06-Python Basic

Python list data types

Posted on September 10, 2021
Last updated on February 1, 2023

6 List Data Structure

List is used when we want to represent a `group of values` as a single entity where `insertion order` required to preserve and `duplicates are allowed` then we should go for list data type

- List is one of the `most frequently used` and very versatile data types used in Python.
- List items are `ordered`, `mutable`, `heterogeneous` and `allow duplicate values`.
- List items are `indexed`, the first item has index `[0]`, the second item has index `[1]` etc.

6.1 List declaration

```
# Def empty list
p = list()
p = []

nlist = [10, 10.5, 'ap', True, 20]
```

6.2 List access

- Acess single element

```
# Access using index
 -9 -8 -7 -6 -5 -4 -3 -2 -1
[ 1, 2, 3, 4, 5, 6, 7, 8, 9 ]
  0  1  2  3  4  5  6  7  8

nlist = [10, 10.5, 'ap', True, 20]
nlist[0]  # 10
nlist[-1]  # 20
```

- Access n-element
    - Syntax: `list[start: end: step]`

```
nlist = [10, 10.5, 'ap', True, 20, 30, 40, 50]

# 0th to 4th index
nlist[0: 5]
nlist[:5]
# [10, 10.5, 'ap', True, 20]

# 4th to end index
nlist[4:]
# [20, 30, 40, 50]

# Last 3 element
nlist[-3:]
# [30, 40, 50]

# Nested Lists
plist = [[10, 20, 30], [40, 50, 60], [70, 80, 90]]
print(plist[0][1])  # 20
print(plist[1][1])  # 50
```

6.3 Mathematical operators for List Objects

- Concatenation operator(+)

```
a=[10,20,30]
b=[40,50,60]
c=a+b
print(c)
# [10,20,30,40,50,60]

c=a+40
# TypeError: can only concatenate list (not "int") to list
```

- Repetition Operator(*)

```
x = [10,20,30]
y = x * 3
print(y)
# [10,20,30,10,20,30,10,20,30]
```

- Membership operators

```
n = [10,20,30,40]
print(10 in n)    # True
print(10 not in n) # False
print(50 in n)    # False
print(50 not in n) # True
```

6.4 Comparing List objects

```
x = ["Dog","Cat","Rat"]
y = ["Dog","Cat","Rat"]
z = ["Dog","CAT","RAT"]
print(x==y) # True
print(x==z) # False
print(x != z) # True
```

When ever we are using relatational operators(<,<=,>,>=) between List objects, `only first element comparison will be performed.`

```
print([50, 20] > [30, 100]) # True

print(["Dog","Cat"] > ["Rat"])  # False

print(max(["Dog","Cat","Rat"]))
# Rat
```

6.5 List in-built function

- index()
  - To find the index of a particular element

```
nlist = [10, 22, 11, 8, 11]

# Return first matched element
nlist.index(11)
# 2

nlist.index(33)
# ValueError: 33 is not in list
```

- count()
  - Count the number of occurrences of a element in the list

```
nlist = [10, 22, 11, 8, 11]

nlist.count(11)
# 2
```

Add element to a list

- append
  - Add an element to the end of the list

```
nlist = [10, 11, 12]

nlist.append(13)
# [10, 11, 12, 13]
```

- insert(pos, val)
  - insert element at a specified index

```
# Inserting 22 at 1th index
nlist.insert(1, 22)
# [10, 22, 11]
```

- `extend()`
  - Insert multiple elements at once, at the end of the list

```
nlist.extend([8, 11])
# [10, 22, 11, 8, 11]
```

Remove element from a list

- `pop()`
  - Remove by index
  - Remove last/indexed element
  - Also returns the popping element

```
nlist = [10, 22, 11, 8, 11]

# To pop last element
nlist.pop()
# 11

# To pop the (index-1)th element
nlist.pop(1)
# 22

nlist # [10, 11, 8]
```

- `del`
  - Remove by index
  - Remove indexed or delete list variable
  - Has no return values

```
nlist = [10, 22, 11, 8, 11]

# delete 3rd element
del nlist[2]

nlist # [10, 22, 8, 11]

# Del variable
del nlist

nlist
# NameError: name 'nlist' is not defined
```

- `remove()`
  - Remove by value
  - Gives ERROR if removed element not found

```
nlist = [10, 22, 11, 8, 11]

nlist.remove(6)
# ValueError: list.remove(x): x not in list

# Remove first matched element
nlist.remove(11)
# [10, 22, 8, 11]
```

Sort list

- `sort()` and `sorted()`

```
nlist = [10, 22, 11, 8, 11]

# Outplace sorting
a = sorted(nlist)
print(a)
# [8, 10, 11, 11, 22]


# Inplace sort
nlist.sort()
# [8, 10, 11, 11, 22]
```

To reverse a list

- `reverse()` and `reversed()`

```
nlist = [10, 22, 11, 8, 11]

# Reverse
nlist[::-1]
# [11, 8, 11, 22, 10]

# Outplace reverse
a = reversed(nlist)
print(list(a))
# [11, 8, 11, 22, 10]

# Inplace reverse
nlist.reverse()

nlist
# [11, 8, 11, 22, 10]
```

6.6 List Comprehensions:

List comprehensions are used for creating `new lists from other iterables` like tuples, strings, arrays, lists, etc

- Advantage
  - More time-efficient and space-efficient than loops.
  - Require fewer lines of code.
  - Transforms iterative statement into a formula

```
[i for i in range(10)]
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

## Sum of even numbers and odd numbers
lst=[1,2,3,4,5,6,7,8]

even_sum = sum([num for num in lst if num%2==0])
odd_sum = sum([num for num in lst if num%2!=0])
print(even_sum)  # 20
print(odd_sum)   # 16

# Flatten a list of lists into a single list
plist = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
# [i for i in j for j in plist] --> # WRONG

[i for j in plist for i in j]
# [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

.

.

.

.

.

.

.

.

.

.

.