```
#########################################################################
#################TP-TD5 :----Les sequences-----###################
#########################################################################

# une sequence=une chaine=une suite
Exemples:
s:chaines de caractères
l:listes
r:range(10)
t:tuple

########################################
########chaines de caractères###########
########################################

s0="bonjour tout le monde"
s1='bonjour tout le monde'
s4='''bonjour tout le monde'''
s0==s1

>>> s="it's a good day"
>>> s1='it\'s a good day' # <\'> c'est une cote
>>> s==s1
-------------------------------------
>>> s2="je m'appelle \"arwa\""
>>> print(s)
-------------------------------------
s3='''il'fait beau.
je vais à l'école
je rentre.
'''
>>> type(s)
----------------------------------------
>>> s[0]
----------------------------------------
>>> s[1]
----------------------------------------
>>> s[17]
 erreur:-----------------
>>> len(s)
----------------------------------------
>>> s[len(s)-1]
----------------------------------------
>>> s[len(s)-2]
----------------------------------------
>>> s[-1]#represent le premier caractère sense inversé
----------------------------------------
>>> s[len(s)-1] ou bien s[-1] le dernier dans s
----------------------------------------
>>> s[len(s)-2] ou bien s[-2] avant le dernier dans s
----------------------------------------
>>> s3[0]
----------------------------------------
>>> s3[1]
----------------------------------------
>>> s3[12]
----------------------------------------
>>> s3[13]
# \n est le caractère 'new line': c'est un caractère non imprimable
```

```
>>> s1+s2
----------------------------------------
>>> s2+s1
----------------------------------------
>>> new_s=s1+'? '+s2+'!'
>>> print(new_s)
----------------------------------------
>>> 'd' in s
----------------------------------------
>>> 'z' in s
----------------------------------------
>>> 'd' not in s #ou bien: not ('d' in s)
----------------------------------------
>>> 'z' not in s
----------------------------------------
# s.index('o') retourne l'indice de la prèmière apparition de 'o'
>>>  s.index('o')
----------------------------------------
>>>  s.index('oo')
----------------------------------------
>>>  s.index('z')
  erreur ----------------------------------------
>>> 'jhjqhgqjh'.index('q')
----------------------------------------
# s.count('a') retourne le nombre d'occurence de 'a'
>>>  s.count('a')
----------------------------------------
>>>  s.count('z')
----------------------------------------
# s.find('o') #retourne l'indice de l'apparition de la prèmière apparition de 'o'
>>>  s.find('o')
----------------------------------------
>>>  s.find('z')
----------------------------------------

######slicing#####
s[a:b] # c'est la sous chaine qui commence de l'indice a inclue jusqu'à l'indice b-
1
s[a:] # c'est la sous chaine qui commence de l'indice a inclue jusqu'à la fin
s[:b] # c'est la sous chaine qui commence du début jusqu'à la position b-1
s[a:b:c] # c'est la sous chaine qui commence de l'indice a inclue jusqu'à l'indice
b-1 avec pas 'saut' de c
s[::] # c'est la sous chaine qui commence du debut jusqu'à la fin
s[::c] # c'est la sous chaine qui commence du debut jusqu'à la fin avec saut de c

>>> s[0:2]
----------------------------------------
>>> s[1:2] ou s[1]
----------------------------------------
>>> s[1:6]
----------------------------------------
>>> s[-3:-1]
----------------------------------------
>>> s[10:-2]
----------------------------------------
>>> s[:]
----------------------------------------
>>> s[::]
----------------------------------------
```

```
>>> s[::-1]
----------------------------------------
# uns sous chaine est une chaine de caractères
>>> len(s[:6])
----------------------------------------
>>> s[:6].index('o')
----------------------------------------
>>> s[3:10].find('o')
----------------------------------------
>>> s[0]='I'
 erreur---------------------------------

==> s est non mutable

########################################
########Liste d'éléments###########
########################################

>>> l=[1,2,'bjr',True,(1,2)]
>>> l0=['a','b',-1j,2.5]
>>> l1=['c',2]
>>> print(l)
----------------------------------------
>>> type(l)
----------------------------------------
>>> l[0]
----------------------------------------
>>> l[1]
----------------------------------------
>>> l[17]
 erreur: --------------------------------
>>> len(l)
----------------------------------------
>>> l[len(l)-1]
----------------------------------------
>>> l[len(l)-2]
----------------------------------------
l[-1]#represent le premier caractère sense inversé
l[len(l)-1] ou bien l[-1] le dernier dans l
l[len(l)-2] ou bien l[-2] avant le dernier dans l

>>> l0+l1
----------------------------------------
l1+l0
----------------------------------------
>>> new_l=l1+l0
>>> print(new_l)
>>> 1 in l
----------------------------------------
>>> 'z' in l
----------------------------------------
>>> 1 not in l #ou bien: not (1 in l)
----------------------------------------
>>> 'z' not in l
----------------------------------------
# l.index(ele) retourne l'indice de la prèmière apparition de ele
l.index('a')
>>>  l.index(1)
----------------------------------------
```

```
>>>  l.index('z')
  erreur ------------------------------
>>>  [1,2,3].index(3)
----------------------------------------
# l.count('a') retourne le nombre d'occurence de 'a'
>>>  l.count('a')
 ----------------------------------------
>>>  l.count(2)
 ----------------------------------------

######slicing#####
l[a:b] # c'est la sous liste qui commence de l'indice a inclue jusqu'à l'indice b-1
l[a:] # c'est la sous liste qui commence de l'indice a inclue jusqu'à la fin
l[:b] # c'est la sous liste qui commence du début jusqu'à la position b-1
l[a:b:c] # c'est la sous liste qui commence de l'indice a inclue jusqu'à l'indice
b-1 avec pas 'saut' de c
l[::] # c'est la sous liste qui commence du debut jusqu'à la fin
l[::c] # c'est la sous liste qui commence du debut jusqu'à la fin avec saut de c

>>> l[0:2]
 ----------------------------------------
>>> l[1:2] ou l[1]
----------------------------------------
>>> l[1:6]
erreur  ----------------------------------------
>>> l[-3:-1]
 ----------------------------------------
>>> l[10:-2]
 ----------------------------------------
>>> l[::-1]
----------------------------------------
# une sous liste est une liste de caractères
>>> len(l[:6])
----------------------------------------

>>> l[:6].index('o')
----------------------------------------
>>> l[0:5:2]
----------------------------------------
>>> l[::]
----------------------------------------
>>> l[::2]
----------------------------------------
>>> l[::3]
----------------------------------------

>>>l[0]=10 #on peut modifier le contenue de la liste
=> Une liste est dite "mutable"

l=l+[2]
>>> l.append(3) #ajoute 3 à l à la fin
----------------------------------------
>>> l.extend([1,2,3])#l=l+[1,2,3] ou l+=[1,2,3]
----------------------------------------
>>> l.extend('jkhjkjhkj')
----------------------------------------
>>>i=2
>>> l.pop(i) #supprime l'element d'indice i sinon erreur
----------------------------------------
```

```
>>> l.remove(ele) #supprime l'element ele s il existe sinon erreur
----------------------------------------
>>> l.reverse() #trier les éléments de l
----------------------------------------
>>> l.sort() #ordonner les listes homogènes (réelles ou str)
----------------------------------------

>>> '1'<'9'<'A'<'Z'<'a'<'z'
----------------------------------------

=> list est mutable

############################################################
######### Même raisonnement pour tuples et ranges##########
############################################################

>>> t=(1,2)
>>> type(t)
----------------------------------------
>>> len(t)
----------------------------------------
>>> t+t
----------------------------------------
>>> t[0]
----------------------------------------
>>> t[-1]
----------------------------------------
>>>t[0]=10
----------------------------------------

==>t est non mutable

############################################################
########### Même raisonnement pour les ranges#############
############################################################

>>> r=range(10,100)

#pour voir le contenue de r, on'utilise le boucle for
>>> for i in r:
    print('le ',i,'emme élément est: ',i)
----------------------------------------
----------------------------------------
----------------------------------------
----------------------------------------
# ou bien
>>> print(list(r))
----------------------------------------

>>> r1=range(10,100,3)
print(list(r1))
----------------------------------------

==>r n est pas mutable

############################################
##########présence/absence de données########
############################################
```

```
# cas ou l'info est absente
#pour le type nemerique
>>> None
#pour le type str
>>> ''
# pour le type tuple
>>> ()
# pour le type liste
[]

#bool(<d>): retourn True si <d> contient de l'info, false si non.
>>> bool('')
-------------------------------------
>>> bool([])
-------------------------------------
>>> bool(())
-------------------------------------
>>> x==''
True
>>> len(x)
0
#Est ce que x contient de l'info?
>>> if x==':
       print('bla bla')
-------------------------------------
#ou bien:
>>> if len(x)==0:
       print('bla bla')
-------------------------------------

#ou bien:

>>> if x:
       print('bla bla')
-------------------------------------

>>> if not x: print(10)
-------------------------------------

Exemple illustratif

>>> x=input('donner une info')
>>> if (x==''):
       print("l'utilisateur refuse de donner une info")

2 eme method: sachant que '' <=> absence de l'info <=> bool('')=False
>>> if not x:
       print("l'utilisateur refuse de donner une info")
```