Exercice 1 : Problème de casse ?

En informatique, la casse désigne le fait de distinguer les lettres majuscules des lettres minuscules.

Le code ci-contre permet de créer 2 dictionnaires :

```
def casse() :
    minuscules = 'abcdefghijklmnopqrstuvwxyz'
    majuscules = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    upper = {}
    lower = {}
    for i in range(26) :
        l = minuscules[i]
        L = majuscules[i]
        upper[l] = L
        lower[L] = l
    return upper , lower

# main --------------------
upper , lower = casse()
```

Ce code est exécuté. Pour chacune des commandes données dans le tableau ci-dessous, indiquer le résultat de l'exécution :

Commande	Résultat
`upper['a']`	`>>> upper['a']` `'A'`

`upper['A']`	`>>> upper['A']`
	`Traceback (most recent call last):`
	`  File "<console>", line 1, in <module>`
	`KeyError: 'A'`

`lower['A']`	`>>> lower['A']` `'a'`
`len(upper)`	`>>> len(upper)` `26`
`for c in upper :` `  print(c , end = " ")`	`>>> (executing file "exercice1-2.py")` `a b c d e f g h i j k l m n o p q r s t u v w x y z`
`for c in upper :` `  print(c , upper[c] , end = " ")`	`>>> (executing file "exercice1-2.py")` `a A b B c C d D e E f F g G h H i I j J k K l L m M n N o O p P q Q r R s S t T u U v V w W x X y Y z Z`

`upper.clear()`
`print(upper)`
	`>>> upper.clear()`
	
	`>>> print(upper)`
	`{}`

On aurait pu écrire le code de la fonction casse() de la manière suivante :

```
def casse_bis() :
    upper = {   "a":"A" , "b":"B" , "c":"C" , "d":"D" , "e":"E" , "f":"F" ,
                "g":"G" , "h":"H" , "i":"I" , "j":"J" , "k":"K" , "l":"L" ,
                "m":"M" , "n":"N" , "o":"O" , "p":"P" , "q":"Q" , "r":"R" ,
                "s":"S" , "t":"T" , "u":"U" , "v":"V" , "w":"W" , "x":"X" ,
                "y":"Y" , "z":"Z"
            }
    lower = {   "A":"a" , "B":"b" , "C":"c" , "D":"d" , "E":"e" , "F":"f" ,
                "G":"g" , "H":"h" , "I":"i" , "J":"j" , "K":"k" , "L":"l" ,
                "M":"m" , "N":"n" , "O":"o" , "P":"p" , "Q":"q" , "R":"r" ,
                "S":"s" , "T":"t" , "U":"u" , "V":"v" , "W":"w" , "X":"x" ,
                "Y":"y" , "Z":"z"
            }
    return upper , lower
```

Exercice 2 : Changer les lettres d'un mot ?

On complète le code précédent :

1- Quelle valeur contient la variable *m* après exécution ?

m contiendra la chaine de caractère :

'BONJOUR'

2- Compléter le tableau ci-dessous donnant le contenu des variables au cours de l'exécution :

mot	*new*	*c*
'bonjour'	' '	'b'
	' ' + 'B' ='B'	'o'
	'B' + 'O' ='BO'	'n'
	'BO' + 'N' ='BON'	'j'
	'BON' + 'J' ='BONJ'	'o'
	'BONJ' + 'O' ='BONJO'	'u'
	'BONJO' + 'U' ='BONJOU'	'r'
	'BONJOU' + 'R' ='BONJOUR'	

```
def casse() :
    minuscules = 'abcdefghijklmnopqrstuvwxyz'
    majuscules = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    upper = {}
    lower = {}
    for i in range(26) :
        l = minuscules[i]
        L = majuscules[i]
        upper[l] = L
        lower[L] = l
    return upper , lower

def majuscule(mot) :
    new = ''
    for c in mot :
        new = new + upper[c]
    return new

# main --------------------
upper , lower = casse()
m = majuscule("bonjour")
```

Exercice 3 : L'informatique ça peut servir !

1- Quel résultat à l'écran donne l'exécution de ce code ?

```
>>> (executing file "exercice3.py")
i love you
ich liebe dich
io amore voi
```

```
def anglais():
    uk = {}
    uk['je'] = 'i'
    uk['aime'] = 'love'
    uk['toi'] = 'you'
    return uk

def allemand():
    d = {'je':'ich' , 'aime' : 'liebe' , 'toi' : 'dich'}
    return d

def italien():
    d = {}
    d['je'] = "io"
    d['aime'] = 'amore'
    d['toi'] = 'voi'
    return d

def traduction(m1 , m2 , m3):
    print(uk[m1],uk[m2],uk[m3])
    print(g[m1],g[m2],g[m3])
    print(it[m1],it[m2],it[m3])

# main
uk = anglais()
g = allemand()
it = italien()
traduction("je","aime","toi")
```

2- Le contenu des dictionnaire est accessible à l'intérieur de la fonction *traduction()*. Pourquoi ?

Les dictionnaires uk, g, it sont retournés dans le programme principal lors de l'exécution des lignes :

```
uk = anglais()
g = allemand()
it = italien()
```

Ces dictionnaires prennent ainsi le statut de variable globale et sont alors accessibles en lecture et en écriture dans la fonction *traduction()* qui est exécutée ensuite.