Exercice 1 : Qu'obtient-on en exécutant ces codes ?

Code	*Shell après exécution*
`list = [2*i for i in range(6)]` `print(list)`	`[0, 2, 4, 6, 8, 10]`

`list = [2*i for i in range(6)]`
`print(list[-1])`
`print(len(l))`
	`10`
	`7`

`list = [2*i for i in range(6)]` `print(list[1:])`	`[2, 4, 6, 8, 10]`
`list = [2*i for i in range(6)]` `print(list[:4])`	`[0, 2, 4, 6]`
`list = [2*i for i in range(6)]` `print(list[1:4])`	`[2, 4, 6]`

`l = [2*i for i in range(6)]`
`del(l[1])`
`print(l)`
	`[0, 4, 6, 8, 10]`

`l = [2*i for i in range(6)]`
`del(l[1:4])`
`print(l)`
	`[0, 8, 10]`

`l1 = [0,1,2,3]`
`l2 = [4,5,6]`
`l = l1 + l2`
`print(l)`
	`[0, 1, 2, 3, 4, 5, 6]`

Exercice 2 : Qu'obtient-on en exécutant ces codes ?

- Exemple a :

```
# Fonctions
def bonne_annee(l) :
    for e in l :
        print(e,end = " ")
# Main
bonne_annee(["je","vous","souhaite","une","bonne","annee"])
```

Donne :

```
je vous souhaite une bonne annee
```

- Exemple b :

```
# Fonctions
def dis_moi(l,code) :
    return l[0]+l[code]

# Main
message = dis_moi(["je t'aime ","pas du tout",
                   "un peu","passionnément"],1)
print(message)
message = dis_moi(["je t'aime ","pas du tout",
                   "un peu","passionnément"],0)
print(message)
```

Donne :

```
je t'aime pas du tout
je t'aime je t'aime
```

- Exemple c :

```
# Fonctions
def nsi_au_bac(liste_notes,liste_appreciation,i) :
    note = liste_notes[i]
    appreciation = liste_appreciation[i]
    return "note : "+ note + " , appréciation : " + appreciation

# Main
a = ["2/20" , "8/20" , "14/20" , "19/20"]
b = ["Nullissime" , "Normal" , "Très bien" , "Tricheur"]

resultat_bac_2022 = nsi_au_bac(a,b,len(a)-1)
print(resultat_bac_2022)
```

Donne :

```
note : 19/20 , appréciation : Tricheur
```

- Exemple d :

```
# Fonctions
def tri(liste) :
    l = []
    for i in range(len(liste)) :
        if liste[i] > 10 :
            l.append(liste[i])
    return l

# Main
l = [18 , 4 , 9 , 14 , 1 , 13]
l_retour = tri(l)
print(l_retour)
print(tri([0,14,1,0,5,9,4,5,3,1,2,3]))
```

Donne :

```
[18, 14, 13]
[14]
```

- Exemple e :

```
# Fonctions
def bornes(l) :
    a = l[0]
    b = l[0]
    for i in range(len(l)) :
        if l[i] > a :
            a = l[i]
        if l[i] < b :
            b = l[i]
    return b , a

# Main
l = [18 , 4 , 9 , 14 , 1 , 13]
a_retour , b_retour = bornes(l)
print(a_retour , b_retour)
print(bornes([0,14,1,0,5,9,4,5,3,1,2,3]))
```

Donne :

```
1 18
(0, 14)
```

Exercice 4 : Une liste de listes

On construit ci-dessous, la liste nommée *liste*. On exécute ce code et on exécute les instructions suivantes dans le *shell*. Que donnent-t-elles :

```
intitules = ["Prénom" , "Nom" , "birth day", "job actuel"]
l1 = ["Emmanuel" , "MACRON" , 1977 , "Président"]
l2 = ["Donald" , "TRUMP" , 1946 , "Ex-Président"]

liste = [intitules , l1 , l2 , ["zaza","WINDSOR",1926,"Queen"]]
```

Instruction	***Shell après exécution***
`>>> liste[2]`	`>>> liste[2]` `['Donald', 'TRUMP', 1946, 'Ex-Président']`
`>>> liste[2][2]`	`>>> liste[2][2]` `1946`
`>>> len(liste)`	`>>> len(liste)` `4`
`>>> len(liste[1])`	`>>> len(liste[1])` `4`
`>>> len(liste[1][1])`	`>>> len(liste[1][1])` `6`
`>>> "MACRON"[2]`	`>>> "MACRON"[2]` `'C'`
`>>> "MACRON"[1:3]`	`>>> "MACRON"[1:3]` `'AC'`
`>>> len("MACRON")`	`>>> len("MACRON")` `6`
`>>> liste[1][1][2]`	`>>> liste[1][1][2]` `'C'`

`>>> liste[1][1][1:3]`	`>>> liste[1][1][1:3]` `'AC'`
`>>> "N" in "MACRON"`	`>>> "N" in "MACRON"` `True`
`>>> "CO" in "MACRON"`	`>>> "CO" in "MACRON"` `False`
`>>liste[1][1][2] == "C"`	`>>liste[1][1][2] == "C"` `True`

⇨ On rajoute dans le code du fichier :

```
liste[0][2] = "date de naissance"
liste[0][3] = "Dernier emploi"
print(liste[0])
```

Qu'obtient-on dans le shell ?

```
['Prénom', 'Nom', 'date de naissance', 'Dernier emploi']
```

⇨ On remplace à présent les 3 lignes précédentes par :

```
for l in liste :
    print(l[0] ,end=" ")
```

Qu'obtient-on dans le shell ?

```
Prénom Emmanuel Donald zaza
```

Exercice 5 : Une chaine de caractère peut être traitée comme une liste

La chaine de caractère *mot* est définie dans un fichier par :

```
mot = "Bonjour les amis, je suis un humainoïde"
```

On exécute ce code et on exécute les instructions suivantes dans le *shell*. Que donnent-t-elles :

Instruction	*Shell après exécution*
`>>> len(mot)` `39`	`>>> len(mot)` `39`
`>>> mot[5]`	`>>> mot[5]` `'u'`
`>>> mot[2:7]+mot[-1]`	`>>> mot[2:7]+mot[-1]` `'njoure'`
`>>> mot[0] == "A"`	`>>> mot[0] == "A"` `False`

`>>> mot[0] = "A"`	`>>> mot[0] = "A"`
	`Traceback (most recent call last):`
	`  File "<console>", line 1, in <module>`
	`TypeError: 'str' object does not support item assignment`

`>>> mot.split()` `['Bonjour', 'les', 'amis,', 'je', 'suis', 'un', 'humainoïde']`	
`>>> mot.split(",")` `['Bonjour les amis', ' je suis un humainoïde']`	
`>>> mot.split("u")` `['Bonjo', 'r les amis, je s', 'is ', 'n h', 'mainoïde']`	

⇨ On exécute le code suivant :

```
mot = "Bonjour les amis, je suis un humainoïde"

for i in range(len(mot)) :
    if i%2 == 0 : print(mot[i],end="")
```

Qu'obtient-on dans le shell ?

```
Bnorlsai,j usu uanïe
```