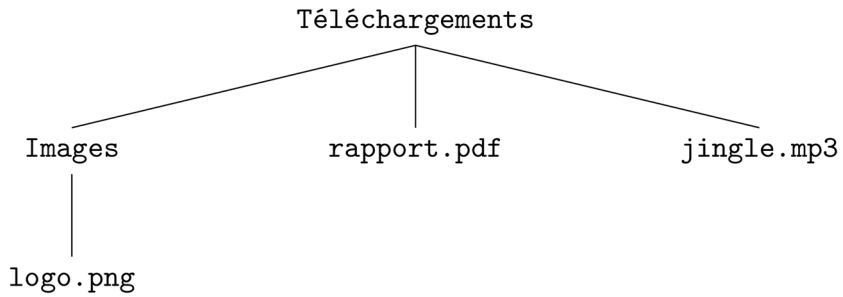
EXERCICE 3 (4 points)

Cet exercice porte sur les structures de données (dictionnaires)

Afin d'organiser les dossiers et les fichiers sur un disque dur, une structure arborescente est utilisée. Les fichiers sont dans des dossiers qui sont eux-mêmes dans d'autres dossiers, etc.
Dans une arborescence, chaque dossier peut contenir des fichiers et des dossiers, qui sont identifiés par leur nom. Le contenu d'un dossier est modélisé par la structure de données **dictionnaire**. Les clés de ce dictionnaire sont des chaînes de caractères donnant le nom des fichiers et des dossiers contenus.

Illustration par un exemple :
Le dossier appelé `Téléchargements` contient deux fichiers `rapport.pdf` et `jingle.mp3`, et un dossier Images contenant simplement le fichier `logo.png`.
Il est représenté ci-dessous.



Ce dossier `Téléchargements` est modélisé en Python par le dictionnaire suivant :
`{"Images": {"logo.png": 36}, "rapport.pdf": 450, "jingle.mp3": 4800}`
Les valeurs numériques sont exprimées en ko (kilo-octets).
`"logo.png": 36` signifie que le fichier `logo.png` occupe un espace mémoire de 36ko sur le disque dur.

On rappelle, ci-dessous, quelques commandes sur l'utilisation d'un dictionnaire :

- `dico = {}` crée un dictionnaire vide appelé `dico`,
- `dico[cle] = contenu` met la valeur `contenu` pour la clé `cle` dans le dictionnaire `dico`,
- `dico[cle]` renvoie la valeur associée à la clé `cle` dans le dictionnaire `dico`,
- `cle in dico` renvoie un booléen indiquant si la clé `cle` est présente dans le dictionnaire `dico`,
- `del dico[cle]` supprime la clé `cle` et sa valeur associée du `dico`.
- `dico.keys()` renvoie la liste des clés du dictionnaire `dico`

L'**adresse** d'un fichier ou d'un dossier correspond au nom de tous les dossiers à parcourir depuis la racine afin d'accéder au fichier ou au dossier. Cette adresse est modélisée en Python par la liste des noms de dossier à parcourir pour y accéder.

Exemple : L'adresse du dossier : `/home/pierre/Documents/` est modélisée par la liste `["home", "pierre", "Documents"]`.

1. Dessiner l'arbre donné par le dictionnaire suivant, qui correspond au dossier `Documents`.

```
Documents = {
  "Administratif":{
    "certificat JDC.pdf ": 1500,
    "attestation recensement.pdf ": 850
                },
 "Cours": {
    "NSI": {
       "TP.html ": 60,
       "dm.odt": 345
           },
    "Philo": {
      "Tractatus logico-philosophicus.epub": 2600
            }
          },
  "liste de courses.txt ": 24
           }
```

2.

a. On donne la fonction `Parcourir` suivante qui prend en paramètres un dossier racine et une liste représentant une adresse, et qui renvoie le contenu du dossier cible correspondant à l'adresse.

Exemple : Si la variable `Documents` contient le dictionnaire de l'exemple de la question 1 alors `Parcourir(Documents, ["Cours", "Philo"])` renvoie le dictionnaire `{"Tractatus logico-philosophicus.epub": 2600}`

Recopier et compléter la ligne 4

```
1. def Parcourir(racine, adr):
2.     dossier = racine
3.     for nom_dossier in adr:
4.         dossier = ..... # A compléter
5.     return dossier
```

b. Soit la fonction suivante :

```
1. def Afficher(racine, adr, nom_fichier):
2.     dossier = Parcourir(racine, adr)
3.     print(dossier[nom_fichier])
```

Qu'affiche l'instruction `Afficher(Documents, ["Cours", "NSI"], "TP.html")` sachant que la variable `Documents` contient le dictionnaire de la question 1 ?

3.

a. La fonction `Ajouter(racine, adr, nom_fichier, taille)` suivante ajoute au dictionnaire `racine`, à l'adresse `adr`, la clé `nom_fichier` associé à la valeur `taille`.

Une ligne de la fonction donnée ci-dessous contient une erreur. Laquelle ? Proposer une correction.

```
1. def Ajouter_fichier(racine, adr, nom_fichier, taille):
2.         dossier = Parcourir(racine, adr)
3.         taille = dossier[nom_fichier]
```

b. Ecrire une fonction `Ajouter_dossier(racine, adr, nom_dossier)` pour créer un dictionnaire représentant un dossier vide appelé `nom_dossier` dans le dictionnaire `racine` à l'adresse `adr`

4. Ecrire une fonction `taille(dossier)` qui prend en paramètre un dictionnaire `dossier` modélisant le contenu du répertoire `dossier` et qui renvoie le total de l'espace mémoire occupé par les fichiers contenus dans le dossier. On considère que le répertoire `dossier` ne contient que des fichiers et aucun sous-dossier.