Exercice 1

Écrire une fonction `recherche` qui prend en paramètres `caractere`, un caractère, et `mot`, une chaîne de caractères, et qui renvoie le nombre d'occurrences de `caractere` dans `mot`, c'est-à-dire le nombre de fois où `caractere` apparaît dans `mot`.

Exemples :

```
>>> recherche('e', "sciences")
2
>>> recherche('i',"mississippi")
4
>>> recherche('a',"mississippi")
0
```

Exercice 2

Soit le couple `(note,coefficient)`:

- `note` est un nombre de type flottant (*`float`) compris entre 0 et 20 ;
- `coefficient` est un nombre entier positif.

Les résultats aux évaluations d'un élève sont regroupés dans une liste composée de couples `(note,coefficient)`.

Écrire une fonction `moyenne` qui renvoie la moyenne pondérée de cette liste donnée en paramètre.

Par exemple, l'expression `moyenne([(15,2),(9,1),(12,3)])` devra renvoyer le résultat du calcul suivant :

$$\frac{2 \times 15 + 1 \times 9 + 3 \times 12}{2 + 1 + 3} = 12{,}5$$

Exercice 3

Écrire une fonction `RechercheMinMax` qui prend en paramètre un tableau de nombres non triés `tab`, et qui renvoie la plus petite et la plus grande valeur du tableau sous la forme d'un dictionnaire à deux clés 'min' et 'max'. Les tableaux seront représentés sous forme de liste Python.

Exemples :

```
>>> tableau = [0, 1, 4, 2, -2, 9, 3, 1, 7, 1]
>>> resultat = rechercheMinMax(tableau)
>>> resultat
{'min': -2, 'max': 9}
```

```
>>> tableau = []
>>> resultat = rechercheMinMax(tableau)
>>> resultat
{'min': None, 'max': None}
```

Exercice 4

Écrire une fonction `conv_bin` qui prend en paramètre un entier positif `n` et renvoie un couple `(b,bit)` où :

- `b` est une liste d'entiers correspondant à la représentation binaire de `n;`
- `bit` correspond aux nombre de bits qui constituent `b`.

Exemple :

```
>>> conv_bin(9)
([1,0,0,1],4)
```

Aide :

- l'opérateur `//` donne le quotient de la division euclidienne : `5//2` donne `2 ;`
- l'opérateur `%` donne le reste de la division euclidienne : `5%2` donne `1 ;`
- `append` est une méthode qui ajoute un élément à une liste existante :

  Soit `T=[5,2,4]`, alors `T.append(10)` ajoute 10 à la liste `T.` Ainsi, `T` devient `[5,2,4,10].`

- `reverse` est une méthode qui renverse les éléments d'une liste.

  Soit `T=[5,2,4,10].` Après `T.reverse()`, la liste devient `[10,4,2,5].`

On remarquera qu'on récupère la représentation binaire d'un entier `n` en partant de la gauche en appliquant successivement les instructions :

```
b = n%2
n = n//2
```

répétées autant que nécessaire.

Exercice 5

L'occurrence d'un caractère dans un phrase est le nombre de fois où ce caractère est présent.

Exemples :

- l'occurrence du caractère 'o' dans 'bonjour' est 2 ;
- l'occurrence du caractère 'b' dans 'Bébé' est 1 ;
- l'occurrence du caractère 'B' dans 'Bébé' est 1 ;
- l'occurrence du caractère ' ' dans 'Hello world !' est 2.

On cherche les occurrences des caractères dans une phrase. On souhaite stocker ces occurrences dans un dictionnaire dont les clefs seraient les caractères de la phrase et les valeurs l'occurrence de ces caractères.

Par exemple : avec la phrase 'Hello world !' le dictionnaire est le suivant :

```
{'H': 1,'e': 1,'l': 3,'o': 2,' ': 2,'w': 1,'r': 1,'d': 1,'!': 1}
```

(l'ordre des clefs n'ayant pas d'importance).

Écrire une fonction `occurence_lettres` avec prenant comme paramètre une variable `phrase` de type `str`. Cette fonction doit renvoyer un dictionnaire de type constitué des occurrences des caractères présents dans la phrase.

Exercice 6

Programmer la fonction `moyenne` prenant en paramètre un tableau d'entiers `tab` (type `list`) qui renvoie la moyenne de ses éléments si le tableau est non vide et affiche `'erreur'` si le tableau est vide.

Exemples :

```
>>> moyenne([5,3,8])
5.333333333333333
>>> moyenne([1,2,3,4,5,6,7,8,9,10])
5.5
>>> moyenne([])
'erreur'
```

Exercice 7

Écrire une fonction python appelée `nb_repetitions` qui prend en paramètres un élément `elt` et une liste `tab` et renvoie le nombre de fois où l'élément apparaît dans la liste.

Exemples :

```
>>> nb_repetitions(5,[2,5,3,5,6,9,5])
3
>>> nb_repetitions('A',['B','A','B','A','R'])
2
>>> nb_repetitions(12,[1,'!',7,21,36,44])
0
```

Exercice 8

On considère des mots à trous : ce sont des chaînes de caractères contenant uniquement des majuscules et des caractères `'*'`. Par exemple `'INFO*MA*IQUE'`, `'***I***E**'` et `'*S*'` sont des mots à trous.

Programmer une fonction `correspond` qui :

- prend en paramètres deux chaînes de caractères `mot` et `mot_a_trous` où `mot_a_trous` est un mot à trous comme indiqué ci-dessus,
- renvoie :
  - `True` si on peut obtenir `mot` en remplaçant convenablement les caractères `'*'` de `mot_a_trous`.
  - `False` sinon.

Exemples :

```
>>> correspond('INFORMATIQUE', 'INFO*MA*IQUE')
True

>>> correspond('AUTOMATIQUE', 'INFO*MA*IQUE')
False
```

Exercice 9

Pour cet exercice :

- On appelle « mot » une chaîne de caractères composée avec des caractères choisis parmi les 26 lettres minuscules ou majuscules de l'alphabet,
- On appelle « phrase » une chaîne de caractères :
  - composée avec un ou plusieurs « mots » séparés entre eux par un seul caractère espace `' '`,
  - se finissant :
    - soit par un point `'.'` qui est alors collé au dernier mot,
    - soit par un point d'exclamation `'!'` ou d'interrogation `'?'` qui est alors séparé du dernier mot par un seul caractère espace `' '`.

Voici quatre exemples de phrases :

- `'Le point d exclamation est separe !'`
- `'Il y a un seul espace entre les mots !'`
- `'Le point final est colle au dernier mot.'`
- `'Gilbouze macarbi acra cor ed filbuzine ?'`

Après avoir remarqué le lien entre le nombre de mots et le nombres de caractères espace dans une phrase, programmer une fonction `nombre_de_mots` qui prend en paramètre une phrase et renvoie le nombre de mots présents dans cette phrase.

Exemples :

```
>>> nombre_de_mots('Le point d exclamation est separe !')
6

>>> nombre_de_mots('Il y a un seul espace entre les mots !')
9
```

Exercice 10

Programmer la fonction `multiplication` prenant en paramètres deux nombres entiers `n1` et `n2`, et qui renvoie le produit de ces deux nombres.

Les seules opérations autorisées sont l'addition et la soustraction.

Exemples :

```
>>> multiplication(3,5)
15
>>> multiplication(-4,-8)
32
>>> multiplication(-2,6)
-12
>>> multiplication(-2,0)
0
```

Exercice 11

Programmer une fonction `renverse`, prenant en paramètre une chaîne de caractères non vide `mot` et renvoie une chaîne de caractères en inversant ceux de la chaîne `mot`.

Exemple :

```
>>> renverse("informatique")
"euqitamrofni"
```

Exercice 12

Sur le réseau social TipTop, on s'intéresse au nombre de « like » des abonnés. Les données sont stockées dans des dictionnaires où les clés sont les pseudos et les valeurs correspondantes sont les nombres de « like » comme ci-dessous :

```
{'Bob': 102, 'Ada': 201, 'Alice': 103, 'Tim': 50}
```

Écrire une fonction `max_dico` qui :

- Prend en paramètre un dictionnaire `dico` non vide dont les clés sont des chaînes de caractères et les valeurs associées sont des entiers ;
- Renvoie un tuple dont :
  - La première valeur est la clé du dictionnaire associée à la valeur maximale ;
  - La seconde valeur est la première valeur maximale présente dans le dictionnaire.

Exemples :

```
>>> max_dico({'Bob': 102, 'Ada': 201, 'Alice': 103, 'Tim': 50})
('Ada', 201)
>>> max_dico({'Alan': 222, 'Ada': 201, 'Eve': 220, 'Tim': 50})
('Alan', 222)
```

Exercice 13

On s'intéresse à la suite d'entiers définie par

$U_1 = 1$, $U_2 = 1$ et, pour tout entier naturel n, par $U_{n+2} = U_{n+1} + U_n$.

Elle s'appelle la suite de Fibonnaci.

Écrire la fonction `fibonacci` qui prend un entier `n > 0` et qui renvoie l'élément d'indice `n` de cette suite.

On utilisera une programmation dynamique (pas de récursivité).

Exemples :

```
>>> fibonacci(1)
1
>>> fibonacci(2)
1
>>> fibonacci(25)
75025
>>> fibonacci(45)
1134903170
```

Exercice 14

Écrire en python deux fonctions :

- `lancer` de paramètre `n`, un entier positif, qui renvoie un tableau de type `list` de `n` entiers obtenus aléatoirement entre 1 et 6 (1 et 6 inclus) ;
- `paire_6` de paramètre `tab`, un tableau de type `list` de `n` entiers entre 1 et 6 obtenus aléatoirement, qui renvoie un booléen égal à `True` si le nombre de 6 est supérieur ou égal à 2, `False` sinon.

On pourra utiliser la fonction `randint(a,b)` du module `random` pour laquelle la documentation officielle est la suivante :

```
Renvoie un entier aléatoire N tel que a <=N <= b.
```

Exemples :

```
>>> lancer1 = lancer(5)
[5, 6, 6, 2, 2]
>>> paire_6(lancer1)
True
>>> lancer2 = lancer(5)
[6, 5, 1, 6, 6]
>>> paire_6(lancer2)
True
>>> lancer3 = lancer(3)
[2, 2, 6]
>>> paire_6(lancer3)
False
>>> lancer4 = lancer(0)
[]
>>> paire_6(lancer4)
False
```