# INFO-H-100 - Programmation
# TP 5 — Boucles `for` et chaînes de caractères

Lorsque l'exercice demande d'écrire une fonction, écrivez la fonction demandée et testez la avec plusieurs valeurs pertinentes.

**Ex. 1.** Dans l'invite de commande, exécutez le code suivant. Assurez-vous de bien le comprendre.

```
message = "Bonjour le monde"
voyelles = "aeiouy"
for letter in message:
    if letter == ' ':
        print "espace"
    elif letter in voyelles:
        print letter.upper()
    else:
        print letter
```

**Ex. 2.** Ecrire une fonction qui teste si un caractère est en minuscule.

```
>>> est_minuscule('q')
True
>>> est_minuscule('S')
False
>>> est_minuscule('!')
False
>>> est_minuscule('4')
False
```

**Ex. 3.** Ecrire une fonction qui teste si un mot est en minuscule. On considère que le mot n'est constitué que de lettres non accentuées.

```
>>> mot_minuscule("bonjour")
True
>>> mot_minuscule("Hello")
False
```

**Ex. 4.** Ecrire une fonction qui crypte une chaîne de caractères en remplaçant chaque caractère par le caractère 3 positions plus loin dans l'ordre ASCII. Exemple :

```
>>> crypt("hello")
'khoor'
>>> crypt("Python is an easy to learn, powerful programming language.")
'S|wkrq#lv#dq#hdv|#wr#ohduq/#srzhuixo#surjudpplqj#odqjxdjh1'
```

**Ex. 5.** Soit la chaîne de caractères `informatique`, écrire des expressions utilisant les tranches (*slices*) et la concaténation pour produire les résultats suivants :

- `info`
- `format`
- `tique`
- `informatiste`
- `infomatique`
- `formation`

**Ex. 6.** Ecrire une fonction qui compte le nombre d'apparitions d'un caractère dans une chaîne donnée.

```
>>> count_chars("bonjour", "j")
1
>>> count_chars("mississipi", "s")
4
```

**Ex. 7.** Ecrire une fonction qui renvoie l'indice de la première apparition d'un caractère dans une chaîne, ou `-1` si le caractère n'apparaît pas.

**Ex. 8.** Ecrire une fonction `replace1` qui reçoit en argument une chaîne de caractères et deux caractères, et qui renvoie une nouvelle chaîne de caractères où la première occurrence du premier caractère a été remplacée par le second caractère. Par exemple :

```
>>> replace1("hello", "e", "a")
'hallo'
>>> replace1("mississipi", "i", "g")
'mgssissipi'
```

**Ex. 9.** Comme l'exercice 8, sauf que toutes les occurrences doivent être remplacées :

```
>>> replace_all("hello", "e", "a")
'hallo'
>>> replace_all("bonjour", "u", "z")
'bonjozr'
>>> replace_all("mississipi", "i", "g")
'mgssgssgpg'
```

**Ex. 10.** Ecrire une fonction qui reçoit une chaîne de caractère et qui renvoie cette chaîne de caractère renversée :

```
>>> reverse("hello")
'olleh'
>>> reverse("Esope reste ici et se repose")
'esoper es te ici etser eposE'
```

**Ex. 11.** Ecrire une fonction qui, en fonction d'un $n$ donné, affiche le triangle suivant :

```
1
1 2
1 2 3
1 2 3 4
...
1 2 3 4 5 ... n
```

**Ex. 12.** Ecrire une fonction qui reçoit en paramètre deux chaînes de caractères `ch1` et `ch2` et qui renvoie la position de la première occurrence de `ch2` dans `ch2`, et `-1` si `ch2` n'est pas une sous-chaîne de `ch1`. Exemple :

```
>>> find("hello", "hello")
0
>>> find("hello", "ello")
1
>>> find("hello", "lo")
3
>>> find("hello", "bonjour")
-1
>>> find("hello", "hell")
0
>>> find("hello", "ll")
2
```

**Ex. 13.** Ecrire une fonction qui reçoit deux chaînes de caractères et qui renvoie la première chaîne dans laquelle la première occurrence de la seconde a été retirée. Exemple :

```
>>> remove("hello", "hell")
'o'
>>> remove("Ceci n'est pas une chaine de caracteres", "Ceci n'est pas ")
'une chaine de caracteres'
>>> remove("bonjour", "salut")
'bonjour'
>>> remove("hello hello", "llo")
'he hello'
```

**Ex. 14.** Ecrire une fonction qui reçoit deux chaînes de caractères et qui renvoie la première chaîne à laquelle ont été retirées toutes les occurrences de la seconde chaîne. Exemple :

```
>>> remove_each("hello", "l")
'heo'
>>> remove_each("String literals are written in single or double quotes", "String literals")
' are written in single or double quotes'
```

**Ex. 15.** Ecrire une fonction qui enlève tous les espaces d'une chaîne de caractères. Exemple :

```
>>> remove_spaces("It has efficient high-level data structures and a simple...")
'Ithasefficienthigh-leveldatastructuresandasimple...'
```

**Ex. 16.** Ecrire une fonction qui détecte si la chaîne de caractère passée en argument est un palindrome. Pour rappel, un palindrome est une phrase qui peut se lire dans les deux sens : l'ordre des lettres est symétrique (à l'exception des majuscules, accents, espaces et poncutations ; dans cet exercice, on suppose qu'il n'y a ni accent, ni ponctuation).

```
>>> palindrome("hello")
False
>>> palindrome("Esope reste ici et se repose")
True
```

# INFO-H-100 - Programmation
# TP 5 — Boucles `for` et chaînes de caractères
Corrections

---

### Solution de l'exercice 1:

Le code proposé va imprimer, ligne par ligne, chaque lettre de la chaîne de caractères, en remplaçant les espaces par le mot espace et les voyelles par leur majuscule.

### Solution de l'exercice 2:

```
def est_minuscule(char):
    return 'a' <= char <= 'z'
```

### Solution de l'exercice 3:

```
def mot_minuscule(mot):
    for lettre in mot:
        if not est_minuscule(lettre):
            return False
    return True
```

### Solution de l'exercice 4:

```
def crypt(string):
    result = ""
    for letter in string:
        result += chr(ord(letter)+3)
    return result
```

### Solution de l'exercice 5:

```
s = "informatique"
print s[0:4]
print s[2:8]
print s[7:]
print s[0:9] + "ste"
print s[0:4] + s[5:]
print s[2:9] + "on"
```

### Solution de l'exercice 6:

```
def count_chars(string,char):
    count = 0
    for c in string:
        if c == char: count += 1
    return count
```

### Solution de l'exercice 7:

```
def find1(string, char):
    for i in range(len(string)):
        if string[i] == char:
            return i
    return -1
```

### Solution de l'exercice 8:

```
def replace1(string, old, new):
    for i in range(len(string)):
        if string[i] == old:
            return string[:i] + new + string[i+1:]
    return string
```

Sans slice et avec un seul `return` :

```
def replace1(string, old, new):
    res = ""
    found = False
    for letter in string:
        if found or letter != old:
            res = res + letter
        else:
            found = True
            res = res + new
    return res
```

## Solution de l'exercice 9:

```
def replace_all(string, old, new):
    while string != replace1(string,old,new):
        string = replace1(string,old,new)
    return string
```

Définition itérative n'utilisant pas `replace1` :

```
def replace_all(string, old, new):
    res = ""
    for letter in string:
        if letter == old:
            res += new
        else:
            res += letter
    return res
```

Définition récursive utilisant `replace1` :

```
def replace_all(string, old, new):
    temp = replace1(string, old, new)
    if temp != string:
        return replace_all(temp, old, new)
    else:
        return string
```

## Solution de l'exercice 10:

```
def reverse(orig):
    res = ""
    length = len(orig)
    for i in range(length):
        res += orig[length-1-i]
    return res
```

En utilisant les slices :

```
def reverse(orig):
    return orig[-1::-1]
```

## Solution de l'exercice 11:

```
def triangle(n):
    for i in range(n):
        for j in range(i+1):
            print j+1,
        print
```

## Solution de l'exercice 12:

```
def find(ch1, ch2):
    length = len(ch2)
    for pos in range(len(ch1) - length + 1):
        if ch1[pos:pos+length] == ch2:
            return pos
    return -1
```

## Solution de l'exercice 13:

```
def remove(orig, pattern):
    pos = find(orig, pattern)
    length = len(pattern)
    if pos != -1:
        return orig[:pos] + orig[pos+length:]
    else:
        return orig
```

## Solution de l'exercice 14:

```
def remove_each(orig, pattern):
    while remove(orig, pattern) != orig:
        orig = remove(orig, pattern)
    return orig
```

## Solution de l'exercice 15:

```
def remove_spaces(orig):
    return remove_each(orig, " ")
```

## Solution de l'exercice 16:

```
def palindrome(string):
    return remove_spaces(string.upper()) == reverse(remove_spaces(string.upper()))
```