Cours Python

Série d'exercices

1 Type de données et opérations

Exercice 1.1

Qu'affichent les calculs suivants ? (Attention au type de retour)

a) 2 ** 6
b) 4 ** 0.5
c) 4 ** -2
d) 2 ** -1
e) 1 + 2 * 3
f) 4 ** 0 + 1
g) 25 ** 0 * 2.0
h) 27 ** 1 / 3
i) 27 ** (1/3)
j) 4 - -4
k) 7 / -2.0
l) 1 + 2 + 3 + 4.0
m) 2 ** 2 ** 3
n) 10 % 3 / 2
o) 27 ** (1/3)
p) 20 % 7 // 2
q) 10 / 2 / 2
r) 10 // 3 % 2

Exercice 1.2

Quelle est l'opération permettant de calculer une racine carrée ?

Exercice 1.3

Quel est le résultat de ces expressions ? Précisez s'il y a une erreur de syntaxe.

a) "bonjour, " * 3
b) "coucou" + 2
c) "salut" + " poilu !"

Exercice 1.4

Donnez le résultat des expressions à l'aide des valeurs suivantes :

$a = 2$
$b = 5$
$c = True$

a) a < b
b) (a + 3) < b
c) a < 4 and a < 1
d) a < 4 or a < 1
e) a < 3 and a > 1
f) a > 1 or a < -1
g) a >= 1 or a <= -1
h) a > 2 or b >= 5
i) c **or** (a > 10)
j) c **and not** c
k) c **or not** c
l) c == (a < b)
m) c != (a > 0)

Exercice 1.5

Le code ci-dessous retourne une erreur de syntaxe. Expliquez pourquoi et trouvez une solution sans modifier la première ligne.

```
>>> age = 33
>>> "J'ai " + age + " ans"
```

Exercice 1.6

Que contiennent les variables à la fin de l'exécution du programme suivant ?

1.

```
x = 10
x = 2 * x
x = x - 2
x = x / 2
```

2.

```
x = y = 5
x = x * 2
y = y * 3
z = x + y
```

Exercice 1.7

Que contiennent les variables à la fin de l'exécution du programme suivant ?

1.

```
a, b = 1, 1
a, b = b, a + b
a, b = b, a + b
a, b = b, a + b
```

2.

```
a = 10
b = 3
q, r = a // b, a % b
a, b, q, r = r, q, b, a
```

Exercice 1.8

Soit le code ci-dessous :

```
x = 10
y = 20
x = x + y
y = x - y
x = x - y
```

1) Quelles sont les valeurs de x et y à la fin du script ?
2) Que réalise ce script ?
3) Réalisez le même comportement à l'aide d'une variable temporaire.
4) Réalisez le même comportement à l'aide d'une affectation multiple.

Exercice 1.9

A l'aide d'une variable positive ou nulle n, réalisez les opérations suivantes :

a) extraire le chiffre des centaines du nombre n. Par exemple, retourne 3 si n vaut 128345 et 0 si n vaut 24.

b) si n est un nombre de secondes, attribuez aux variables *heure*, *minute* et *seconde* l'heure correspondante. Aidez-vous de l'affectation multiple. Par exemple si $n = 12670$ alors $heure = 3$, $minute = 31$ et $seconde = 10$

2 IHM et fonctions

Exercice 2.1

Réalisez une application permettant de calculer un intérêt composé annuel. L'utilisateur doit être capable d'entrer les informations suivantes :

— Le capital de départ
— Le taux d'intérêt
— Le nombre d'années

Le capital final après n années se calcul ainsi :

$$C_f = C_i \cdot (1 + \rho)^{\alpha}$$

C_f correspond au capital final, C_i est le capital initial, ρ est l'intérêt (pour un intérêt de 4% ρ = 0.04) et α représente le nombre d'années.

Question subsidiaire : Un de vos lointains ancêtres dont vous êtes le seul héritié a accordé un crédit d'une valeur de 5 ct à Baltazar le roi mage l'année de la naissance d'un certain Jésus il y a 2017 ans. L'intérêt relativement bas de 1% a toutefois permis de faire fructifier le capital initial. Combien pouvez-vous réclamer à la famille du débiteur ? Cela suffit-il pour payer une tournée générale à toute la classe ?

Exercice 2.2

Ecrivez les fonctions suivantes :

— $f(x) = 3 * x + 2$, de signature : (num) $\rightarrow$ (num)

— $g(x) = x^2$, de signature : (num) $\rightarrow$ (num)

— $h(x, y) = x + 2 \cdot y$, de signature : (num, num) $\rightarrow$ (num)

— $i(x) =$ "x", de signature : (num) $\rightarrow$ (str)

— $j(x) = f(g(x)) = (f \circ g)(x)$, de signature : (num) $\rightarrow$ (num)

— $k(x, y) = i(h(x, y)) = (i \circ h)(x, y)$, de signature : (num, num) $\rightarrow$ (str)

Exercice 2.3

a) Ecrivez une fonction `strictement_positif(a)` indiquant si un entier est strictement positif. Opérateurs autorisés : $<$, $>$, $>=$, $<=$.

b) Ecrivez une fonction `maximum(a, b)` prenant en paramètre deux entiers et retournant la valeur du plus grand des deux. Opérateurs autorisés : $==$, $<$, $>$, $<=$, $>=$.

c) A l'aide de la fonction précédente, écrivez une fonction `minimum(a, b )` permettant de retourner la valeur minimum de deux entiers. Opérateurs autorisés : $==$, $!=$.

d) A l'aide de la fonction précédente, écrivez une fonction `negatif(a))` indiquant si un entier est négatif ou nul. Opérateurs autorisés : $==$, $!=$.

e) Ecrivez une fonction `pair(a)` indiquant si un entier est pair. Opérateurs autorisés : $\%$, $==$, $!=$.

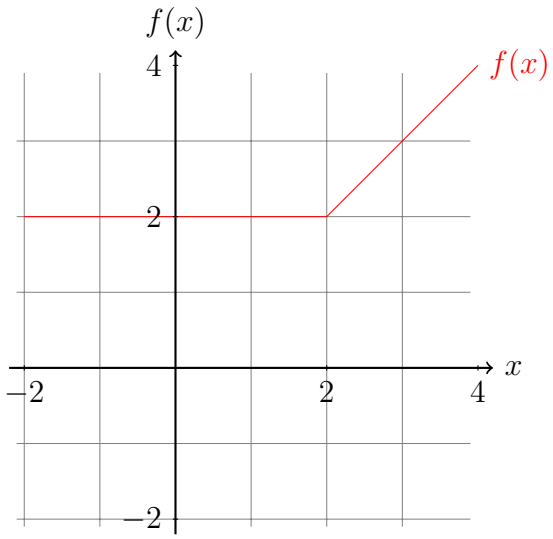
f) Proposez une fonction permettant de retourner l'opposé d'un nombre. (Exemple : l'opposé de 5 est -5, celui de -3 est 3)

g) A l'aide des fonctions `negatif` et `oppose` , écrivez une fonction qui retourne la valeur absolue d'un nombre

Exercice 2.4

Réalisez l'algorithme représentant la fonction ci-dessous sans utiliser de branchement conditionnel :

$$f(x) = \begin{cases} x & \text{si } x \geq 2 \\ 2 & sinon \end{cases}$$



3 Les boucles

Exercice 3.1

A l'aide d'un paramètre n, écrivez une fonction `afficher_sapin(n)` qui affiche la suite de symboles sous cette forme (exemple avec n = 5) :

```
    *
   ***
  *****
 *******
*********
```

Exercice 3.2

Réalisez une fonction permettant de calculer une approximation de π :

$$\sum_{n=1}^{N} \frac{1}{n^4} = \frac{\pi^4}{90}$$

Cette fonction prend N en paramètre et retourne π.

4 Structures de données

Exercice 4.1

Ecrivez une fonction `demander_valeur_entiere()` qui demande à l'utilisateur une valeur entière et la retourne. Cette fonction redemande à l'utilisateur d'entrer une valeur tant que celle-ci n'est pas une valeur entière. Il est possible de passer en argument de la fonction le texte à afficher à l'utilisateur. Aidez-vous de la méthode isnumeric() qui s'applique sur les chaînes de caractères. Exemples :

— "2".isnumeric() retourne `True`

— "2sd".isnumeric() retourne `False`

— "-2".isnumeric() retourne `False`

Vous remarquerez que la méthode isnumeric() ne fonctionne qu'avec des valeurs positives ou nulles. Trouvez une solution pour que votre fonction puisse s'appliquer avec des valeurs négatives. Exemple de fonctionnement :

```
Entrez une valeur entière : asdf
Valeur erronée
Entrez une valeur entière : 3.8
Valeur erronée
Entrez une valeur entière : -4
La valeur retournée est -4
```

Exercice 4.2

A l'aide de la fonction précédente, écrivez deux fonctions :

— une première qui demande une valeur entière suppérieur ou égale à une borne,

— une seconde qui demande une valeur entière entre une borne min et max.

Exercice 4.3

La suite de Fibonacci est une suite d'entiers dans laquelle chaque terme est la somme des deux termes qui le précèdent. L'algorithme commence avec les nombre 0 et 1. Après 10 itérations on optient la suite suivante : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55.
Mettez en oeuvre l'algorithme ci-dessous à l'aide du langage Python. **Favorisez l'affectation multiple**.

5 Slicing

Exercice 5.1

Réalisez un programme qui demande une chaîne de caractères à l'utilisateur et affiche :

a) *True* si les lettres de la chaîne sont en majuscules

b) Le nombre d'occurences de la première lettre dans la chaîne

c) Le nombre d'occurences de la dernière lettre dans la chaîne

d) *True* si la première et dernière lettre sont identiques

Exercice 5.2

Un palindrome est un mot ou une phrase qui signifie la même chose si on le lit de gauche à droite ou de droite à gauche. Quelques exemples de palindromes : kayak, mon nom, à, été, selles, sexes, tôt, ...

Réalisez un programme retournant *True* si le mot ou la phrase entrée par l'utilisateur est un palindrome, *False* sinon.

6 Structures de données avancées

Exercice 6.1 Fonctions sur les listes

Réalisez un script comprenant un ensemble de fonctionnalités.

Exercice 6.1.1 longueur

Cette fonction prend une liste en paramètre et retourne sa longueur. Par exemple, `longueur([1,4,4])` retourne 3. N'utilisez pas la fonction `len` de Python qui réalise la même fonctionnalité.

Exercice 6.1.2 moyenne

Cette fonction prend une liste en paramètre et retourne sa moyenne. Par exemple, `moyenne([4,5,3])` retourne 4.0.

Exercice 6.1.3 plus_grand_ou_egal

Cette fonction prend une liste en paramètre et un seuil et retourne une nouvelle liste avec les éléments plus grands ou égaux au seuil. Par exemple, `plus_grand_ou_egal([4,5,12, 9, 3], 7)` retourne [12, 9].

Exercice 6.1.4 plus_petit_ou_egal

Cette fonction prend une liste en paramètre et un seuil et retourne une nouvelle liste avec les éléments plus petits ou égaux au seuil. Par exemple, `plus_petit_ou_egal([4,5,12, 9, 3], 9)` retourne [4, 5, 9, 3].

Exercice 6.1.5 positive

Cette fonction prend une liste en paramètre et retourne une nouvelle liste avec les éléments positifs ou nuls. Par exemple, `positive([4, -2, 3, 1])` retourne [4, 3, 1].

Exercice 6.1.6 negative

Cette fonction prend une liste en paramètre et retourne une nouvelle liste avec les éléments négatifs ou nuls. Par exemple, `negative([4, -2, 3, 1])` retourne [-2].

Exercice 6.1.7 dans

Cette fonction prend une liste et un nombre et vérifie si ce nombre se trouve dans la liste. N'utilisez pas l'opérateur `in` . Par exemple, `dans([1,2,3], 3)` retourne True.

Exercice 6.1.8 minimum

Cette fonction prend une liste et retourne l'élément le plus petit de la liste. Par exemple, `minimum([3,2,1,6])` retourne 1. N'utilisez pas la fonction `min` de Python qui réalise la même fonctionnalité.

Exercice 6.1.9 maximum

Cette fonction prend une liste et retourne l'élément le plus grand de la liste. Par exemple, `minimum([3,2,1,6])` retourne 6. N'utilisez pas la fonction `max` de Python qui réalise la même fonctionnalité.

Exercice 6.1.10 repeter

Cette procédure affiche n fois une chaîne de caractères entrée en paramètre. N'utilisez pas l'opérateur `*` de Python.

Exercice 6.1.11 binaire

Cette fonction prend un nombre binaire, représenté par une chaîne de caractères de '0' et de '1', et retourne la valeur décimale. On suppose que la chaîne de caractères est valide. Par exemple, `binaire("0110")` retourne 6.

Exercice 6.1.12 and_lst

Cette fonction prend une liste de valeur booléenne en paramètre et retourne sa réduction à l'aide de l'opération ET logique. Par exemple, `and_lst([True, True, False])` retourne False car l'expression `True and True and False` retourne False

Exercice 6.1.13 or_lst

Cette fonction prend une liste de valeur booléenne en paramètre et retourne sa réduction à l'aide de l'opération OU logique. Par exemple, `or_lst([True, True, False])` retourne True car l'expression `True or True or False` retourne False

Exercice 6.1.14 table_mult

Cette procédure affiche une table de multiplication. Elle prend en paramètre la valeur maximum des multiples. Par exemple, l'appel de la procédure `table_mult(4)` affichera le résultat ci-dessous. Le multiple minimum est 2.

```
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
```

Exercice 6.1.15 zip

Cette fonction 'fermeture éclaire' prend deux listes et retourne une liste de tuples ou chaque tuple est une composition d'élément des deux listes. Si les deux listes ne sont pas de même taille alors il faut retourner une liste vide. Par exemple, `zip([1,2,3], ["a", "b", "c"])` retourne [(1, "a"), (2, "b"), (3, "c")].

Exercice 6.1.16 unzip

Cette fonction réalise l'opération inverse. Elle prend une liste de tuple et retourne deux listes. Par exemple, `unzip([(1, "a"), (2, "b"), (3, "c"))` retourne [1,2,3], ["a", "b", "c"].

Exercice 6.1.17 transform

Cette fonction prend une liste et une fonction f en paramètre et retourne une nouvelle liste où la fonction f est appliquée sur chaque élément. Par exemple :

```
def f(x):
  return 2*x+2

transform([1,2,3], f) # retourne [4, 6, 8]
```

Exercice 6.1.18 filter

Cette fonction prend une liste et un prédicat et retourne une nouvelle liste où chaque élément vérifie ce prédicat. Par exemple :

```
def majeur(age):
  return age >= 18

filter([12,21,13, 45], majeur) # retourne [21, 45]
```

Exercice 6.2

Ecrivez une fonction `remove_duplicates(lst)` qui prend en paramètre une liste et qui retourne une nouvelle liste ne contenant aucun élément dupliqué. L'ordre ne doit pas forcément être respecté.

Entrée : une liste contenant des entiers, des caractères ou des chaînes de caractères.
Sortie : une liste dont chaque élément n'apparaît qu'une fois

Exemple : `remove_duplicates([1,2,2,'a','a'])` retournera une liste similaire à `[1,2,'a']`

Exercice 6.3

Trois personnes (Mathilde, Jean-Claude et Gustave) ont choisi des activités. Chaque activité possède un numéro unique allant de 1 à n.

- Quelle est l'opération permettant de connaître les activités communes à ces trois personnes ?
- Quelle est l'opération permettant de connaître si une activité lambda a été choisie par au moins une personne ?

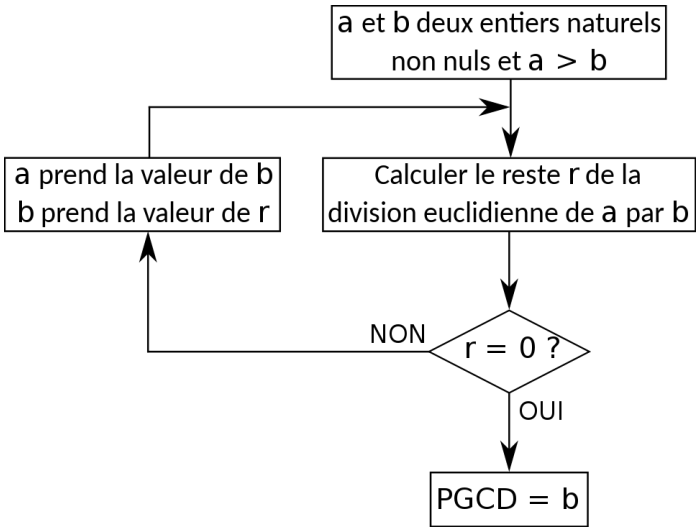
Aide : Les choix de chaque personne peuvent être représentés par un ensemble. Mathilde = 1,3,4,5 signifique que Mathilde a choisi les activité 1, 3, 4 et 5.

Exercice 6.4 Euclide

Algorithme d'Euclide
Le but de cet algorithme est de trouver le plus grand diviseur commun de deux entiers.

Description de wikipedia :
Soient deux entiers naturels [non nuls] a et b [$a > b$], dont on cherche le PGCD. [...]. Une suite d'entiers $(a_n)_n$ est définie par récurrence de pas 2, plus précisément par divisions euclidiennes successives ; la suite est initialisée par $a_0 = a, a_1 = b$, puis propagée par la règle de récurrence : tant que a_{n+1} est non nul, a_{n+2} est défini comme le reste de la division euclidienne de a_n par a_{n+1}. On commence donc par calculer le reste de la division de a par b, qu'on note r ; puis on remplace a par b, puis b par r, et on réapplique le procédé depuis le début.
On obtient ainsi une suite, qui vaut 0 à un certain rang ; le PGCD cherché est le terme précédent de la suite.

Diagramme de l'algorithme :
Aidez-vous du diagramme ci-dessous pour mettre en oeuvre l'algorithme d'Euclide à l'aide de Python.



Exercice 6.5 Fractions

A l'aide de l'aglorithme d'Euclide, réalisez un module permettant de représenter et de manipuler des fractions.
Une fraction peut-être représenté par un tuple dont le premier élément est le numérateur et le second élément et le dénominateur.
Les fonctions que vous devez implémenter sont les suivantes :

```
# prend en paramètre une fraction et retourne
# sa version irréductible
# signature: Tuple -> (int, int)
def reduce(fraction):
  pass

# Créer une nouvelle fraction irréductible
# ex: fraction(10, 6) retourne (5,3)
# signature: (int, int) -> (int, int)
def fraction(a,b):
  pass

# additionne deux fractions
def add(fr1, fr2):
  pass
```

```python
# soustraction de deux fractions
def sub(fr1, fr2):
  pass

# multiplication de deux fractions
def mul(fr1, fr2):
  pass

# division de deux fractions
def div(fr1, fr2):
  pass
```

Exercice 6.6 OpenData : modèle numérique de surface

Réalisez une application permettant de créer une image à partir de données altimétriques. Le format du fichier importé est un format ESRI ASCII Raster (extension *.asc). Ce format représente un modèle numérique de terrain (altitude) ou de surface (altitude avec bâtiments, végétations, ...). Le fichier exporté est une image au format pgm ASCII (extension *.pgm) en niveau de gris.
Les formats asc et pgm respectent une spécification permettant leur lecture. Il s'agit de fichier ASCII contenant un en-tête suivi des données sous la forme d'une matrice. Le format pgm a été présenté au cours. Ci-dessous, un extrait d'un fichier asc :

```
ncols 2497
nrows 1743
xllcorner 2499526.5
yllcorner 1116878
cellsize 0.5
nodata_value 0
396.82 398.8 399.05 399.06 399.06 397.49 396.76 396.69 396.71 396.69 396.67
373.48 373.47 373.48 373.5 373.45 373.43 373.39 373.35 373.35 373.38 373.43
373.42 373.42 373.43 373.44 373.46 373.47 373.49 373.5 373.52 373.53 373.43
...
```

L'en-tête est composé des informations suivantes :

- **ncols** représente le nombre de colonnes de la matrice des altitudes
- **nrows** représente le nombre de lignes
- **xllcorner** et yllcorner les coordonnées x et y de la première mesure (haut gauche)
- **cellsize** l'unité de distance séparant chaque mesure
- **nodata_value** la valeur d'altitude par défaut pour une cellule hors périmètre.

Puis, chaque valeur représente une altitude de gauche à droite et de haut en bas.

Dans le fichier exporté, chaque altitude devra correspondre à un niveau de gris allant de 0 à $N-1$ si N est le nombre de nuances de gris. Vous devez donc normaliser les altitudes pour les faire correspondre à une nuance. Faites attentions aux types, les altitudes sont des nombres décimaux et les nuances sont des entiers.

L'idée principale de votre algorithme et la suivante :

- Parcourez le fichier asc pour trouver l'altitude minimale et l'altitude maximale

— La différence entre l'altitude minimale et maximale peut correspondre aux nombres de nuances mais vous pouvez définir une autre valeur

— A l'aide de l'altitude minimale, de l'altitude maximale et du nombre de nuance, trouvez la fonction mathématique permettant de transformer une altitude en une nuance. (L'altitude minimale correspond à la valeur 0 (couleur noire) et l'altitude maximale à la valeur N (couleur blanche))

— Ecrivez l'en-tête du fichier pgm

— Parcourez à nouveau le fichier asc. Pour chaque altitude, transformez celle-ci en une nuance et enregistrez-là dans le fichier pgm.

Exercice 6.7 Rest & JSON : horaires des transports publics suisses

Réalisez un programme permettant d'afficher les prochains départs de transports publics à un arrêt donné.
Il existe une application libre en ligne qui permet de récupérer des informations sur les départs et les connections de tous les transports publics suisses.
Récupérez le script transport.py et complétez le code pour afficher les départs de bus pour l'arrêt de la HEIG-VD (ou tout autre arrêt). Indiquez le numéro du type de transport, l'heure de départ et sa direction.
Aidez vous également de la docutmentation de l'api libre des transports publics suisses :
`http://transport.opendata.ch/docs.html#stationboard`

Voici un exemple d'affichage :

```
Horaires pour Yverdon-les-Bains, HEIG-VD:
NFB 601 05:35:00        Yverdon-les-Bains, gare
NFB 603 05:37:00        Yverdon-les-Bains, Blancherie
NFB 601 05:42:00        Yverdon-les-Bains, gare
NFB 601 05:57:00        Yverdon-les-Bains, gare
NFB 603 05:57:00        Yverdon-les-Bains, Blancherie
NFB 601 06:05:00        Yverdon-les-Bains, gare
NFB 601 06:17:00        Yverdon-les-Bains, gare
NFB 603 06:17:00        Yverdon-les-Bains, Blancherie
NFB 601 06:25:00        Yverdon-les-Bains, gare
NFB 601 06:37:00        Yverdon-les-Bains, gare
NFB 603 06:37:00        Yverdon-les-Bains, Blancherie
```

Exercice 6.8 Tri de dictionnaire

Considérez le dictionnaire suivant ayant pour clé un identifiant unique et comme valeur une liste ['Nom', 'Prénom', Age]. Réalisez une fonction qui permet d'afficher ce dictionnaire trié en fonction de l'age.

```
personnes = {"e3f6": ['Carson', 'Bill', 32],
             "2b75": ['Roman', 'Frank', 18],
             "5d05": ['Lespinas', 'Mike', 42],
             "c36d": ['Labrosse', 'Germain', 23],
             "20fe": ['Bousquet', 'Ivan', 55],
             "cc5a": ['Girard', 'Alfred', 68],
             "ed64": ['Dumon', 'Nicolas', 40]}
```

Exercice 6.9 Commande "tree"

Réalisez la fonction tree qui affiche le contenu d'une arborescence. Utilisez la librairie 'os' pour parcourir les dossiers.

```
>>> tree('/tmp/cours_python3-public')
+ examples
| - template.py
+ exercices
| - exercices.pdf
- README.md
- Python-partie1.pdf
```

Exercice 6.10 Détection de doublons dans un système de fichier

Trouvez une solution permettant de détecter et d'afficher le path des fichiers identiques dans une arborescence. A titre d'exemple, dans le dossier ci-dessous tous les fichiers commençant par la même lettre ont le même contenu.

```
- a.txt
- b.txt
- b2.txt
- c.txt
- d.txt
- d1.txt
- d2.txt
```

Le résultat de la détection retourne alors :

```
2 duplicates: ['sample_tree/b2.txt', 'sample_tree/b.txt']
3 duplicates: ['sample_tree/d1.txt', 'sample_tree/d.txt', 'sample_tree/d2.txt']
```

On peut calculer le md5 d'un fichier avec la commande suivante :

```
md5sum = md5(open(file_path, 'rb').read()).hexdigest()
```

Pensez aux dictionnaires !

Exercice 6.11 Premier parser yaml

Clonez l'arborescence 'docker-yml' pour réaliser votre propre parseur. Vous devez créer une première fonctionnalité qui affiche pour chaque fichier de configuration, l'image docker utilisée. Puis, réalisez une seconde fonctionnalité qui permet d'afficher pour chaque fichier, quels sont les ports qui sont mappés entre le conteneur docker et la machine hôte.
Trouvez des algorithmes sans expression régulière. La librairie 'os' et sa fonction 'walk' peut vous être utile.

Exercice 6.12 Dictionnaire et exceptions

Un dictionnaire en Python peut être vu comme une liste de tuple.
Réalisez un module qui met à disposition un ensemble de fonctionnalités permettant la gestion d'un dictionnaire. Utilisez un style purement impératif.

```
>>> dictionnaire = []
>>> add(dictionnaire, "nom", "Cavat")
>>> add(dictionnaire, "age", 26) # dictionnaire = [("nom", "Cavat"), ("age", 26)]
>>> val = drop(dictionnaire, "age") #val = 26, dictionnaire = [("nom", "Cavat")]
```

Réalisez les fonctions suivantes :

- index(dic, v) : retourne la première clé trouvée associée à la valeur v.
- is_in(dic, k) : vérifie si la clé se trouve dans le dictionnaire
- add(dic, k, v) : ajoute un élément dans le dictionnaire. Remplace la valeur v si k existe déjà
- get(dic, k) : retourne la valeur correspondant à une clé
- get_or_else(dic, k, v) : retourne v si la clé k n'existe pas
- length(dic) : retourne la longeur du dictionnaire
- drop(dic, k) : supprime le couple clé/valeur du dictionnaire et retourne la valeur associée
- values(dic) : retourne toutes les valeurs du dictionnaire
- key(dic) : retourne toutes les clés du dictionnaire
- reverse(dic) : retourne un dictionnaire ou les clés et les valeurs sont inversées

Créez vos propres exceptions pour gérer les différents problèmes que l'on pourrait rencontrer :

```
# par exemple:
class KeyException(Exception): pass
```

7 Compréhension de liste

Exercice 7.1 Fonctions

A l'aide de compréhension de liste, écrivez les fonctions ci-dessous :

- positive(xs) : retourne tous les éléments positifs dans la liste xs
- simplify(xs) : retourne le premier élément de chaque tuple dans la liste xs (exemple : simplify([(1,2), (3,4)]) retourne [1,3]
- negate(xs) : retourne l'opposé de chaque nombre de la liste xs si l'élément est pair
- dico(xs) : retourne un dictionnaire à l'aide d'une liste de tuple entré en paramètre (exemple : dico([("id",2), ("age",4)]) retourne ["id" : 2, "age" : 4]

8 Typage statique

Exercice 8.1 Contrôle des types avec mypy

Reprenez les exercices 2.1, 2.2 et 2.3 en ajoutant les types. Testez vos fonctions et vérifiez votre script à l'aide de mypy.

9 Programmation orientée objet

Exercice 9.1 Classe dictionnaire

Transformez votre dictionnaire réalisé en 6.12 pour le transformer en classe. Il devra être instanciable de la façon suivante :

```
mon_dict = Dictionnaire()
mon_dict.add(k, v)
mon_dict.values()
...
```

Ajoutez ensuite l'affichage du dictionnaire afin de pouvoir faire

```
print(mon_dict)
```

Surcharger les méthodes __getitem()__ et __setitem()__ pour pouvoir faire lecture et écriture avec les crochets :

```
mon_dict = Dictionnaire()
une_valeur = mon_dict["toto"]
mon_dict["toto"] = 12
```

Exercice 9.2 Contrôle de type

Adaptez votre exercice 6.5 pour faire une classe Fraction. Surchargez l'opérateur + avec la méthode __add__() :

```
a = Fraction(1, 2)
b = Fraction(3, 2)
c = a + b
```

Vérifiez les types avec la méthode isinstance(). L'opération doit accepter les int, les floats et les Fractions.

```
a = Fraction()
b = a + 12
```

10 Intérateurs et générateurs

Exercice 10.1 Parcours de dictionnaire

Modifiez votre classe Dictionnaire pour le rendre itérable.

```
mon_dict = Dictionnaire()
for k, v in mon_dict:
    print(k, v)
```

Exercice 10.2 Fibonacci

Créez un générateur de la suite de Fibonacci (cf exercice 4.3).

Exercice 10.3 Générateur et fichiers

Créez un générateur qui parcours un fichier de configuration de switch et retourne uniquement les lignes commençant par 'vlan members'.

11 Multi-threading

Exercice 11.1 module threading

Reprenez l'exercice 6.7 et calculez le temps nécessaire à l'affichage des 10 prochains départs pour les 10 stations ci-dessous :

```
stations = ["Yverdon-les-bains", "Geneve", "Lucerne", "Vevey", "Brig",
            "Coppet", "Romont", "Bern", "Zurich", "Winterthur" ]
```

Adaptez ensuite votre code pour réaliser ce traitement en façon concurrente avec le module threading.

Exercice 11.2 Queue

Implémentez une 2ème solution à l'aide de Queue, pouvant avoir un nombre différents de worker et de tâches.
Bonus : Adaptez la solution pour que l'affichage respecte l'ordre de la liste de stations.