Python Data Types - List

In this lesson, we learn 2 important data types:

- List : an array of objects
- Tuple : like List, but immutable (or unchangeable)

List Function Summary

- len() the length of the list
- index() find an item's location
- in check existence
- append() add more items from the back
- insert() add item at a given position
- my_list = [] empty a list
- sort() sort a list
- reverse() reverse a list
- extend() extend (join) two lists
- del removing items from list
- pop() removing from the back of a list
- range() create a list with numbers

List (https://docs.python.org/3/library/stdtypes.html#list)

List is a sequence of objects : number, character, string, object.

List sequence **delimitor** is square brackets: [,]

List may be called Array, Vector, Tensor in other lang.

In [1]:
```
empty_list = []

number_list = [-1, 0, 1]

my_shopping_list = ['Milk', 'Eggs', 'Cheese', 'Butter']
```

In [2]:
```
type(empty_list), type(number_list), type(my_shopping_list)
```

Out[2]:
```
(list, list, list)
```

index

The index of a list is a sequence number of an item in the list

The index is used to access the element of the list

Syntax:

my_list[index]

In [27]:
```
# varaible snake_name is a string
snake_name = ['p', 'y','t','h','o','n']
# first char
snake_name[0]
```

Out[27]:
```
'p'
```

index is zero-based

Negative index means counting from the end of the list. -1 means the last element of the list

In [24]:
```
# varaible snake_name is a string
snake_name =  ['p', 'y','t','h','o','n']
# last char
snake_name[-1]
```

Out[24]:
```
'n'
```

Becareful of the index number of the last element:

In [28]:
```
snake_name =  ['p', 'y','t','h','o','n']
snake_name[6]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-28-b1fa574bf374> in <module>
      1 snake_name =  ['p', 'y','t','h','o','n']
----> 2 snake_name[6]

IndexError: list index out of range
```

In [26]:
```
snake_name3[5]
```

Out[26]: 'n'

In [51]:
```
# going backward using negative index:
# last char
snake_name3[-1]
```

Out[51]: 'n'

In [52]:
```
snake_name3[-3]
```

Out[52]: 'h'

string is a special list of characters

In [4]:
```
# varaible snake_name is a string
snake_name = "python"
```

In [42]:
```
type(snake_name)
```

Out[42]: str

In [43]:
```
# varaible snake_name2 is a list
snake_name2 = ['p', 'y','t','h','o','n']
```

In [44]:
```
type(snake_name2)
```

Out[44]: list

In [20]:
```
# convert string to list
snake_name3 = list(snake_name)
print(snake_name3)
```

```
['p', 'y', 't', 'h', 'o', 'n']
```

In [21]:
```
# length of list
len(snake_name3)
```

Out[21]: 6

- word is a list of alphabets
- sentence is a list of words and punctuations.
- paragraph is a list of sentences
- chapter ...

In [53]:
```
# python list can be made of different types
my_list = ['This', 'book', 'costs', 10.50, '$']
```

common operations / functions

len() - count list's length

In [2]:
```
my_list = ['This', 'book', 'costs', 10.50, '$']
len(my_list)
```

Out[2]: 5

In [3]:
```
snake_name =  ['p', 'y','t','h','o','n']
snake_name[len(snake_name) - 1]
```

Out[3]: 'n'

In [4]:
```
type(my_list[-2]), type(my_list[1])
```

Out[4]: (float, str)

index() - find an item's location

In [5]:
```
print(my_list.index('book'))
```

```
1
```

in - check existence

In [6]:
```
print('book' in my_list)
```

```
True
```

In [7]:
```
print('cost' in my_list)
```

```
False
```

append() - add more items from the back

In [8]:
```
my_list.append('I am going to order it')
```

In [9]:
```
print(my_list)
```

```
['This', 'book', 'costs', 10.5, '$', 'I am going to order it']
```

insert() - add item at a given position

In [10]:
```
my_list.insert(1, 'computer')
```

In [11]:
```
print(my_list)
```

```
['This', 'computer', 'book', 'costs', 10.5, '$', 'I am going to order it']
```

empty a list

In [63]:
```
my_list = []
print(my_list)
```

```
[]
```

In [64]:
```
len(my_list)
```

Out[64]:
```
0
```

sort() - sort a list

In [65]:
```
num_list = [120, 10, -1, 9999]
```

In [66]:
```
num_list.sort()
print(num_list)
```

```
[-1, 10, 120, 9999]
```

In [67]:
```
# sort reverse order
num_list.sort(reverse=True)
print(num_list)
```

```
[9999, 120, 10, -1]
```

reverse() - reverse a list

In [68]:
```
num_list
```

Out[68]:
```
[9999, 120, 10, -1]
```

In [69]:
```
num_list.reverse()
num_list
```

Out[69]:
```
[-1, 10, 120, 9999]
```

extend() - a list

In [48]:
```
snake_name =  ['p', 'y','t','h','o','n']
PC_Components = ['RAM', 'Power Supply', 'Hard Drive']

snake_name.extend(PC_Components)
print(snake_name)
```

```
['p', 'y', 't', 'h', 'o', 'n', 'RAM', 'Power Supply', 'Hard Drive']
```

In [50]:
```
snake_name =  ['p', 'y','t','h','o','n']
PC_Components = ['RAM', 'Power Supply', 'Hard Drive']

# what if you use append()

snake_name.append(PC_Components)
print(snake_name)
```

```
['p', 'y', 't', 'h', 'o', 'n', ['RAM', 'Power Supply', 'Hard Drive']]
```

del - removing item from list

In [ ]:
```
del PC_Components_Checklist[-1]
```

In [ ]:
```
print(PC_Components_Checklist)
```

pop() - remove from back

In [ ]:
```
last_item = PC_Components_Checklist.pop()
print(last_item)
```

In [ ]:
```
print(PC_Components_Checklist)
```

range()

quickly generate an array - list of numbers

In [18]:
```
arr = list(range(15))
arr
```

Out[18]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

generate a list of words from a sentence: split()

Strings can be split into a set of substrings when they are separated by a repeated character. If a string consists of a simple sentence, the string can be split based on spaces. The split() function returns a list of substrings. The split() function takes one argument, the character that separates the parts of the string.

In [45]:
```
split_fn_description = """
Strings can be split into a set of substrings when they are separated by a re
"""

word_list = split_fn_description.split()
word_list
```

Out[45]:
```
['Strings',
 'can',
 'be',
 'split',
 'into',
 'a',
 'set',
 'of',
 'substrings',
 'when',
 'they',
 'are',
 'separated',
 'by',
 'a',
 'repeated',
 'character.',
 'If',
 'a',
 'string',
 'consists',
 'of',
 'a',
 'simple',
 'sentence,',
 'the',
 'string',
 'can',
 'be',
 'split',
 'based',
 'on',
 'spaces.',
 'The',
 'split()',
 'function',
 'returns',
 'a',
 'list',
 'of',
 'substrings.',
 'The',
 'split()',
 'function',
 'takes',
 'one',
 'argument,',
 'the',
 'character',
```

```
 'that',
 'separates',
 'the',
 'parts',
 'of',
 'the',
 'string.']
```

In [47]:
```
split_fn_description = """
Strings can be split into a set of substrings when they are separated by a re
"""

sentence_list = split_fn_description.split('.')
sentence_list
```

Out[47]:
```
['\nStrings can be split into a set of substrings when they are separated b
y a repeated character',
 ' If a string consists of a simple sentence, the string can be split based
on spaces',
 ' The split() function returns a list of substrings',
 ' The split() function takes one argument, the character that separates th
e parts of the string',
 '\n']
```

Split a string for every line

the new line is a special character in a string: \n

In [52]:
```
essay = '''The Zen of Python
by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.'''

# write your code at below

lines = essay.split('\n')  # number lines

for l in lines:
    print(l)
```

```
The Zen of Python
by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
```

Multi Dimensional List

The component of a list can be a list

In [17]:
```
main_Checklist = ['CPU', 'Motherboard', 'Graphics Card', 'RAM']
print('main_Checklist\n', main_Checklist)
print()

extra_Checklist = ['Operating Sys','Keyboard']
print('extra_Checklist\n', extra_Checklist)
print()

computer_Component_Checklist = [main_Checklist, extra_Checklist]
print('computer_Component_Checklist\n', computer_Component_Checklist)
print()

print('computer_Component_Checklist[0]\n', computer_Component_Checklist[0])
print('computer_Component_Checklist[1]\n', computer_Component_Checklist[1])
```

```
main_Checklist
 ['CPU', 'Motherboard', 'Graphics Card', 'RAM']

extra_Checklist
 ['Operating Sys', 'Keyboard']

computer_Component_Checklist
 [['CPU', 'Motherboard', 'Graphics Card', 'RAM'], ['Operating Sys', 'Keyboa
rd']]

computer_Component_Checklist[0]
 ['CPU', 'Motherboard', 'Graphics Card', 'RAM']
computer_Component_Checklist[1]
 ['Operating Sys', 'Keyboard']
```

Iterate Through a List with For Loop

In [32]:
```
# method 1
snake_name =  ['p', 'y','t','h','o','n']
for letter in snake_name:
    print(letter, end='')
```

```
python
```

In [33]:
```
# method 2
snake_name =  ['p', 'y','t','h','o','n']
total = len(snake_name)
for i in range(total):
    print(snake_name[i], end='')
```

```
python
```

Iterate Through a List with For Loop, keeping the index number

In [34]:
```
# Method 1
snake_name =  ['p', 'y','t','h','o','n']
for i in range(len(snake_name)):
    print(i, ') ',snake_name[i])
```

```
0 )  p
1 )  y
2 )  t
3 )  h
4 )  o
5 )  n
```

Using **enumerate**

In [35]:
```
# Method 2
snake_name =  ['p', 'y','t','h','o','n']
for i, l in enumerate(snake_name):
    print(i, ') ', l)
```

```
0 )  p
1 )  y
2 )  t
3 )  h
4 )  o
5 )  n
```

Tuple (https://docs.python.org/3/library/stdtypes.html#tuple)

Tuple is a list whose item can not be changed.

Tuple sequence **delimitor** is parentheses: (,)

In [ ]:
```
t = ('I', 'play', 'tennis')
```

In [ ]:
```
print(t)
```

In [ ]:
```
type(t)
```

In [ ]:
```
len(t)
```

In [ ]:
```
t[0]
```

In [ ]:
```
t[1]
```

In [ ]:
```
t[2]
```

In [ ]:
```
t[2] = 'ping-pong'
```

In [ ]:
```
# convert tuple to list
lst = list(t)
```

In [ ]:
```
print(lst)
```

In [ ]:
```
type(lst)
```

In [ ]:
```
lst[2] = 'ping-pong'
```

In [ ]:
```
# convert modified list back to tuple
tpl = tuple(lst)
```

In [ ]:
```
type(tpl)
```

In [ ]:
```
print(tpl)
```

Note: We will cover topics on looping later

In [ ]:
```

```