```
Teacher: Mohammad Foroghi (+98 922 621 91 73)
date:  Friday, May 8, 2020
subject of class: Data types
```

Data types

```
1. Boolean
    I. True
    II. False

2. Numeric
    I. Int           Integer      -1 -1000 -250 1 2 3 ...
    II. float        float        1.1 0.23 -0.45 -100.56
    III. complex     complex      5 + 2j  10 - 3j

3. Sequence
    I. str           String             'Hello'  "Hello"
    II. sets         set   {}           {1, 'hi', 250}
     # not important
    III. tuples      tuple ()           (1, '2', ('hi', 250))
    # not important
    IV. Lists        list  []           [1, {'d', 1}, 50, ('kg',)]
     # important
    V. Dict          Dictionaries  {}   {'apple': 'sib', 'bannana':
'moz'}    # important
```

Booleans Type

In [17]:

```
print(True)
print(False)

print(type(True))
```

```
True
False
<class 'bool'>
```

In [15]:
```
'''
True:
    Everything exceptions False
False:
    False None 0 [] () {} {}
'''

# True
print(bool(100))
print(bool('hello'))
print(bool([1, 2, 3]))
print("\n")

# False
print(bool(False))
print(bool({}))
print(bool([]))
```

```
True
True
True


False
False
False
```

Numerics Types

Integer

In [19]:
```
a = 1
print(type(a))

a = 10002464944
print(type(a))

a = -10002464944
print(type(a))
```

```
<class 'int'>
<class 'int'>
<class 'int'>
```

Floats Types

In [22]:
```
a = -0.23
print(type(a))

a = 1.78
print(type(a))

a = -1000.568
print(type(a))

a = 1045454874.26448
print(type(a))
```

```
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
```

complex

In [23]:
```
a = complex(2, 3)
print(a)
print(type(a))

a = complex(-2, 5)
print(a)
print(type(a))

a = complex(2, -30)
print(a)
print(type(a))

a = complex(-25, -3)
print(a)
print(type(a))
```

```
(2+3j)
<class 'complex'>
(-2+5j)
<class 'complex'>
(2-30j)
<class 'complex'>
(-25-3j)
<class 'complex'>
```

Sequence Types

String

In [11]:
```
a = 'hello'    # 'h' + 'e' + 'l' + 'l' + 'o'
               #  0     1     2     3     4

print(type(a))
print(type(b))
print("\n")

print(a[0])
print(a[1])
print(a[2])
print(a[3])
print(a[4])
```

```
<class 'str'>
<class 'str'>


h
e
l
l
o
```

In [9]:
```
# Unexcepted indent ---> syntax Error
a = 1
  print('hello')
1+2
```

```
  File "<ipython-input-9-0577c4a74e51>", line 4
    1+2
    ^
IndentationError: unexpected indent
```

In [43]:
```
# [0: 5] 0 1 2 3 4
a = 'it\'s me'

print(a[0])
print(a[1])
print(a[6])
print(a[3:7])
```

```
i
t
e
s me
ham


d
a
m
```

In [15]:
```
b = 'Mohammad'
print('b[2:5] = ', b[2:5])
print("\n")
print('b[-1] = ', b[-1])
print(b[-2])
print(b[-4])
print("\n")
print(b[-4:-1])
print("\n")
print(b[2:])
print(b[:5])
print("\n")
print('b[2::] = ', b[2::])
print('b[::2] =', b[::2])
print(b[::3])
print("\n")
print(b[::-2])   # Exceptions
print(b[::-3])
```

```
b[2:5] =  ham


b[-1] =  d
a
m


mma


hammad
Moham


b[2::] =  hammad
b[::2] = Mhma
Maa


dmao
dmo
```

```
Homework 1:

input: Mohammad
output: dammahoM
```

In [2]:
```
# Solution:
a = 'Mohammad'
print(a[::-1])
print('a[-1:] =', a[-1:])
#  M   o   h   a   m  m   a   d  |  M o h a m m a d
# -8  -7  -6  -5  -4 -3  -2  -1  |  0 1 2 3 4 5 6 7
print('a[-1:-8] =', a[-1:-8])
print('a[::-1] =', a[::-1])
```

```
dammahoM
a[-1:] = d
a[-1:-8] =
a[::-1] = dammahoM
```

String Methods

```
format

split
splitlines

capitalize
upper
casefold
lower

count
find <---> index
partition
replace

center
join
swapcase
```

In [37]:

```
# format
Firstname = 'Mohammad'
Lastname = 'Foroughi'

# point: print("This is Mohammad Foroughi")

print('This is {} {}'.format(Firstname, Lastname))
print('This is {} {}'.format(Lastname, Firstname))
print("\n")

print('This is {0} {1}'.format(Firstname, Lastname))
print('This is {1} {0}'.format(Firstname, Lastname))
print("\n")

print(f'This is {Firstname} {Lastname}')
print("\n")

print('This is {Firstname} {Lastname}. I like {like}'.format(Firstname="Moha
print("\n")

string = 'This is {} {}.'
print(string.format(Firstname, Lastname))

```

```
This is Mohammad Foroughi
This is Foroughi Mohammad


This is Mohammad Foroughi
This is Foroughi Mohammad


This is Mohammad Foroughi


This is Mohammad Foroughi. I like icecream


This is Mohammad Foroughi.
```

In [44]:
```
# Split

a = "Mammad is overthinking"
print(a.split())
print("\n")

b = "Mammad"          # 'M' 'a' 'm' 'm' 'a' 'd'
for element in b:
    print(element)

# splitlines
c = "This is Mohammad. \n I like banana. \n 123456789"
print(c.splitlines())
```

```
['Mammad', 'is', 'overthinking']


M
a
m
m
a
d
['This is Mohammad. ', ' I like banana. ', ' 123456789']
```

In [54]:
```
a = "mohammad is overthinking"
# capitalize
print("Capitalize :", a.capitalize())

# upper
print("upper :", a.upper())
print("\n")


b = "MOHammad is overthinking"
# casefold
print("casefold :", b.casefold())


c = "THIS IS MAMMAD"
# lower
print("lower :", c.lower())
```

```
Capitalize : Mohammad is overthinking
upper : MOHAMMAD IS OVERTHINKING


casefold : mohammad is overthinking
lower : this is mammad
```

In [64]:
```
a = 'Mohammad'
# count

print(a.count('m'))

b = "one two two three three three"
print('one: ', b.count('one'))
print('two: ', b.count('two'))
print('three: ', b.count('three'))
print('ree: ', b.count('ree'))
print('tw: ', b.count('tw'))
```

```
2
one:  1
two:  2
three:  3
ree:  3
tw:  2
```

In [68]:
```
a = 'Mohammad'

# find
print(a.find("M"))
print(a.find("m"))
print(a.find("a"))
print(a.find("Z"))        # find ---> Nothing: -1
print("\n")

# index == find
print(a.index("M"))
print(a.index("m"))
print(a.index("a"))
print(a.index("Z"))
```

```
0
4
3
-1


0
4
3

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-68-0d7cb1947c69> in <module>
     12 print(a.index("m"))
     13 print(a.index("a"))
---> 14 print(a.index("Z"))

ValueError: substring not found
```

In [69]:
```
# partition
a = "This is Mohammad and I like Icecream"
print(a.partition("and"))
```

```
('This is Mohammad ', 'and', ' I like Icecream')
```

In [70]:
```
# replace
a = "This is Mohammad and I like Icecream"
print(a.replace("Icecream", 'banana'))
```

```
This is Mohammad and I like banana
```

In [77]:
```
'''
for beautiful display:
    center
    join
    swapcase
'''

a = 'Mohammad'
print(a.center(10, '#'))
```

```
#Mohammad#
```

Sets Type

In [83]:
```
a = {'a', 2}                # not important
print(type(a))
print(a)

# 'set' object is not subscriptable === not callabe: a[1]
```

```
<class 'set'>
{2, 'a'}
```

Set Methods

```
add
clear
copy

difference
intersection

discard
isdisjoint
issubset
issuperset

pop
```

```
remove
union
update
```

In [89]:

```
# add
a = {1, 2, 3}

a.add(5)
a.add(4)
a.add("hello")
print(a)
```

```
{1, 2, 3, 4, 5, 'hello'}
```

In [91]:

```
# clear
a = {1, 2, 3}

a.clear()
print(a)
print(bool(a))
```

```
set()
False
```

In [92]:

```
# copy
a = {1, 2, 3}

b = a.copy()
print(b)
```

```
{1, 2, 3}
```

In [97]:
```
# difference
class_1 = {'Ali', 'Mammad', 'gholi'}
class_2 = {'asghar', 'jafar', 'akbar', 'Mammad'}

x = class_1.difference(class_2)
print(x)

y = class_2.difference(class_1)
print(y)
print("\n")


# intersection
z1 = class_1.intersection(class_2)
print(z1)

z2 = class_2.intersection(class_1)
print(z2)
```

```
{'gholi', 'Ali'}
{'akbar', 'jafar', 'asghar'}


{'Mammad'}
{'Mammad'}
```

In [98]:
```
# discard
class_2 = {'asghar', 'jafar', 'akbar', 'Mammad'}

class_2.discard('Mammad')
print(class_2)
```

```
{'akbar', 'jafar', 'asghar'}
```

In [111]:
```
class_1 = {'Ali', 'Mammad', 'gholi'}
class_2 = {'ali', 'jafar', 'akbar'}

#isdisjoint    not similar at all
x = class_1.isdisjoint(class_2)
print(x)

class_1 = {'apple', 'banana', 'cherry'}
class_2 = {'apple', 'banana', 'cherry'}
# issubset      Subset
y = class_1.issubset(class_2)
print(y)

# issuperset   All elemnts same
z = class_1.issuperset(class_2)
print(z)
```

```
True
False
True
True
```

In [119]:
```
# pop
class_1 = {'Ali', 'Mammad', 'gholi'}
class_1.pop()
print(class_1)

# remove
class_1 = {'Ali', 'Mammad', 'gholi'}
class_1.remove('Ali')
print(class_1)
```

```
{'Ali', 'Mammad'}
{'gholi', 'Mammad'}
```

In [140]:
```
class_1 = {'Ali', 'Mammad', 'gholi'}
class_2 = {'asghar', 'jafar', 'akbar', 'Mammad'}

# union
x = class_2.union(class_1)
print(x)

# update similar to add
class_1.update({1, 2, 3})
print(class_1)
```

```
{'asghar', 'gholi', 'Ali', 'akbar', 'Mammad', 'jafar'}
{1, 2, 3, 'gholi', 'Ali', 'Mammad'}
{'s', 'akbar', 'b', 'asghar', 'jafar', 'Mammad', 'i'}
```

Tuple

In [134]:
```
# we cannot add and modify element to tuples
class_1 = ('Ali', 'Mammad', 'gholi', (1, 2, 3))
print(class_1)
print(type(class_1))
print(class_1[1])
print(class_1[1:4])
print(class_1[::-1])
```

```
('Ali', 'Mammad', 'gholi', (1, 2, 3))
<class 'tuple'>
Mammad
('Mammad', 'gholi', (1, 2, 3))
((1, 2, 3), 'gholi', 'Mammad', 'Ali')
```

Tuple Methods

In [130]:
```
'''
count()
index()
'''

a = ('Ali', 'Mammad', 'gholi', 'gholi', (1, 2, 3))
# count
print(a.count('gholi'))

# index
print(a.index('gholi'))
print(a.index('Mammad'))
print(a.index((1, 2, 3)))
```

```
2
2
1
4
```

LIST

In [4]:
```
# Flexible

a = [1, 2, 3]
print(a)
print(type(a))

print(a[0])
print(a[2])
```

```
[1, 2, 3]
<class 'list'>
1
3
```

In [11]:
```
a = [1, 1.25, 2-5j, ('ali', 1), {'Mammad', 2}, {'a': 'one', 'b': 2}, [1, 2,
print(a)

print(a[1])
print(a[3])
print(a[5])

print(a[1: 5])
print(a[::-1])
```

```
[1, 1.25, (2-5j), ('ali', 1), {'Mammad', 2}, {'a': 'one', 'b': 2}, [1, 2, [35,
20]]]
1.25
('ali', 1)
{'a': 'one', 'b': 2}
[1.25, (2-5j), ('ali', 1), {'Mammad', 2}]
[[1, 2, [35, 20]], {'a': 'one', 'b': 2}, {'Mammad', 2}, ('ali', 1), (2-5j), 1.2
5, 1]
```

lists methods

```
append
extend
insert

copy

count
index

pop
remove
clear
del

sort
```

In [33]:

```
a = []

#append
a.append(1)
print(a)

a.append('Mammad')
print(a)

b = [1, 2, 3]
a.append(b)
print(a)

'''
a = [1, 2, 3]
b = [3, 4, 5]
a.append(b)  ---> [1, 2, 3, [3, 4, 5]]
a.extend(b)  ---> [1, 2, 3, 3, 4, 5]
'''

print('\n')
# extend
c = [1, 2, 3]
d = [3, 4, 5]
c.extend(d)
print(c)

# approachs:
# 1
e = [1, 2, 3]
f = [3, 4, 5]
print('e + f:', e + f)

# 2
g = [1, 2, 3]
h = [3, 4, 5]

for element in h:
    g.append(element)
print('g:', g)


for element in g:
    h.append(element)
print(h)

print('\n')
# insert
i = [1, 2, 3]
i.insert(1, 1000)   # insert(index, value)
print('i: ', i)

i.insert(2, 'Mammad')
print(i)
```

```
[1]
[1, 'Mammad']
[1, 'Mammad', [1, 2, 3]]


[1, 2, 3, 3, 4, 5]
e + f: [1, 2, 3, 3, 4, 5]
g: [1, 2, 3, 3, 4, 5]
[3, 4, 5, 1, 2, 3, 3, 4, 5]


i:  [1, 1000, 2, 3]
[1, 1000, 'Mammad', 2, 3]
```

In [34]:

```python
a = [1, 2, 3]
b = a.copy()
print(b)
```

```
[1, 2, 3]
```

In [38]:

```python
# count
a = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4]
print(a.count(1))
print(a.count(2))
print(a.count(3))
print(a.count(4))

print("\n")
# index
print('a.index(4):', a.index(4))
```

```
1
2
3
5


a.index(4): 6
```

In [50]:

```
a = [1, 2, 3]
# pop
a.pop()
print(a)

b = [1, 2, 3]
b.pop(1)        # pop(index)
print(b)


print("\n")
# remove
c = ['hello', 'world', 5, 100]
c.remove('hello')               # remove(Value)
print(c)

c.remove(100)
print(c)

print("\n")

# clear
d = [1, 2, 3]
d.clear()
print(d)

print("\n")
# del
e = [1, 2, 3]
del e
# print(e)

f = [1, 2, 3]
del f[1]
print(f)
```

```
[1, 2]
[1, 3]


['world', 5, 100]
['world', 5]


[]


[1, 3]
```

In [59]:

```
# sort
a = [1000, 200, -105, 1, 10, 50]
a.sort()
print(a)

b = ['Mammad', 'Amin', 'Amir', 'Asefe', 'Iman']
b.sort()
print(b)

b.sort(reverse=False)
print('when reverse is False(default):', b)

b.sort(reverse=True)
print('when reverse is True:', b)


# len()    It's Not A Method. It's a built-in function. Return number of ele
a = 'Mammad'
print(len(a))
a = [1, 2, 3, 4]
print(len(a))
a = {1, 2, 3, 4, 5}
print(len(a))
a = {'a': 1, 'b': 2}
print(len(a))

'''
def function(x):
    for element in x:
        return len(x)

number_of_Mammad_elements = function('Mammad Foroughi')
print(number_of_Mammad_elements)


c = ['Asefeh', 'Iman', 'Ali', 'Amin', 'Amir', 'Mohammad']
c.sort(reverse=False, key=function(c))
print(c)
'''
```

```
[-105, 1, 10, 50, 200, 1000]
['Amin', 'Amir', 'Asefe', 'Iman', 'Mammad']
when reverse is False(default): ['Amin', 'Amir', 'Asefe', 'Iman', 'Mammad']
when reverse is True: ['Mammad', 'Iman', 'Asefe', 'Amir', 'Amin']
6
4
5
2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-59-eaec0e68fa1a> in <module>
     33
     34 c = ['Asefeh', 'Iman', 'Ali', 'Amin', 'Amir', 'Mohammad']
```

```
---> 35 c.sort(reverse=False, key=function(c))
     36 print(c)

TypeError: 'int' object is not callable
```

In [66]:

```
# Modify Lists
a = [1, 2, 3]

a[0] = 'Mammad'
print(a)

a[2] = [100, 200, 300]
print(a)

a.append('jafar')
print(a)

a.insert(5, 'Asefeh')
print(a)

print(a[4])
print(a[5])
```

```
['Mammad', 2, 3]
['Mammad', 2, [100, 200, 300]]
['Mammad', 2, [100, 200, 300], 'jafar']
['Mammad', 2, [100, 200, 300], 'jafar', 'Asefeh']
Asefeh

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-66-aeafea481bac> in <module>
     15
     16 print(a[4])
---> 17 print(a[5])

IndexError: list index out of range
```

In [71]:

```
# Nested Lists
a = [[1, 2, 3], 5, 50, [6, 7, 8, 10]]
print(a)

print(a[0])
print(a[0][2])
print(a[3][2])       # Don't Mistake: a[3, 2]   ----> Should be: a[3][2]
```

```
[[1, 2, 3], 5, 50, [6, 7, 8, 10]]
[1, 2, 3]
3
8
```

In [13]:

```
### Built-in Function
# sum
a = [1, 2, 3]
b = sum(a)
print(b)

a = {1, 2, 3, 4}
b = sum(a)
print(b)

a = (1, 2, 3, 4, 5)
b = sum(a)
print(b)

print("\n")
# len
a = [1, 2, 3]
print(len(a))

print("\n")
# min and max
a = [1, 2, 3]
print(min(a))
print(max(a))

print("\n")
# sorted                    only for numbers
a = [100, -20, 30, 5]
print(sorted(a))

print("\n")
# reversed
a = [100, -20, 30, 5]
print(list(reversed(a)))

print("\n")
# range(value1, value2)
print(range(5))      # 0 1 2 3 4
print(range(1, 5))   # 1 2 3 4

for element in range(5):
    print(element)
```

```
6
10
15


3


1
3


[-20, 5, 30, 100]
```

```
[5, 30, -20, 100]


range(0, 5)
range(1, 5)
0
1
2
3
4
```

Dictionaries

In [21]:

```python
# format_dict = {'key1': 'value1', 'key2': 'value2', ....}
car_exhibition = {
    'car1' : 'red',
    'car2' : 'blue',
    'car3' : 'white'
}

print(car_exhibition)

print("\n")
# Calling
print(car_exhibition['car1'])
print(car_exhibition['car2'])
```

```
{'car1': 'red', 'car2': 'blue', 'car3': 'white'}


red
blue
```

In [24]:
```
# Nested Dictionaries
car_exhibition = {
    'car1' : {'Brand': 'Benz', 'Model': 'E250', 'color': 'red', 'year': 2020
    'car2' : {'Brand': 'BMW', 'Model': 'M6', 'color': 'blue', 'year': 2017}
}

print(car_exhibition)

print("\n")
# Calling
print(car_exhibition['car1'])
print(car_exhibition['car1']['Brand'])
print(car_exhibition['car2']['year'])

'''
import pandas as pd
df = pd.DataFrame(car_exhibition.items())
df
'''
```

```
{'car1': {'Brand': 'Benz', 'Model': 'E250', 'color': 'red', 'year': 2020}, 'car
2': {'Brand': 'BMW', 'Model': 'M6', 'color': 'blue', 'year': 2017}}


{'Brand': 'Benz', 'Model': 'E250', 'color': 'red', 'year': 2020}
Benz
2017
```

Out[24]:
```
'\nimport pandas as pd\ndf = pd.DataFrame(car_exhibition.items())\ndf\n'
```

In [36]:

```
# Complex Nested Dictionaries
Buy = {
    'build_1': {
        'sq-ft': 3721,
        'price($)': 6395000,
        'Address': '162 E 93rd St, New York, NY 10128',
        'features': {
            'Total Rooms': 12,
            'Garden': 'Yes',
            'Pets Allowed': 'No'

        }
    },
    'build_2': {
        'sq-ft': 7281,
        'price': 21800000,
        'Address': '78 Morton, New York, NY 10014',
        'features': {
            'Total Rooms': 14,
            'Roof Deck YN': 'Yes',
            'Pets Allowed': 'Yes'
        }
    }
}

print(Buy)
print("\n")

# Calling
print('build 1:', Buy['build_1'])
print('\n')
print('build 2:', Buy['build_2'])

print("\n")
print(Buy['build_1']['sq-ft'])
print(Buy['build_2']['Address'])

print("\n")
print('build 2:', Buy['build_2']['features'])
print('build 2:', Buy['build_2']['features']['Total Rooms'])
```

```
{'build_1': {'sq-ft': 3721, 'price($)': 6395000, 'Address': '162 E 93rd St, N
ew York, NY 10128', 'features': {'Total Rooms': 12, 'Garden': 'Yes', 'Pets Al
lowed': 'No'}}, 'build_2': {'sq-ft': 7281, 'price': 21800000, 'Address': '78
Morton, New York, NY 10014', 'features': {'Total Rooms': 14, 'Roof Deck YN':
'Yes', 'Pets Allowed': 'Yes'}}}


build 1: {'sq-ft': 3721, 'price($)': 6395000, 'Address': '162 E 93rd St, New
York, NY 10128', 'features': {'Total Rooms': 12, 'Garden': 'Yes', 'Pets Allow
ed': 'No'}}


build 2: {'sq-ft': 7281, 'price': 21800000, 'Address': '78 Morton, New York,
NY 10014', 'features': {'Total Rooms': 14, 'Roof Deck YN': 'Yes', 'Pets Allow
```

```
ed': 'Yes'}}


3721
78 Morton, New York, NY 10014


build 2: {'Total Rooms': 14, 'Roof Deck YN': 'Yes', 'Pets Allowed': 'Yes'}
build 2: 14
```

Dict Methods

```
pop()
popitem()
clear()

update()
setdefault()
copy()

fromkeys()

items()
keys()
values()

get()
```

In [42]:

```
# pop  ----> delete by key
cars= {
    'car1' : 'red',
    'car2' : 'blue',
    'car3' : 'white'
}

cars.pop('car3')
print(cars)

# popitem ----> delete last element
cars= {
    'car1' : 'red',
    'car2' : 'blue',
    'car3' : 'white',
    'car4' : 'jcidjfi',
    'car5' : 'jdnhfi'
}

cars.popitem()
print(cars)

# clear
cars.clear()
print(cars)

# del               ---> WE don't have sth like this
del cars['car4']
print(cars)

del cars      # ---> We have it
print(cars)
```

```
{'car1': 'red', 'car2': 'blue'}
{'car1': 'red', 'car2': 'blue', 'car3': 'white', 'car4': 'jcidjfi'}
{}

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-42-1a909cb3acaf> in <module>
     26
     27 # del
---> 28 del cars['car4']
     29 print(cars)
     30

KeyError: 'car4'
```

In [66]:

```
# update()
cars= {
    'car1' : 'red',
    'car2' : 'blue',
    'car3' : 'white'
}

cars.update({'car5': 'purple'})
print(cars)

# setdefault()
cars.setdefault('car6', 'pink')
print(cars)

cars['car4'] = 'pink'
print(cars)

# copy()
cars= {
    'car1' : 'red',
    'car2' : 'blue',
    'car3' : 'white'
}


x = cars.copy()
print('x: ', x)

```

```
{'car1': 'red', 'car2': 'blue', 'car3': 'white', 'car5': 'purple'}
{'car1': 'red', 'car2': 'blue', 'car3': 'white', 'car5': 'purple', 'car6': 'pin
k'}
{'car1': 'red', 'car2': 'blue', 'car3': 'white', 'car4': 'pink'}
x:  {'car1': 'red', 'car2': 'blue', 'car3': 'white', 'car4': 'pink'}
```

In [60]:
```
# fromkeys
keys = ['key1', 'key2', 'key3']
value = None

x = dict.fromkeys(keys, value)
print(x)

keys = ['key1', 'key2', 'key3']
values = ['value1', 'value2', 'value3']
y = dict.fromkeys(keys, values)
print(y)

print("\n")
# Approaches
# 1
keys = ['key1', 'key2', 'key3']
values = ['value1', 'value2', 'value3']

a = [(1, 'red'), (2, 'blue'), (3, 'white'),]   # --- > zip(value1, value2, .
print(dict(a))

print('new: ', dict(zip(keys, values)))

# 2 ----> in for loop section
```

```
{'key1': None, 'key2': None, 'key3': None}
{'key1': ['value1', 'value2', 'value3'], 'key2': ['value1', 'value2', 'value
3'], 'key3': ['value1', 'value2', 'value3']}


{1: 'red', 2: 'blue', 3: 'white'}
new:  {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
```

In [61]:
```
# items()
cars= {
    'car1' : 'red',
    'car2' : 'blue',
    'car3' : 'white'
}

print(cars.items())

# keys()
print(cars.keys())

# values()
print(cars.values())
```

```
dict_items([('car1', 'red'), ('car2', 'blue'), ('car3', 'white')])
dict_keys(['car1', 'car2', 'car3'])
dict_values(['red', 'blue', 'white'])
```

In [62]:
```
# get   --- > for calling
cars= {
    'car1' : 'red',
    'car2' : 'blue',
    'car3' : 'white'
}


print(cars['car1'])
print(cars.get('car1'))
```

```
red
red
```

In [65]:
```
# Modify Dictionaries
cars= {
    'car1' : 'red',
    'car2' : 'blue',
    'car3' : 'white'
}
print(cars)

cars['car1'] = 'purple'
print(cars)

cars['car2'] = 2
print(cars)
```

```
{'car1': 'red', 'car2': 'blue', 'car3': 'white'}
{'car1': 'purple', 'car2': 'blue', 'car3': 'white'}
{'car1': 'purple', 'car2': 2, 'car3': 'white'}
```

In [73]:

```
a = 1
print(a)
print(float(a))

a = float(a)
print(type(a))

a = 1.1
print(a)
print(int(a))

# complex
a = 1 + 1j
print(a)
# real
# imag


a = 1
b = str(a)
print(b)
print(type(b))

print("\n")
a = (1, 2, 3)
a_list = list(a)
print(a_list)

a_set = set(a)
print(a_set)

b = {1, 2, 3}
b_list = list(b)
print(b_list)

b_tuple = tuple(b)
print(b_tuple)

c = [1, 2, 3]
c_list = list(c)
print(c_list)
```

```
1
1.0
<class 'float'>
1.1
1
(1+1j)
1
<class 'str'>


[1, 2, 3]
{1, 2, 3}
[1, 2, 3]
```

```
(1, 2, 3)
[1, 2, 3]
```

```
Homework 2:

We can't Add element to tuples.  ---> tuple no
trick to add element to tupples.
```