List

idx: 0 ← [1, 'vision', '30']
1 above 'vision', 2 above '30'

index is important.

List

an ordered sequence of elements, can mix element types

a list is denoted by square brackets, []

a list contains elements
- usually homogeneous (ie, all integers)
- can contain mixed types (not common)

list elements can be changed so a list is mutable

اعضای لیست قابل تغییرند!

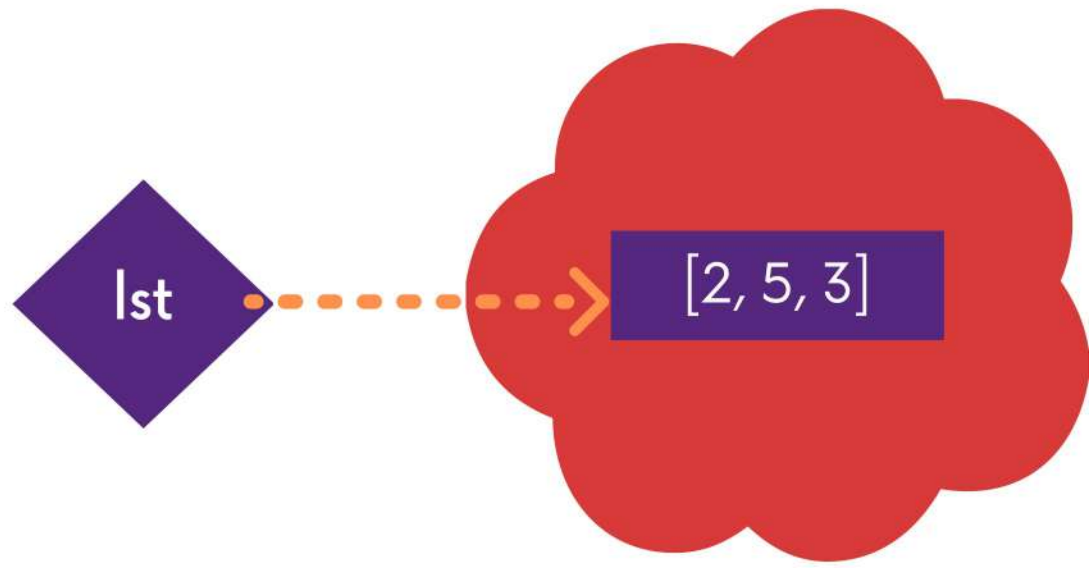
immutable

```
>>> empty_lst = []
>>> lst = [2, 'a', 4, [1, 2]]
>>> len(lst)                              #out: 4
>>> lst[0]                                #out: 2
>>> lst[0] + 1                            #out: 3
>>> lst[3]                                #out: [1, 2]
>>> lst[4]                                #out: error
```

- lists are mutable!
- assigning to an element at an index changes the value

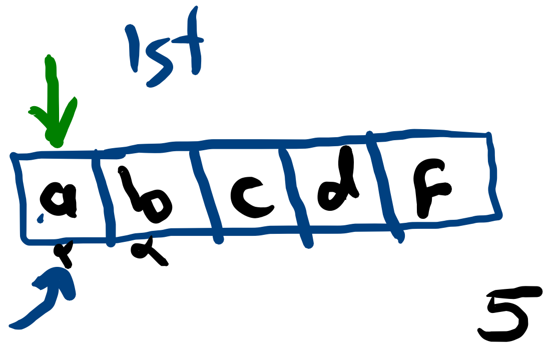
List Manipulation

```
              0     1   2
>>> lst = [2, 1, 3]
>>> lst[1] = 5
>>> lst                  #out: [2, 5, 3]
#note this is the same object lst
```



mutable vs immutable ?

Iteration over a List



i = a
i = b
⋮
i = f

5

for i in range(5) :

0, 1, 2, 3, 4

lst[i] lst[i] 2 3

0

common pattern, iterate over list elements

```
#Sum of elements
>>> total = 0
>>> for i in range(len(lst)):
        total += lst[i]

>>> print(total)
```

```
#Sum of elements
#like Strings, can iterate
#over list elements directly!
>>> total = 0
>>> for i in lst:
        total += i

>>> print(total)
```

- list elements are indexed 0 to len(L)-1
- range(n) goes from 0 to n-1

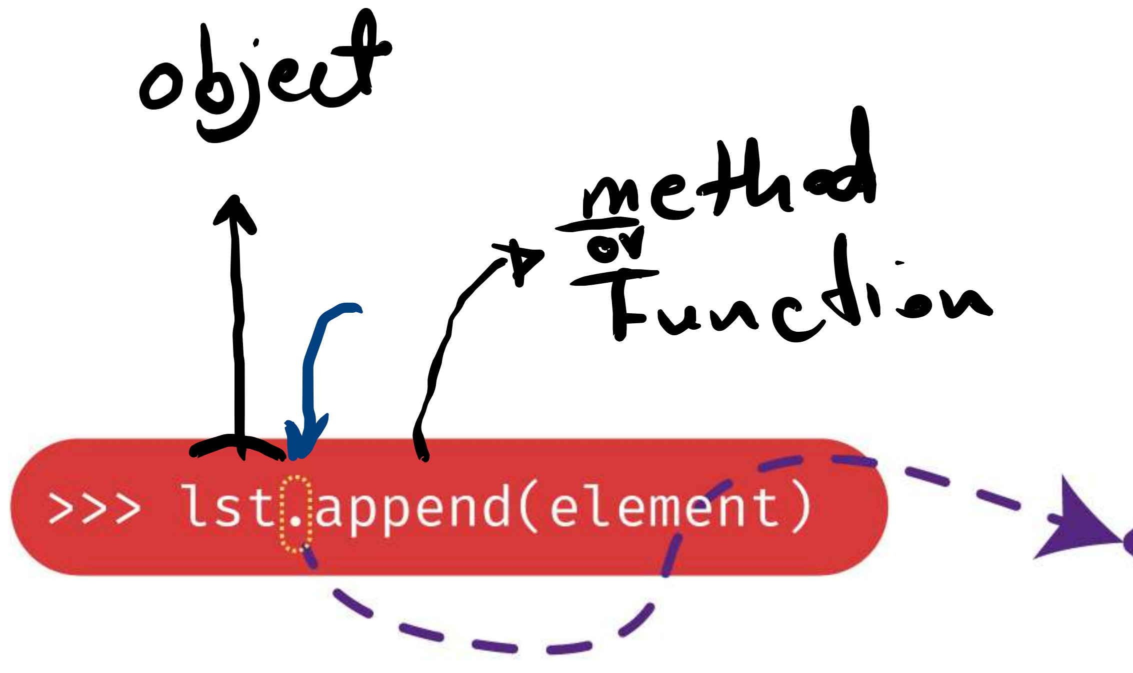
Operation on List - ADD

add elements or list to end of list with

```
#add elements
>>> lst.append(element)
#Example
>>> lst = [1, 2, 3]
>>> lst.append(4) #out: [1, 2, 3, 4]
```

```
#add list
#First way!
>>> lst1 = [1, 5, 8]
>>> lst2 = [4, 6, 9]
>>> lst = lst1 + lst2 = [1, 5, 8, 4, 6, 9]
# lst1 and lst2 unchanged
#Second way!
>>> lst1.extend([0, 6])
#lst1 = [1, 5, 8, 0, 6]
```

What is the Dot ?



```
>>> lst.append(element)
```

- lists are Python objects, everything in Python is an object
- objects have data
- objects have methods and functions
- access this information by object_name.do_something()
- will learn more about these later

[a, b, c, d]
0 1 2 3

Operation on List - Remove

- delete element at a specific index

```
>>> del(lst[index])
```

- remove element at end of list with

```
>>> lst.pop()
```

pop(1)

- remove a specific element with

```
>>> lst.remove(element)
```

```
>>> lst = [2,1,3,6,3,7,0]
>>> del(lst[1])
#out: lst= [2,3,6,3,7,0]
>>> lst.remove(2)
#out: lst= [3,6,3,7,0]
>>> lst.pop()
#out: lst= [1, 3, 7]
```

lst.rem(b) = [a, c, d]

lst = ['v', 'i', 's', 'i', 'o', 'n'] →

For

↳ String! → Join!

☺.join(lst)

out: vision   v_i_s_i_o_n

Convert Lists to String & back

string to list, returns a list with every character from s an element in lst → `>>> list(s)`

split a string on a character parameter, splits on spaces if called without a parameter → `>>> s.split()`

to turn a list of characters into a string, can give a character in quotes to add char between every element → `>>> ''.join(L)`

ch

```
>>> s = "I<3 cs"
# s is string!

>>> list(s)
# out: ['I','<','3','c','s']

>>> s.split('<')
#out: ['I', '3 cs']

>>> lst = ['a','b','c']
#lst is List

>>> ''.join(lst)
#out: "abc"

>>>'_'.join(lst)
#out: "a_b_c"
```

```
lst = [0, 2, 1]
lst_2 = sorted(lst) -> out: [0, 1, 2]
print(lst)
        [0, 2, 1]
```

```
sort -> lst = [0, 2, 1]
        lst.sort()
        print(lst) -> [0, 1, 2]
```

different!

Other List operations

```
>>> lst.sort()
```

```
>>> sorted(lst)
```

```
>>> lst.reverse()
```

```
>>> lst=[9,6,0,3]
>>> sorted(lst)
# returns sorted list, does not mutate lst
>>> lst.sort()
# mutates lst=[0,3,6,9]
>>> lst.reverse()
#mutates lst=[9,6,3,0]
```

Exercises 04

List

450

List

Exercise 01

- The following is a list of 10 students ages:

ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]

remove

1. Sort the list and find the min and max age. (50)
2. Add the min age and the max age again to the list.(50)
3. Find the median age (one middle item or two middle items divided by two).(100)
4. Find the average age (sum of all items divided by their number).(100)
5. Find the range of the ages (max minus min).(50)
6. Compare the value of (min - average) and (max - average), use abs() method.(100)

$> \quad < \quad =$

Tuples

Speed { Tuple ↑
List ↓

Tuples

- an ordered sequence of elements, can mix element types
- cannot change element values, immutable

```
>>> tpl = ()      #empty tuple
```

```
>>> tpl = (2, "book", 3)
>>> tpl[0]        #out: 2

>>> tpl = (2, "book", 3) + (5, 6)  #out: (2, "book", 3, 5, 6)
>>> tpl[1:2]      #out: ("book", )
                  #extra comma means a tuple with one element.
>>> tpl[1:3]      #out: ("book", 3)
>>> len(tpl)      #out: 5
>>> tpl[1]=4      #gives error, can't modify object (immutable)
```

UI

me = ('#FF1920',
('#FCF189')

✓

Tuples

- used to return **more than one value** from a function

```python
def quotient_and_remainder(x, y):
    q = x // y
    r = x % y
    return (q, r)

(quot, rem) = quotient_and_remainder(4,5)
```

integer division

reuse

Manipulating Tuples

- used to return more than one value from a function

```
>>> t = ('foo', 'bar', 'baz', 'qux', 'quux', 'corge')
>>> t[0]                               #out: 'foo'
>>> t[-1]                              #out: 'corge'
>>> t[1::2]    [ start : end : step ]  #out: ('bar', 'qux', 'corge')
>>> t[::-1]                            #out: ('corge', 'quux', 'qux',
                                       #'baz', 'bar', 'foo')
>>> t[2] = 'Bark!'                     #out: TypeError: 'tuple' object does
                                       #not support item assignment
```

1 3

```
#Important Point
>>> A = ("book")        #Type: String
>>> B = ("book",)       #Type: Tuple
```

Tuples vs List

Tuple $\uparrow$ list $\downarrow$

speed

Program execution is faster when manipulating a tuple than it is for the equivalent list. (This is probably not going to be noticeable when the list or tuple is small.)

$\rightarrow$ list $\dashrightarrow$ Numpy

Other Tuple Operations

#add an element

#remove an element

#replace an element

```
t = (0, 1, 2)
t_add = t + (3, 4, 5)
#out: t_add = (0, 1, 2, 3, 4, 5)


# if you want to insert a new item at any
#position, or remove elements you need to
convert a tuple to a #list.
l = list(t)
l.insert(2, 100)
#out: l = [0, 1, 100, 2]


#remove
l = list(t)
l.remove(1)
t_remove = tuple(l)
#t_remove: (0, 2)
```

(0, 1, 2)

$\rightarrow$ [0, 1, 2]

$\downarrow$

[0, 1, 100, 2]

[0, 2]

$\downarrow$

(0, 2)

src: note.nkmk.me

Exercises 05

Tuple

1100

Tuple

Exercise 01

1. Create an empty tuple (50)
2. Create a tuple containing names of your sisters and your brothers, separately (imaginary siblings are fine) (100)
3. Join brothers and sisters tuples and assign it to siblings (100)
4. How many siblings do you have? (50)
5. Modify the siblings tuple and add the name of your father and mother and assign it to family_members (100)

500

Tuple

Exercise 02

1. Create fruits, vegetables and animal products tuples, separately. (50)
2. Join the above three tuples and assign it to a variable called food_stuff_tp. (100)
3. Change the about food_stuff_tp tuple to a food_stuff_lt list (50)
4. Slice out the middle item of items from the food_stuff_tp tuple or food_stuff_lt list. (100) $\leftarrow$
5. Slice out the first three items and the last three items from food_staff_lt list (100)
6. Delete the food_staff_tp tuple completely (100) search

Tuple

Exercise 03

1. Check if 'Estonia' is a nordic country, print True, Otherwise print False!
2. Check if 'Iceland' is a nordic country, print True, Otherwise print False!

```
nordic_countries = ('Denmark', 'Finland','Iceland', 'Norway', 'Sweden')
```

Dictionary

{"Ali" : 20 , "Zahra": 19}

Key value

Dictionary

- Dictionaries are used to store data values in key:value pairs.

- A dictionary is a collection which is ordered, changeable and do not allow duplicates.

When we say that dictionaries are ordered, it means that the items have a defined order, and that order will not change. Unordered means that the items does not have a defined order, you cannot refer to an item by using an index.

Dictionaries are changeable, meaning that we can change, add or remove items after the dictionary has been created.

Dictionaries cannot have two items with the same key

- Dictionaries are written with curly brackets, and have keys and values:

```
thisdict = {"brand": "Ford","model": "Mustang"}
```

Dictionary Manipulation

add	`dict[key]= value`
test	`key in dict`
delete	`del(dict[key])`
Keys	`dict.keys()`
Value	`dict.values()`

```
>>> grades = {'Ana':'B','John':'A+','Denise':'A','Katy':'A'}

>>> grades['Sylvan'] = 'A'    # add entry

>>> 'John' in grades      #test if key dictionary
#out: returns True

>>> 'Daniel' in grades
#out: returns False

>>> del(grades['Ana'])    #delete entry

>>> grades.keys()           #keys
#out: returns ['Denise','Katy','John','Ana']
#get an iterable that acts like a tuple of all keys
#no guaranteed order

>>> grades.values()          #values
#out: returns ['A', 'A', 'A+', 'B']
#get an iterable that acts like a tuple of all values
#no guaranteed order
```

$$\{ \underbrace{\text{"Ali"}}_{\text{key}} : \underbrace{[20, 18, 17]}_{\text{value}} \}$$

DICTIONARY KEYS & VALUES

Values	Keys
any type (immutable and mutable)	must be unique
can be duplicates	immutable type (int, float, string, tuple,bool)

Exercises 06

Dictionary

800

Dictionary

Exercise 01

1. Create an empty dictionary called dog (50)
2. Add name, color, breed, legs, age to the dog dictionary (100)

Dictionary

Exercise 02

1. Create a student dictionary and add first_name, last_name, gender, age, skills, country, city and address as keys for the dictionary (100)
2. Get the length of the student dictionary (50)
3. Get the value of skills and check the data type, it should be a list (50)
4. Modify the skills values by adding one or two skills (100)
5. Get the dictionary keys as a list (100)
6. Get the dictionary values as a list (100)
7. Change the dictionary to a list of tuples using items() method (50)
8. Delete one of the items in the dictionary(50)
9. Delete ~~one of the~~ dictionaries(50) ✓
   all

Set

Set

- A Python set is an unordered list of immutable elements. It means:

- Elements in a set are unordered.
- Elements in a set are unique. A set doesn't allow duplicate elements.
- Elements in a set cannot be changed. For example, they can be numbers, strings, and tuples, but cannot be lists or dictionaries.

```
{'r', 'l', 't', 'e'}  #curly brace {}
```

Set Manipulation

empty set	`set()`
cast	`set(list)`
size	`len(set)`
check existance	`element in set`
Adding elements	`set.add(element)`
Removing an element	`set.remove(element)`
remove without error	`set.discard(element)`
Removing all elements	`set.clear()`

```
letters = set(['P','C'])
print(letters)    1st
#out: {'P', 'C'}        { }

ratings = {1, 2, 3, 4, 5}
size = len(ratings)
print(size)
#out: 5
                              { }
ratings = {1, 2, 3, 4, 5}
rating = 1
if rating in ratings:    True   { }
   print('there is')
#out: there is

s = {'P', 'S', 'P'}
s.remove('S')
```

Exercises 07

Set

750

Set

Exercise 01

1. Find the length of the set it_companies (50)
2. Add 'Twitter' to it_companies (50)
3. Insert multiple IT companies at once to the set it_companies (100)
4. Remove one of the companies from the set it_companies(50)
5. What is the difference between remove and discard (100)

$\downarrow$

$\underbrace{\text{it\_companies}}_{\text{set}}$ = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}

Set

Exercise 02

- Convert the ages to a set and compare the length of the list and the set, which one is bigger?

age = [22, 19, 24, 25, 26, 24, 25, 24]

Set

Exercise 04 (Search!)

- (I am a teacher and I love to inspire and teach people.) How many unique words have been used in the sentence?

  Use the split methods and set to get the unique words.