Programmation impérative

Exercice 1 : Plus grand commun diviseur
Le listing 1 donne la spécification d'une fonction appelée « pgcd » qui calcule le plus petit commun diviseur de deux entiers strictement positifs. Ce programme utilise la possibilité en Python 3 de préciser le type des paramètres et le type de retour d'une fonction (ces informations de type sont ignorées de l'interpréteur Python mais d'autres outils, PyCharm par exemple, peuvent les exploiter) et docstring pour documenter le code. Le commentaire inclut des exemples d'utilisation de la fonction avec les résultats attendus. Il s'agit de tests qui seront exécutés par doctest.

1.1 Lire le programme fourni au listing 1. L'exécuter. Constater que les tests écrits dans les commentaires sont effectivement réalisés... et échouent.

1.2 Écrire le code de la fonction « pgcd » (remplacer `pass` par les bonnes instructions). Le pgcd sera calculé suivant l'algorithme d'EUCLIDE :

$$pgcd(a,b) = \begin{cases} a & \text{si } a = b \\ pgcd(a-b,b) & \text{si } a > b \\ pgcd(a,b-a) & \text{si } a < b \end{cases}$$

Tous les tests doivent maintenant réussir. Peut-on en conclure que le programme est correct ?

1.3 Ajouter un nouveau test dans la documentation qui calcule le pgcd de 2^{25} et 1. Constater que le temps d'exécution est long (sinon remplacer 25 par 30). Proposer une nouvelle implantation plus efficace de la fonction pgcd.

Exercice 2 : Racine carrée
Déterminer la racine carrée d'un réel en utilisant la suite ci-après qui converge vers $\sqrt{a}$:

$$\begin{cases} u_{n+1} = (u_n + a/u_n)/2 \\ u_0 = a \end{cases}$$

Exercice 3 : Indices d'éléments dans une liste
Dans cet exercice, on s'intéresse à la recherche d'un élément dans une liste.

3.1 Écrire la spécification d'une fonction « indice » qui retourne la position d'un élément dans une liste d'entier. On donnera aussi des exemples d'utilisation. On considère que le premier élément de la liste est à la position 0.

3.2 Écrire l'implantation de cette fonction.

3.3 Ajouter deux paramètres optionnels sur la fonction « indice » pour préciser l'indice de début et l'indice de fin sur la liste. L'élément est cherché dans l'intervalle [debut..fin[. Par défaut, l'élément est cherché sur toute la liste. Le commentaire et les tests doivent être complétés avant de modifier le code.

Listing 1 – Programme Python pour calculer le Pgcd (fichier pgcd.py)

```
def pgcd(a: int, b: int) -> int:
    """Le pgcd de a et b, deux entiers strictements positifs.

    >>> pgcd(10, 15)
    5

    >>> pgcd(19, 60)
    1

    >>> pgcd(100, 100)
    100

    >>> pgcd(-10, 10)
    Traceback (most recent call last):
    ...
    ValueError

    >>> pgcd(25, -5)
    Traceback (most recent call last):
    ...
    ValueError

    >>> pgcd(25, 0)
    Traceback (most recent call last):
    ...
    ValueError

    """
    pass


if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

3.4 *(optionnelle)* Ajouter un nouveau paramètre optionnel qui indique le numéro de l'occurrence cherchée (par défaut, c'est l'indice de la première occurrence qui sera retourné).

Exercice 4 : Jeu du devin

Le jeu du devin se joue à deux joueurs. Le premier joueur choisit un nombre compris entre 1 et 999. Le second doit le trouver en un minimum d'essais. À chaque proposition, le premier joueur indique si le nombre proposé est plus grand ou plus petit que le nombre à trouver. En fin de partie, le nombre d'essais est donné.

Indication : On suppose qu'il existe une fonction qui permet d'obtenir un nombre aléatoire compris entre 1 et 999.

4.1 *La machine fait deviner le nombre.* Écrire un programme dans lequel la machine choisit un nombre et le fait deviner à l'utilisateur.

4.2 *La machine joue.* Écrire un programme dans lequel l'utilisateur choisit un nombre et la machine doit le trouver. Pour chaque nombre proposé, l'utilisateur indique s'il est trop petit (`'p'` ou `'P'`), trop grand (`'g'` ou `'G'`) ou trouvé (`'t'`, `'T'`).

Indication : On utilisera une recherche par dichotomie pour trouver le nombre.

4.3 *Le programme complet.* Écrire le programme de jeu qui donne le choix à l'utilisateur entre deviner ou faire deviner le nombre.

4.4 *On peut recommencer.* Compléter le programme précédent pour que l'ordinateur propose de faire une nouvelle partie lorsque la précédente est terminée.

Exercice 5 : Tri par insertion séquentielle

Soit $A[1..N]$ un vecteur de N entiers relatifs quelconques, l'objectif est de trier le vecteur A. Le vecteur A est trié si $A[1] \leq A[2] \leq \cdots \leq A[N]$.

Le tri utilisé est le tri par insertion séquentielle. C'est un tri en $(N-1)$ étapes. L'étape i consiste à placer le $(i+1)^{\text{e}}$ élément du vecteur à sa place dans le sous-vecteur $A(1..i+1)$ sachant que $A(1..i)$ a été trié par les étapes précédentes. Dans le cas du tri par insertion séquentielle, la recherche de la position d'insertion, se fait séquentiellement en comparant successivement l'élément à insérer aux i premiers éléments du vecteur.

Exemple : Voici les différentes valeurs du vecteur 8 2 9 5 1 7 après chaque étape (la partie encadrée correspond à la partie du vecteur déjà traitée et donc triée) :

vecteur initial	:	[8] 2 9 5 1 7
après l'étape 1	:	[2 8] 9 5 1 7
après l'étape 2	:	[2 8 9] 5 1 7
après l'étape 3	:	[2 5 8 9] 1 7
après l'étape 4	:	[1 2 5 8 9] 7
après l'étape 5	:	[1 2 5 7 8 9]

5.1 Écrire un sous-programme pour trier dans l'ordre croissant les éléments d'un vecteur. On s'utilisera le principe du tri par insertion séquentielle.

5.2 En conservant le principe du tri par insertion, expliquer comment améliorer l'efficacité de cet algorithme.

Exercice 6 : Nombres amis
Deux nombres N et M sont amis si la somme des diviseurs de M (en excluant M lui-même) est égale à N et la somme des diviseurs de N (en excluant N lui-même) est égale à M.

Écrire un programme qui affiche tous les couples (N, M) de nombres amis tels que $0 < N < M \leq MAX$, MAX étant lu au clavier.
Indication : Les nombres amis compris entre 1 et 100000 sont (220, 284), (1184, 1210), (2620, 2924), (5020, 5564), (6232, 6368), (10744, 10856), (12285, 14595), (17296, 18416), (66928, 66992), (67095, 71145), (63020, 76084), (69615, 87633) et (79750, 88730).

Exercice 7 : Nombres de Amstrong
Les *nombres de Amstrong* appelés parfois *nombres cubes* sont des nombres entiers qui ont la particularité d'être égaux à la somme des cubes de leurs chiffres. Par exemple, 153 est un nombre de Amstrong car on a :

$$153 = 1^3 + 5^3 + 3^3.$$

Afficher tous les nombres de Amstrong sachant qu'ils sont tous compris entre 100 et 499.
Indication : Les nombres de Amstrong sont : 153, 370, 371 et 407.

Exercice 8 : Vérification expérimentale de la loi de Benford
Prenons un ensemble de données numériques quelconques et classons-les en fonction de leur premier chiffre significatif. Qu'obtenons-nous ? Intuitivement, nous nous attendons à trouver 9 classes [1] contenant une quantité équivalente d'échantillons, c'est-à-dire environ 11 % de nombres commençant par 1, puis 11 % par le chiffre 2, etc.

Pourtant, on s'aperçoit généralement qu'une telle distribution intuitive n'est pas du tout respectée ! Le chiffre 1 est bien plus représenté, le chiffre 2 un peu moins... le chiffre 9 n'apparaissant que dans guère plus de 4 % des cas.

C'est en constatant l'usure excessive du premier volume des tables de logarithmes que Newcomb découvrit en 1881 une telle irrégularité et proposa une loi de distribution (reprise plus tard par Benford). Cette loi stipule que le chiffre $i \neq 0$ apparaît avec une probabilité de l'ordre de $log_{10}(1 + 1/i)$.

Cette loi n'a toujours pas pu être démontrée ni son cadre d'application clairement défini. Elle trouve cependant quelques applications (détection de fraudes par exemple). On trouvera plus d'informations dans « La Recherche », hors série numéro 2, août 1999.

L'objectif de cet exercice est de vérifier, sur quelques exemples, la loi de Benford.

8.1 Écrire un programme permettant de mesurer la fréquence du premier chiffre significatif des valeurs d'un échantillon (d'entiers). Les résultats seront affichés sous forme de pourcentage. On considère que les valeurs de l'échantillon seront saisies au clavier et se termine par l'entier 0.

8.2 Appliquer le programme sur un ensemble de mesures, par exemple sur la taille des fichiers contenus dans vos répertoires.

1. 9 et non 10 puisque le chiffre 0, par définition, ne peut pas être le premier chiffre significatif d'un nombre.