

Function	Details
List.index(<item>)	यह पास किये गए आइटम का index return करता है |
List.append(<item>)	यह पास किये item को list के आखिरी में जोड़ता है |
List.extend(<list>)	यह एक list को argument के रूप में लेके सम्बंधित list(जिसके साथ फंक्शन कॉल किया गया है) के आखिरी में पूरी list जुड़ जाती है
List.insert(<pos>,<item>)	यह फंक्शन दिए गए स्थान (pos) पर दिया गया item insert कर देता है |
List.pop(<index>)	यह दिए गए index वाले item को delete करके return करता है | इसमें index को पास करना वैकल्पिक होता है | यदि इसे पास न करें तो लास्ट item pop होगा |
List.remove(<value>)	यह केवल दिए गए value को delete करेगा वह भी जो list में पहले मिलेगा, लेकिन delete की गयी value को return नहीं करेगा |

List Functions और Methods

Function	Details
List.clear ( )	यह list के सारी values को remove कर देता है | और empty list बना देता है |
List.count (<item>)	यह दिए गए item को count करके return करता है कि list में पास किया गया item कितने बार आया है |
List.reverse ( )	यह फंक्शन list को reverse आर्डर में set कर देता है | यह कोई नयी list नहीं बनता है |
List.sort ( )	यह फंक्शन list को बढ़ते क्रम से set कर देता है यदि हमें घटते क्रम में list को set करना है तो list.sort(reverse =True) लिखना होगा|

List Functions और Methods

- List.index ( ) function:

```
>>> lst=[13,18,11,16,18,14]
>>> lst.index(18)
1
```

- List.append( ) function:

```
>>> lst=[13,18,11,16,18,14]
>>> lst.append(27)
>>> lst
[13, 18, 11, 16, 18, 14, 27]
```

- List.extend( ) function:

```
>>> lst=[13,18,11,16,18,14]
>>> lst1=[67,78,89]
>>> lst.extend(lst1)
>>> lst
[13, 18, 11, 16, 18, 14, 67, 78, 89]
```

- List.insert( ) function:

```
>>> t1=['a','e','u']
>>> t1.insert(2,'i')
>>> t1
['a', 'e', 'i', 'u']
```

List Functions और Methods

- List.pop ( ) function:

```
>>> lst=[13,18,11,16,18,14]
>>> lst.pop()
14
>>> lst.pop(2)
11
>>> lst
[13, 18, 16, 18]
```

- List.remove( ) function:

```
>>> lst=[13,18,11,16,18,14]
>>> lst.remove(18)
>>> lst
[13, 11, 16, 18, 14]
```

```
>>> lst=[2,3,4,5]
>>> lst
[2, 3, 4, 5]
>>> lst.clear()
>>> lst
[]
```

- List.count( ) function:

```
>>> lst=["one","two","three","three","four"]
>>> lst.count("three")
2
```

List Functions और Methods

- List.reverse( ) function:

```
>>> lst=["one","two","three",4,5]
>>> lst.reverse()
>>> lst
[5, 4, 'three', 'two', 'one']
```

- List.sort( ) function:

```
>>> t1=['e','i','q','p','a','u','o','r']
>>> t1.sort()
>>> t1
['a', 'e', 'i', 'o', 'p', 'q', 'r', 'u']
```

Important

- यदि आपको reverse आर्डर में sort करना हो तो निम्न फंक्शन लिखते हैं – **List.sort(reverse=True)**
- यदि किसी list में complex number दिया हुआ है तो sort कार्य नहीं करेगा |

और अधिक पाठ्य-सामग्री हेतु निम्न लिंक पर क्लिक करें -

www.pythontrends.wordpress.com