```
##########################################
#######TD6: Itérables:Mutable ou non? ######
###Chaines/Listes/Dictionnaires/Ensembles###
##########################################

#############Les chaines#############
s='bjr tout le monde'
>>> s[0]
-------------------------------------
>>> s[0]='B'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: --------------------------

==> on ne peut pas toucher le contenue d'une chaine
==> les chaines ne sont pas mutables(modifiable, changeable)
==> pour modifier un non mutable, il faut écraser l ancienne version (avec
affectation)

>>> s='B'+s[1:len(s)] #ou bien s='B'+s[1:]
-------------------------------------
>>> s
-------------------------------------
>>> s.upper()
-------------------------------------
>>> s
-------------------------------------
>>> s.lower()
-------------------------------------
>>> 'bla bla'.capitalize()
-------------------------------------
>>> s.lower().capitalize()
-------------------------------------
>>> s.lower().upper().capitalize().replace('o','*')
-------------------------------------
>>> s.capitalize() # nous donne une version(copie) de s bien titré. ( sans toucher
s)
-------------------------------------
>>> 'bla bla'.capitalize()
-------------------------------------

>>> s.endswith('e') # est ce que s se termine par 'e'
-------------------------------------
>>> s.startswith('e') # est ce que s commence par 'e'
-------------------------------------

>>> s.lower().startswith('e').capitalize().replace('o','*')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: ------------------------------------

>>> s.count('u')
-------------------------------------
>>> s.find('e') #reservé que pour les str. elle renvoie index de la premiere
apparution de 'e'; -1 si non
-------------------------------------
>>> s.find('z')
```

```
-------------------------------------
>>> s.index()
-------------------------------------
>>> s
-------------------------------------
>>> s.isalnum()
-------------------------------------
>>> '6315465564'.isalnum()
-------------------------------------
>>> '63154,65564'.isalnum()
-------------------------------------
>>> s.isalpha()
-------------------------------------
>>> s
-------------------------------------
>>> 'dfdsfsdfdsf'.isalpha()
-------------------------------------
>>> 'dfdsfs,dfdsf'.isalpha()
-------------------------------------

>>> ch='';ch0='aa';ch1='bb';ch=ch0+ch1
>>> ch
-------------------------------------
>>> ch=ch+'cc' #est la même que ch+='cc'
-------------------------------------

>>> 't'*5
-------------------------------------

###########Les listes###############
>>> l=[1, 2, 'bjr', 'bjr', (10, 2, 'ab', 5)]
>>> l
l=[1, 2, 'bjr', 'bjr', (10, 2, 'ab', 5)]
>>> l[0]=10
>>> l
[-----, 2, 'bjr', 'bjr', (10, 2, 'ab', 5)]

==> l est modifiable=mutable=changeable

>>> l=l+l
>>> l
-------------------------------------

voir Tp5
len(t)
l.count()
l.index()

 l.append('o') # ajouter 'o' à la fin de l

>>> l.append(10)
>>> l.append([1,2,3])
>>> l
[10, 2, 'bjr', 'bjr', (10, 2, 'ab', 5),
 10, 2, 'bjr', 'bjr', (10, 2, 'ab', 5), 'o', 10, [1, 2, 3]]

>>> l.append(1,2)
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: append() takes exactly one argument (2 given)

==> append prend un seul parameter

l=l+'a' ----> errrrrrrrrrrrrrrrrrrrreur

l.append('a') est la même que l=l+['a']
c'est la même aussi que l+=['a']

#l.extend(ordonnée)
>>> l.extend([1,2,3,5])
>>> l
[10, 2, 'bjr', 'bjr', (10, 2, 'ab', 5), 10, 2, 'bjr',
'bjr', (10, 2, 'ab', 5), 'o', 10, [1, 2, 3], 1, 2, 3, 5]
>>> l.extend((1,2,3,5))
>>> l
[10, 2, 'bjr', 'bjr', (10, 2, 'ab', 5), 10, 2, 'bjr',
'bjr', (10, 2, 'ab', 5), 'o', 10, [1, 2, 3],
1, 2, 3, 5, 1, 2, 3, 5]
>>> l.extend('12345')
>>> l
[10, 2, 'bjr', 'bjr', (10, 2, 'ab', 5), 10, 2, 'bjr',
 'bjr', (10, 2, 'ab', 5), 'o', 10, [1, 2, 3],
  1, 2, 3, 5, 1, 2, 3, 5, '1', '2', '3', '4', '5']

>>> l0
[3, 1, 'e', -5]
>>> l0.extend(1,2,3,4,5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: extend() takes exactly one argument (5 given)


>>> l.reverse() # renverser les éléments
>>> l
['5', '4', '3', '2', '1', 5, 3, 2, 1, 5, 3, 2, 1, [1, 2, 3],
 10, 'o', (10, 2, 'ab', 5), 'bjr', 'bjr', 2, 10,
 (10, 2, 'ab', 5), 'bjr', 'bjr', 2, 10]

comparer les comparables?
l.sort()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '<' not supported between instances of 'int' and 'str'


********************

>>> l0.sort()
--------------------------------------------------
>>> l0.insert(1,'e')
>>> l0
[20, -----, 3, 1, -5]
>>> l0.insert(-1,'e')
>>> l0
```

```
[20, 'e', 3, 1, 'e',-------]
]
>>> l0.pop(1)
'e'
>>> l0
[20, 3, 1, 'e', -5]
>>> l0.pop(100)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: pop index ------ of range

>>> l4=['a','n',10]
>>> l4.pop(1).upper()

>>> l0.remove(20)
>>> l0
[3, 1, 'e', -5]
>
>>> l0.remove(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: list.remove(x): x --------- list

>>> l0
[3, 1, 'e', -5]

>>> [1,2,3]*5
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]

#--------- List en compréhension----------
>>> L=[x**2 for x in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> L[3]
9
#--------- List vide----------
>>> []
>>> bool([])
-----------------------------------
###############Les tupes #################
>>> t=(1,2,'ab',5)

>>> t[0]=10
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' -------------------------

==> on ne peut pas toucher le contenue d'un tuple
==> les tuples ne sont pas mutables(modifiable, changeable)
==> les tuples sont NON mutables
==> pour modifier un non mutable, il faut écraser l ancienne version (avec une
affectation)

>>> (1,2,3)*5
(1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3)

#(10) est l'entier 10
#(10,) est le tuple qui contient l'entier 10
```

```
>>> t=(10,)+t[1:]
>>> t
-------------------------------------
>>> (1+2)
-------------------------------------
>>> -(1)
-------------------------------------
>>> -(1)+2
-------------------------------------
>>> (1,2)
(1, 2)
>>> ()
()
>>> (1,2)+(1)
erreur ---------------------------------------
>>> (1,2)+(1)
-------------------------------------
>>> (10,)
-------------------------------------
>>> type((10,))
<class '------'>
>>> type((10))
<class '-----'>

voir Tp5
len(t)
t.count()
t.index()

#--------- Tuple en compréhension----------
>>> T=(x**2 for x in range(10))
(0, 1, 4, 9, 16, 25, 36, 49, 64, 81)
>>> T[3]
9
#--------- Tuple vide----------
>>> ()
>>> bool(())
-----------------------------------
#########################################################################
#########################Dictionnaire#####################################""
#########################################################################

D={cle_0:val_0,cle_1:val_1,cle_2:val_2,...,cle_n:val_n,}

Exemple: un etudaint est réprésenté par ce dictionnaire
d={'nom':'nour','age':18,'notes':[15,14,10,5]}

for k in d:
    print('la clé est: ',k, ' et sa valeur: ',d[k])

>>> d
{'nom': 'nour', 'age': 18, 'notes': [15, 14, 10, 5]}
>>> d['poids']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: ---------------------------
>>> d['poids']=59.5
```

```
>>> d
{'nom': 'nour', 'age': 18, 'notes': [15, 14, 10, 5], '-----': ----}
>>> d['poids']=59
>>> d
{'nom': 'nour', 'age': 18, 'notes': [15, 14, 10, 5], '-----': -----}
>>> len(d)
-------------------------------------
>>> d.keys()
dict_keys(['------', '------', '-----', '------'])
>>> d.values()
dict_values(['nour', 18, [15, 14, 10, 5], 59])
>>> d.items()
dict_items([('nom', 'nour'), ('age', 18), ('notes', [15, 14, 10, 5]), ('poids',
59)])
>>> for k in d:
      print(k)
nom
age
notes
poids

>>> for k in d:
        print('la clé est: ',k, ' et sa valeur: ',d[k])

la clé est:  nom  et sa valeur:  nour
la clé est:  age  et sa valeur:  18
la clé est:  notes  et sa valeur:  [15, 14, 10, 5]
la clé est:  poids  et sa valeur:  59


>>> d={'etud1':'hamed', 'etude2':'manal', 'etude3':'soumaya', 'etude4':'mhammed',
'etude5':'rania', 'etud6':'ahmed', 'etude':'yosri'}
>> type(d)
<class '--------'>
>>> d['etud1']
-------------------------------------
>>> d['etud6']
-------------------------------------

Exemple: On peut représenter toute liste par un dictionnaire. En effet:
>>> lista=[1,2,3]
>>> lista[0]
1
>>> lista[1]
2
>>> lista[2]
3
>>> dicta={0:1,1:2,2:3}
>>> dicta[2]
3
>>> dicta[1]
2
>>> dicta[0]
1

N.B.: les clés dans un dictionnaire doivent êtes non mutable (hashable).
>>> {2:1}
```

```
{2: 1}
>>> {'2':1}
{'2': 1}
>>> {[2]:1}
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: -------------------
>>> {(2):1}
{2: 1}
>>> {(1,2):1}
{(1, 2): 1}
>>> {{1,2}:1}
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: --------------------

>>> {2:1,1:2}=={1:2,2:1}
----------------------------------------
>>> {2:1,1:2,1:1}
----------------------------------------


#--------- Dictionnaire en compréhension----------
>>> D={x:x**2 for x in range(10)}
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
>>> D[3]
9
#--------- Dictionnaire vide----------
>>> {}
>>> bool({})
----------------------------------------

################################################################################
##########################Ensembles#####################################""
################################################################################
s={1}
s={1,'bjr',1,'a',(2,3),1}

un ensemble est itérable

>>> for e in s:
...     print(e)
(2, 3)
1
bjr
a

>>> s
{(2, 3), 1, 'bjr', 'a'}
>>> s={1,'bjr',1,'a',(2,3),1,True}
>>> s
{(2, 3), 1, 'bjr', 'a'}
>>> {True}
{True}
>>> True+5*True
6
>>> type(s)
```

```
<class '------'>

N.B.: La fonction set() est un outil fort pour supprimer les doublons dans un
itérable.

>>> set('jhjhjhjhjhjhjhjhjhjh')
{----,-----}
>>> set([1,2,3,1,2,1,5,1,41,1,74,1])
----------------------------------------
>>> set((1,2,3,1,2,1,5,1,41,1,74,1))
----------------------------------------

>>> s
{(2, 3), 1, 'bjr', 'a'}
>>> s[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object is not ----------------

>>> {1,'a'}<s #inclus?
True
>>> {1,'a'}<=s
True

>>> {1}|{2,3} #Union?
{1, 2, 3}

>>> s.union({58})   #s.union(<itérable>)
{(2, 3), 1, 'bjr', 'a', 58}
>>> s
{(2, 3), 1, 'bjr', 'a'}
>>> s&{1,'b','a'} #intersection
{1, 'a'}
>>> s.intersection({1,'b','a'}) #s1.intersection(<itérable>)
{1, 'a'}
>>> s
{(2, 3), 1, 'bjr', 'a'}

>>> l
[1, 2, 'bjr', True, 2, (1, 2), 1, 1, 2, 'a', 1]
>>> set(l)
{1, 'bjr', 2, (1, 2), 'a'}
>>> list(set(l))
[1, 'bjr', 2, (1, 2), 'a']
>>> [e for e in set(l)]
[1, 'bjr', 2, (1, 2), 'a']
>>> len(s)
4

#--------- ensemble en compréhension----------
>>> e={x**2 for x in range(10)}
>>> e[3]
error:.....................................

#--------- ensemble vide----------
>>> set()
>>> bool(set)
```