08 May 2021 05:58

Introducing Lists — Janani Ravi

(1) List : ordered collection elements

(2) Square brackets

(3) Any data types

(4) Duplicates ok

(5) mixed data types

(6) Zero-based

(7) Access by index # [ ]

(8) list = [1, 2, 3]

list[3]

// IndexError

Performing Useful List Ops

(9) len(lst)

(10) lists are vectors

(11) /// list = [1, 2, 3]

list[2] = 4 ok

!! but list[3] = 4 // Index error

//// list[3:] = 4 // Type error

list[3:] = [4] // OK

list /// [1, 2, 3, 4]

raised when fn or operation applied to obj of INCORRECT type

(12) mnemonic last element of list at position length-1

(13) a list[len(list) - 1]

(14) getlast = lambda x: x[len(x)-1] getlast([1,2,3]) // 3

08 May 2021 12:11

Janani Ravi

Performing Useful list operations

(1) a += 1 ≡ a = a + 1

(2) Salary * 2

(3) += like list.extend()

```
>>> ccl=['Toyota Camry', 'Honda accord', 'Honda civic']
>>> ccl+=['Ford Focus', 'Ford Anglia']
>>> ccl
['Toyota Camry', 'Honda accord', 'Honda civic', 'Ford Focus', 'Ford Anglia']
```

(4) list.sort()

(5) list.reverse() // In place modific

(6) list.pop(), list.pop(1)

(7) list.append()

(8) list.extend()

list.append([1, 2]) // [... [1, 2]]

list.extend([1, 2]) // [... 1, 2]

list += [1, 2] // [... 1, 2]

(9) list.count()

list x = 2

list.count('ford focus')

(10) set(list) // removes duplicates

(11) list(lst) // new list.

(12) mylist.clear() !!

(13) newlist = mylist.copy // Deep copy

(14) del (mylist)

(15) shallow copy : Assignment operator

mylist2 = mylist

(16) mylist.remove( )

mylist.remove('ford anglia')

08 May 2021 21:27

Built-in fns

(1) max(mylist)

(2) TypeError
'>' not supported between instances of 'str' and 'int'

(3) min(list)

(4) len(list)

(5) sorted(list) | original list NOT affected

(6) sum(list)

(7) list_num *= 2 (2 must be integer)
- concatenate, doubles list

(8) all(list) - check if all elements in list are true

(9) all([]) True!! mnemonic AET

(10) any([]) False NYO(aklie)

10 May 2021 12:12

slicing ops on strings

1. String in Python essentially sequence of characters
2. mystring[0:18] — exclusive
3. my_str [::2] — step size
4. mystr [18:0:-1]
5. note that this won't work as expected!

   mystr [18:-1:-1] (negative indexing!)

   // ' '
6.

```
mystr="hamcheeseandeggs"
len(mystr)
16

mystr[16:0:-1]
'sggednaeseehcma'   <-- 'h' missing

mystr[16:-1:-1]     <-- Oops!!
''
mystr[::-1]
'sggednaeseehcmah'  <-- Reverse
```

```
mystr[::-1]
'sggednaeseehcmah'

>>> mystr[16:-17:-1]
'sggednaeseehcmah'

mystr[strlen:-(len(mystr)+1):-1]
'sggednaeseehcmah'

mystr[strlen:-len(mystr)-1:-1]
'sggednaeseehcmah'
```

Reverse

N.t.c

we

(7) Reverse string

mystr[::-1]

(8) string in python actually a class

(9) class fns can be invoked ! with dot operator.

(10) mystr.split()

08 May 2021 22:36

list slicing

1. my_list[0:4]
   - up to but not including
2. myList = [1, 2, 3, 4]
   my_list[0:4] // get all elements
   - Length of List
3. list slicing produces deep copy
4. To get all elements, specify end value equal to the number of elements or greater.
5. my_list[4:] // empty list
   my_lst[4] // index Error
6. lst[-4:-1] legit

08 May 2021 23:35

Step size

1. default is one
2. my_cars[::2]
3. my_list[::-1] reverse elements in list
4. my_list = ['a', 'b', 'c', 'd', 'e']
   my_list[6::-1] = ["aa", "bb", "cc"]

```
mylist=['a', 'b', 'c', 'd', 'e']
mylist[6:-1]=['aa','bb','cc']
mylist
 //  ['a', 'b', 'c', 'd', 'e', 'aa', 'bb', 'cc']
mylist[100:-1]=["tom", "dick"]
mylist
  // ['a', 'b', 'c', 'd', 'e', 'aa', 'bb', 'cc', 'tom', 'dick']
```

// Note.

5. 'ford focus' in my_list
   ≡
6. my_list.pop(index_value)
7. my_list.pop(-2)

08 May 2021 23:58

Strings

(1) Can think of string as list of characters

(2) x = "world"

x[0]

x[6] // IndexError

(3) strings are immutable

x[1] = "b" X - Wrong. TypeError

(4) a,b,c,d,e = x ✓✓

a // 'w'

b // '\0'

(5) a,b,c,d = x. // Value error

"Too many values to unpack"!

```
aa,bb,cc,dd=x
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    aa,bb,cc,dd=x
ValueError: too many values to unpack (expected 4)
```

(6) Note that the following is valid

aa, bb, _, _, _ = x

(7) place = input('where r u from: ')

(8) strings also support slicing

(9) to remove a variable, use del

del place. // frees memory

09 May 2021 06:56

Functions on strings

(1) starts with

```
plc .startswith('d')   // True or False
                       // returns Boolean
```

(2) endswith

Note

```
plce.endswith('city')   // True
```

```
"thomas".endswith('mas')
True
```

(3) mystr.count('x')

mystr.count('tom')

(4) str.count(sub, start=0, end=len(str))

```
res=mystr2.count('tom',2)
type(res)
<class 'int'>
```

returns an int

(5) str.upper()

(6) str.lower()

(7) str.find(' ')

returns index position (as int)
of character

```
mystr2.find('tom')   //   6
 type(mystr2.find('tom'))
//   <class 'int'>
mystr2.find('zz')   //   -1
```

NOTE

W3Schools

"The find method is almost the same as index method, the only diff being that the index method raises an exception if value not found.

```
mystr2.index('zz')
Traceback (most recent call last):
  File "<pyshell#36>", line 1, in <module>
    mystr2.index('zz')
ValueError: substring not found
```

Val Error

(8) str.index(),

(9) str.split(separator, maxsplit)

```
 mystr.split('l')
['Thomas Dow', 'ing']
```

Default is -1 (all)

(10) str.join( )

- reverse of split method

(all occurrences)

```
jc2='|'
jc2.join(["spam", "eggs", "cheese"])
'spam|eggs|cheese'
```

(11) mystr.split( ) (default: into words)

(12) mystr.upper( )

(13) mystr.isupper( )

(14) mystr.lower( )

10 May 2021 20:43

Tuples

① like lists, tuples are ordered collection of elements

② ( ) — Tuples.

③ lists, tuples, strings, dictionaries, sets are all classes

④ tuples are immutuple !

⑤ Similar to lists, tuples are ordered collection of elements

⑥ my_tuple = ()

⑨ type (my_tuple) // tuple

⑩ int_tuple = (1, 2, 3)

⑪ str_tuple = ('ham', 'eggs')

⑫ mixed_tuple = ('ham', 1, 2, 3)

⑬ int_tuple * 3 // (1,2,3, 1,2,3, 1,2,3)
mixed_tuple * 3 // ✓

⑭ mixed_type = "ham", "eggs", 1, 2, 3
(no brackets necessary)

⑮ When ...

(15) When assign values to variable where values separated by commas, Python interprets this as tuple

(16) We refer to elements of list elements of tuple often referred to as Fields

(17) a, b, c = my_tuple

a = 1
b = 2
c = 3

(18) a, b, _ = my_tuple

But _ is Assigned to 3

(19) a, b = my_tuple.

//Value error: Not enough values to unpack. (...)

(20) CAN have nested tuples

(21) can have list within tuple.

(22) my_tup = ((1,2,3), [1, 2, 3], "ham")

(23) my_tup[0]

(24) out-of-range

Index Error

```
int_tuple   //  (1, 2, 3)

int_tuple[4]
Traceback (most recent call last):
  File "<pyshell#45>", line 1, in <module>
    int_tuple[4]
IndexError: tuple index out of range

 int_tuple[4:]
()
```

(25) mixed_tuple[1][2] (nested Tuple)

(26) Performing slicing operation on list gives list

Performing slicing operations on tuple gives tuple

(27) my_tuple.index('H')

(28) 'e' in my_tuple (in Keyword)

(29) 'a' not in my_tuple

29 'e' not in my_tuple

30 sum(my_tuple)

31 list(my_tuple)

32 set(my_tuple) // removes duplicates

11 May 2021 11:24

Tuple immutability

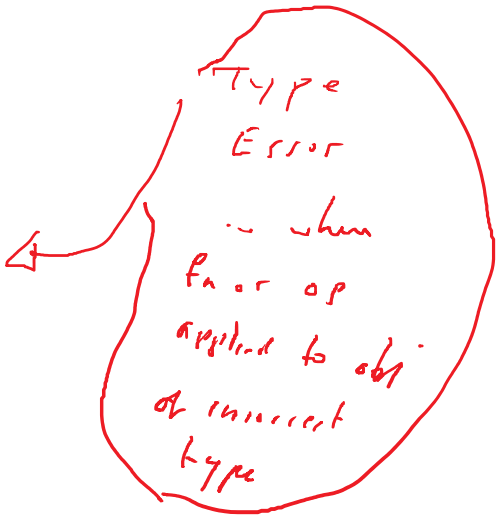
(1) Sig diff between lists and tuples is that tuples are immutable

(mnemonic: immutuple)

```
Int_tuple // (1, 2, 3)

int_tuple[1]=4
Traceback (most recent call last):
  File "<pyshell#48>", line 1, in <module>
    int_tuple[1]=4
TypeError: 'tuple' object does not support item assignment

lst_tup=list(int_tuple)
lst_tup   //   [1, 2, 3]
lst_tup[1]=4
lst_tup   //   [1, 4, 3]
```



(2) list within A nested tuple may be updated

tup[1][2]

(3) Cannot delete element of tuple, but can delete tuple

```
del int_tuple[3]
Traceback (most recent call last):
```

```
del int_tuple[3]
Traceback (most recent call last):
  File "<pyshell#59>", line 1, in <module>
    del int_tuple[3]
TypeError: 'tuple' object doesn't support item deletion

del int_tuple
int_tuple
Traceback (most recent call last):
  File "<pyshell#61>", line 1, in <module>
    int_tuple
NameError: name 'int_tuple' is not defined
```

Type error

now Name error

4 Zip function

returns iterator of tuples

```
tupa = (1,2,3)
tupb = ("a","b","c")
list(tupa, tupb)
```

5

```
result = tuple(zipped)
// A tuple of tuples!
```

6 To unzip

```
tuple_x, tuple_y = zip(*result)
```

7 zip Also works with lists

```
zipped_list = zip(lista, listb)
result = list(zipped)
// A list of tuples
```

// A list of tuples

12 May 2021 10:09

Other complex data types

(1) Lists !! (repetitions)

(2) Set : An unordered collections of unique elements