# Formation Python, administration système
# Sujet des travaux pratiques

Matthieu Falce

Décembre 2024

## 1 Manipulation de la syntaxe python

### 1.1 *Fizz Buzz*

Concepts : 1) variables 2) boucles 3) conditions

Implémentez le code correspondant à l'algorithme suivant :

*Pour les nombres de 1 à 100,*
*si le nombre est divisible par 3, écrivez Fizz ;*
*s'il est divisible par 5, écrivez Buzz ;*
*s'il est divisible par les deux, écrivez FizzBuzz ;*
*sinon écrivez le nombre.*

### 1.2 Plus ou moins

Concepts : 1) variables 2) boucles 3) conditions 4) IO 5) exceptions

Vous allez coder un jeu simple pour enfant.

Étapes :

1. l'ordinateur choisit un nombre et l'utilisateur doit le deviner
2. l'ordinateur demande à l'utilisateur d'entrer un nombre
3. l'ordinateur ne peut dire que *c'est plus* ou *c'est moins*
4. Le jeu s'arrête quand l'utilisateur a trouvé le bon nombre.

Questions :

1. Comment pouvez-vous gérer les entrées invalides ?

### 1.3 Slices

Concepts : 1) structures de données 2) découpages 3) gestion des exceptions

Nous allons réimplémenter plusieurs commandes *shell* permettant d'afficher le contenu d'un fichier de différentes façons.

Questions :

1. implémenter la commande `unix cat` (affiche le contenu dans la console)
2. implémenter la commande `unix tac` (affiche le contenu à l'envers)
3. implémenter la commande `unix head` (affiche les N premières lignes du fichier)
4. implémenter la commande `unix tail` (affiche les N dernières lignes du fichier)
5. n'afficher qu'une ligne sur 3 entre la 5$^{e}$ ligne et la 10$^{e}$ ligne avant la fin
6. qu'en conclure sur les performances ? Comment faire avec de gros gros fichiers ?

## 1.4 Magic 8 ball

Concepts : 1) aléatoire 2) types 3) manipulation de fichiers 4) manipulation de chaînes

Vous n'arrivez pas à vous décider. Un programme peut vous aider à vous guider dans la vie.

Étapes :

1. chargez les phrases du fichier `exercices/python/magic_8_ball/media/phrases_magic_8_ball.txt`
2. affichez en une au hasard

## 1.5 Comparaison de textes

Concepts : 1) manipulation de fichiers sur le réseau 2) familiarisation avec les types de bases (ensembles, dictionnaires) 3) découverte de la bibliothèque standard

Vous allez récupérer 2 livres du projet Gutemberg.

— `https://www.gutenberg.org/cache/epub/51804/pg51804.txt`

— `https://www.gutenberg.org/files/53311/53311-0.txt`

Questions :

1. comment télécharger depuis internet ?
2. quels sont les mots qui apparaissent dans le texte ?
3. lister les mots qui n'apparaissent qu'une fois dans chaque texte. Si l'on regroupe les textes ?
4. lister les mots communs aux deux textes
5. quels sont les 10 mots qui apparaissent le plus ?

Réfléchissez à la complexité algorithmique ($\mathcal{O}(n)$, $\mathcal{O}(n^2)$, $\mathcal{O}(\log(n))$) de votre solution.

## 1.6 C'est loooong

Concepts : 1) fonctions d'ordre supérieur 2) décorateur 3) gestionnaire de contexte (context manager)

Mesurez de façon générique le temps d'exécution d'une fonction.
La modification doit être la moins invasive possible.

Étapes :

1. codez une fonction qui fait une pause de 1s
2. créez un décorateur de mesure du temps
3. créez un context manager de mesure du temps

Questions :

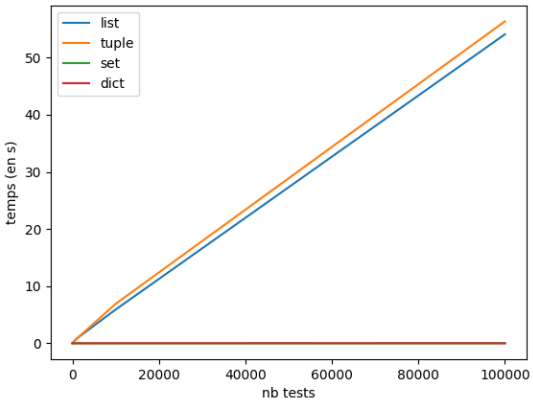
1. comment faire une pause dans un programme en Python ?
2. comment mesurer le temps d'execution d'un bout de programme en Python ?
3. comment faire des analyses statistiques plus poussées sur le temps d'exécution ?
4. Des solutions inclues dans python existent-t-elles ?

## 1.7 Analyse de complexité

Concepts : 1) structures de données 2) complexité

Nous allons analyser la complexité algorithmique des opérations `in` (*est présent*) pour divers conteneurs de la bibliothèque standard (`list`, `tuple`, `dict`, `set`).
Pour cela, il faut mesurer un temps d'exécution en fonction de la taille des paramètres en entrée.

Un graphique d'analyse de complexité est présent à droite




Questions :

1. quel protocole d'analyse utiliser ?
2. comment l'implémenter ?
3. qu'en conclure ? (qu'en dit la documentation)
4. (pro) comment réaliser ce graphique ?

# 2 Programmation orientée objet

## 2.1 Formation

Dans cette section vous allez améliorer le même code petit à petit afin d'avoir une application permettant de gérer les formations.

### 2.1.1 Tout commence par des humains

Concepts : 1) création de classes 2) méthodes et attributs 3) méthodes magiques

Créez une classe `Personne`. Elle va contenir les informations d'une personne (en gros les informations d'une carte d'identité).
Une personne doit pouvoir dire si elle est majeure. Attention, la limite d'âge de la majorité peut changer et doit donc être dynamique.

```
p1 = Personne(naissance=1990, nom="Matthieu")
p2 = Personne(naissance=2015, nom="Paul")

assert p1.est_majeur()
assert not p2.est_majeur()

Personne.AGE_MAJORITE =40

print(p1.est_majeur() == False)
print(p2.est_majeur() == False)
```

La personne doit pouvoir se présenter en faisant :

```
p1 = Personne(naissance=annee_naissance, nom=nom)
print(p1)
```

### 2.1.2 Attributs, Getters, Setters

Concepts : 1) `properties`

L'age d'un `Eleve` ne se recalcule pas. Comment régler ce problème de la façon la moins intrusive possible ?

On veut pouvoir faire

```
p1 = Personne("1", 1950)
print(p1.age == 68)

p1.age = 10
print(p1.annee_naissance == 2008)

p1.annee_naissance = 0
print(p1.age == 2018)
```

#### 2.1.3 Puis des élèves et des profs

Concepts : 1) héritage 2) surcharge

Dessinez les diagrammes de classes correspondant à ce qui est demandé.
Créez `Eleve` et `Prof` qui sont des `Personnes`.
Un `Prof` doit pouvoir dire si un élève a payé ses frais de scolarité.

#### 2.1.4 Et enfin, une formation

Concepts : 1) composition

Une `Formation`, c'est des élèves et des profs en même temps.
Un élève doit être majeur et avoir payé ses frais de scolarité pour participer. Modifiez le prof pour qu'il sache si un élève peut s'inscrire ou pas.

### 2.2 Convertisseur de température

Concepts : 1) `property` 2) accès aux attributs

Comment implémenter un convertisseur de température Celsius / Fahrenheit ?
Pour rappel :

$$T_{[°C]} = (T_{[°F]} - 32) * 5/9$$
$$T_{[°F]} = T_{[°C]} * 9/5 + 32$$

Questions :

1. essayer de gérer 2 attributs simultanément. Quels problèmes rencontre-t-on ?
2. essayer la même chose en utilisant les `properties`. Quel est l'intérêt de cette solution ?

## 3 Modules

### 3.1 Bases de données

Concepts : 1) manipulation de la DB api

Vous allez réaliser une application en ligne de commande permettant de stocker les notes d'un utilisateur dans une base de données.

Questions :

1. quelles sont les informations à stocker ?
2. comment effectuer des recherches spécifiques ?

### 3.2 Expressions régulières

**Tous, trouvez les tous**
Vous allez travailler sur le fichier : `medias/modules/pythonVersions.txt`

Questions :

1. quelles sont les versions de python présentes dans le fichier ?
2. comment peut on savoir laquelle est la plus présente ?

**Cherchez / trouvez**
Concepts : 1) syntaxe des expressions régulières 2) manipulation des expressions régulières

Trouvez les occurrences et positions des mots “chat” dans le texte `medias/modules/texteRegex.txt` .

Questions :

1. Comment modifier l’expression pour reconnaître aussi le mot “cat”.
2. Comment modifier l’expression pour ne pas reconnaître les mots contenant le mot “chat ou “cat (cathédrale par exemple) ?
3. est-ce que les expressions régulières sont plus adaptées que les outils de manipulation de chaînes inclus en python dans ce cas ?

**Identifiants**
Nous avons des chaines de cette forme ci :

— `id/233b6a88-fd22-4e28-a636-6ebea175dc34/review`
— `id/f28b9b71-ba38-4/review`
— `id/40312d67-85c2-4/review`
— `id/48a8da09-1ae8-474/review`
— `id/523c5cdf-d830-4b97-b/review`
— `id/124defcd-e4d1-4c2a/review`

Questions :

1. Voyez-vous un motif dans ces chaînes ?
2. Comment extraire l’information stockée ?

**C’est valide**

Forme canonique d’une adresse email : `identifiant@domaine.tld`

Questions :

1. Comment peut-on imaginer valider une adresse e-mail ?
2. Imaginez qu’un utilisateur entre une adresse, comment vérifier si elle correspond à la forme canonique ?
3. Comment se servir de cela pour valider les mails présents dans `medias/modules/listingMail.json` ?

## 4 Administration système

Cette section comporte une sorte de projet pour faire découvrir la puissance de python dans le cadre de l’administration système (manipulation de textes, de fichiers, de scripts...).

## 4.1 Manipulation de chaines de charactères

Concepts : 1) manipulation de fichiers 2) manipulation de string 3) manipulation d'expressions régulières 4) formatage de résultats

Nous avons accès au fichier de log d'une base de donnée `meilisearch`. Nous voulons extraire des informations de ce fichier

Étapes :

1. ouvrez le fichier `medias/administrationSysteme/analyseTexte/meilisearch.log`
2. utilisez python pour déterminer les informations suivantes
   - afficher le numéro de version de l'outil
   - afficher le port d'écoute de l'outil
   - quel jour de la semaine l'outil a-t-il été lancé
3. mettez en forme ces données de façon structurée

## 4.2 Manipulation de fichiers

Concepts : 1) manipulation de fichiers 2) manipulation de zips 3) utilisation de bibliothèques 4) envoie d'emails

Nous allons utiliser Python pour manipuler des fichiers (accéder à leur nom, déterminer leur type, les renommer, les déplacer, les archiver).
Nous disposons d'un fichier zip contenant de nombreux fichiers dont les extensions ont été changées.

Étapes :

1. dézipper (en utilisant python) le fichier `medias/administrationSysteme/manipulationFichiers/fichiersVrac.tar.gz`
2. retrouvez les extensions correctes des fichiers à l'aide de leur `magic number` [1]
3. réordonnez les fichiers en les regroupant par extension `recovered/extension/filename` (par exemple `recovers/jpg/toto.jpg`)
4. zippez le contenu de `recovered`
5. envoyez ce zip par mail (vous pouvez créer un serveur SMTP de test avec `python -m smtpd -c DebuggingServer -n localhost:1025`)

## 4.3 Manipulation d'adresse IP

Concepts : 1) manipulation d'entiers (décimaux et binaires), de chaines, de classes et de conteneurs 2) utilisation de bibliothèques python 3) lecture de documentation 4) générateurs et `yield` 5) réalisation de tests unitaires

Vous allez devoir faire un générateur d'adresses IP disponibles à partir d'une adresse réseau et d'un masque réseau.
Par exemple l'adresse `192.168.1.0` avec le masque `255.255.255.0` va donner les adresses `192.168.1.0` à `192.168.1.255`.
Nous voulons générer cette liste. De même nous voulons savoir si les adresses `192.168.1.65` et `192.168.0.55` sont dans le sous réseau.
Les détails algorithmiques se trouvent expliqués ici : `https://www.baeldung.com/cs/get-ip-range-from-subnet-mask`. Il vous faudra trouver comment générer la première adresse et la dernière de la plage de votre sous réseau.
Je veux pouvoir utiliser la bibliothèque en faisant :

```
ip = "192.168.0.1"
mask = "255.255.255.0"
network = NetworkIP(ip, mask)
for ip in network.list_ips_in_network():
print(ip)

for ip in network:
print(ip)
```

1. `https://en.wikipedia.org/wiki/List_of_file_signatures`

```
assert  network.is_ip_in_network("192.168.0.34")
assert "192.168.0.34" in network

ip = "192.168.0.1/21"
network = NetworkIP(ip)
for ip in network:
print(ip)
```

Étapes :

1. créez des fonctions vous permettant de transformer une IP de chaîne à entier (avec sa représentation binaire) et réciproquement.
2. trouvez comment faire les `ET` et `OU` binaires en python
3. créez une classe pour encapsuler toutes ces étapes

Questions :

1. comment faire pour utiliser directement notre objet dans un `in` ou un `for` ?
2. regardez la bibliothèque `https://docs.python.org/3/library/ipaddress.html`. Peut-elle vous simplifier la vie ? Est-ce qu'utiliser cette bibliothèque change forcément l'API de la classe que nous venons de créer ?

## 4.4 Manipulation de logs

Concepts : 1) manipulation de fichiers 2) manipulation de zips 3) manipulation d'expressions régulières 4) formatage de résultats 5) utilisation de bibliothèques

Nous avons accès au fichier de log d'un serveur web `apache` (common log format). Nous voulons extraire des informations de ce fichier. Les données ont été trouvées ici : `https://github.com/elastic/examples/tree/master/Common%20Data%20Formats/apache_logs`.
Une ligne de log apache ressemble à ceci :
`38.99.236.50 - - [20/May/2015:21:05:31 +0000] "GET /favicon.ico HTTP/1.1" 200 3638 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.107 Safari/537.36"`.
Elle contient les informations suivantes [2] :

— l'IP du client qui s'est connecté
— la date de la requête
— les informations sur la ressource demandée (verb + URI)
— le status HTTP de la requête
— le poids de la réponse délivrée (en octets)
— le `referer` de la page (quel état le site qui a dirigé vers la ressource)
— le `user agent` du client qui a effectué la requête

Étapes :

1. dézipper (en utilisant python) le fichier `medias/administrationSysteme/analyseLogs/apache.zip` et parcourez les logs
2. extrayez les données de chaque lignes
3. enrichissez ces données à l'aide de bibliothèques python
   — déterminer l'OS et le navigateur (parsing du `User Agent`)
   — déterminez le pays (en utilisant l'`IP`, la bibliothèque `geoip2` et la base de données `medias/analyseLogs/GeoLite2-Country_20220125.tar`, qu'il faudra dézipper)
4. mettez en forme ces données de façon structurée, vous pouvez utiliser la bibliothèque Rich `https://rich.readthedocs.io/en/stable/introduction.html` ou TDQM `https://tqdm.github.io/` pour égayer votre ligne de commande.
   — trouvez le top 10 des pays qui font des requêtes
   — combien de requêtes débouchant sur une erreur 500 y a t-il eu ?

2. `https://httpd.apache.org/docs/2.4/fr/logs.html`

— quelle est la taille totale échangée durant la durée des logs ?
— sur combien de jours les logs portent-ils ?

5. maintenant que ces logs sont structurés, vous pouvez les insérer à une base de données SQLite
6. permettez l'utilisation de l'outil en ligne de commande (avec la bibliothèque `click`) et permettez sa distribution / installation par des tiers

## 4.5 Extraction de données (`scrapping`)

Concepts : 1) utilisation de framework 2) manipulation de HTML

Vous devez extraire toutes les citations de ce site : `http://quotes.toscrape.com/` pour les récupérer dans des fichiers CSV et JSON.

Étapes :

1. utilisez scrapy en suivant le tutoriel situé ici : `https://docs.scrapy.org/en/latest/intro/tutorial.html`
2. refaites la même manipulation en utilisant la bibliothèque `requests` (`https://docs.python-requests.org/en/latest/`)

Questions :

1. dans les deux cas comment fonctionne la recherche des éléments d'intérêts ?
2. quel est l'intérêt de `scrapy` par rapport à `requests` et réciproquement