



# Formation Python

*Partie 3 : StdLib*

# StdLib

## Définition

Trois types de modules :

1. Modules que nous créons nous même.
2. Modules tiers.
3. Modules de la bibliothèque standard

Les modules de la bibliothèque standard sont inclus dans Python et contiennent de nombreux outils utiles. Ils sont maintenus par l'équipe de développement de Python, vous pouvez donc compter sur leur fiabilité.

Remarque : les paquets et modules de la bibliothèque standard ne sont PAS la même chose que les objets intégrés (p. ex. print, open, zip, enumerate). Vous devez toujours importer les modules/packages de la bibliothèque standard, mais vous n'avez pas besoin de les installer depuis un autre endroit.

# StdLib

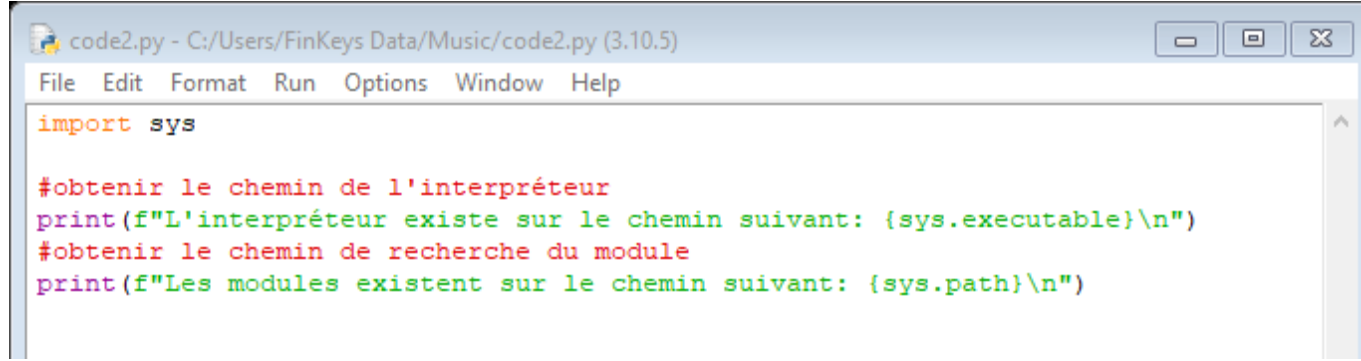
## Définition

Trois types de modules :

1. sys: fonctions et variables opérant sur l'interpréteur Python
2. os: la fonctionnalité du système d'exploitation
3. shutil: la manipulation des fichiers

# StdLib

## sys



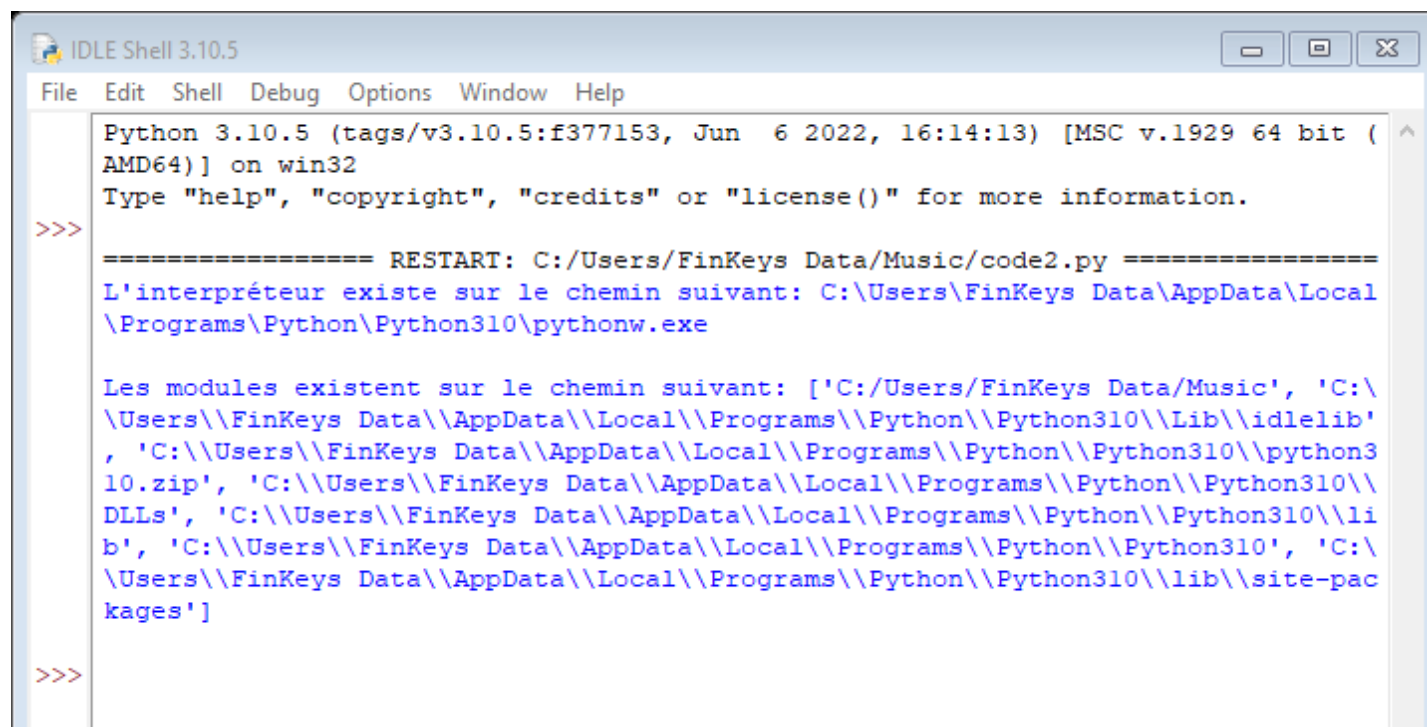
A screenshot of a Python IDE window titled "code2.py - C:/Users/FinKeys Data/Music/code2.py (3.10.5)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code editor displays the following Python code:

```
import sys

#obtenir le chemin de l'interpréteur
print(f"L'interpréteur existe sur le chemin suivant: {sys.executable}\n")
#obtenir le chemin de recherche du module
print(f"Les modules existent sur le chemin suivant: {sys.path}\n")
```

# StdLib

sys

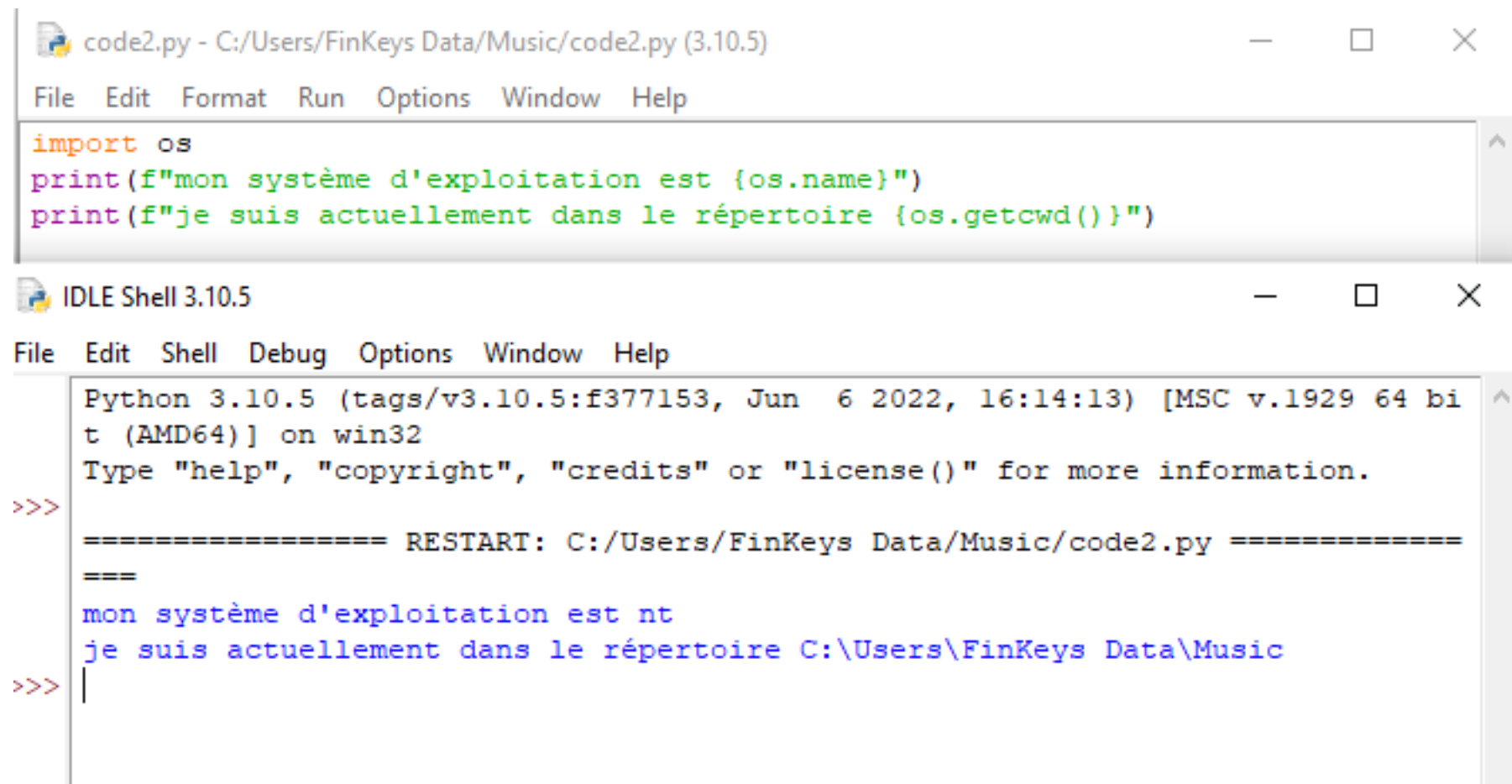


```
IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/FinKeys Data/Music/code2.py =====
L'interpréteur existe sur le chemin suivant: C:\Users\FinKeys Data\AppData\Local\Programs\Python\Python310\pythonw.exe

Les modules existent sur le chemin suivant: ['C:/Users/FinKeys Data/Music', 'C:\Users\FinKeys Data\AppData\Local\Programs\Python\Python310\Lib\idlelib', 'C:\\Users\\FinKeys Data\\AppData\\Local\\Programs\\Python\\Python310\\python310.zip', 'C:\\Users\\FinKeys Data\\AppData\\Local\\Programs\\Python\\Python310\\DLLs', 'C:\\Users\\FinKeys Data\\AppData\\Local\\Programs\\Python\\Python310\\lib', 'C:\\Users\\FinKeys Data\\AppData\\Local\\Programs\\Python\\Python310', 'C:\\Users\\FinKeys Data\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages']
>>>
```

# StdLib

## OS



The image shows a screenshot of a Python IDE with two windows. The top window, titled 'code2.py - C:/Users/FinKeys Data/Music/code2.py (3.10.5)', contains the following Python code:

```
import os
print(f"mon système d'exploitation est {os.name}")
print(f"je suis actuellement dans le répertoire {os.getcwd()}")
```

The bottom window, titled 'IDLE Shell 3.10.5', shows the output of running the script. It displays the Python version and architecture, followed by the execution of the script's print statements:

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bi
t (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/FinKeys Data/Music/code2.py =====
===
mon système d'exploitation est nt
je suis actuellement dans le répertoire C:\Users\FinKeys Data\Music
>>>
```

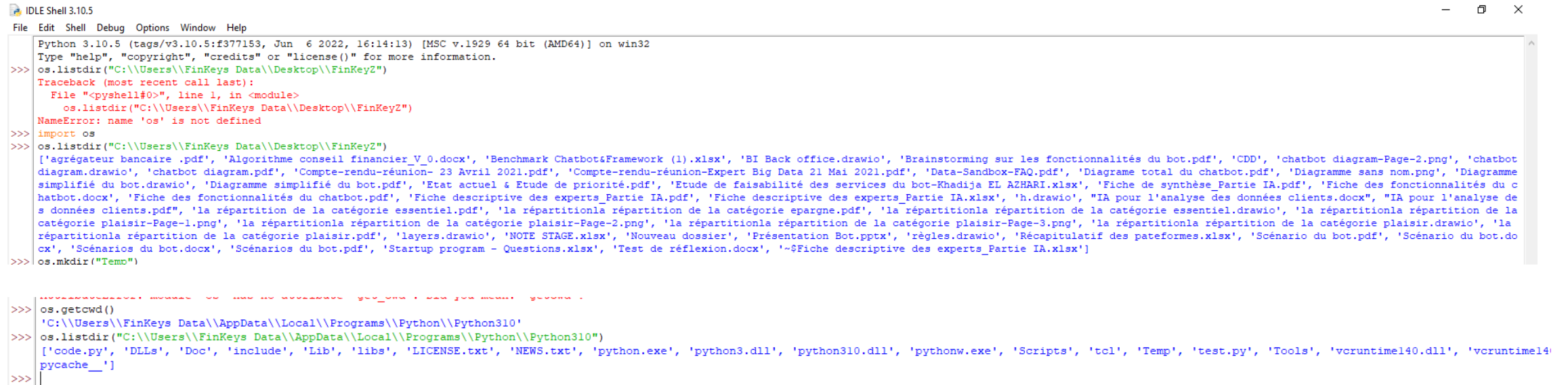
# StdLib

## OS

```
>>> os.chdir("C:\\Users\\FinKeys Data\\Downloads")
>>> os.getcwd()
'C:\\Users\\FinKeys Data\\Downloads'
>>> os.listdir()
['.ipynb_checkpoints', '01 (1).docx', '01 a valider', '01 a valider (1)', '01 a valider (1).zip', '01 a valider (2)', '01 a valider (2).zip', '01 a valider .zip', '01.docx', '02.docx', '03 (1).docx', '03.docx', '03_valides', '03_valides.zip', '060921-174.pdf', '1-s2.0-S2666920X21000278-main.pdf', '1. Méthodes de collecte_Introduction et généralités.pdf', '1.png', '10.1109_ECAI46879.2019.9042015.ris', '10288-33938-1-PB.pdf', '111 (1).xlsx', '111.xlsx', '147-exemple-cv-gratuit-a-telecharger.docx', '15ale08705b121c1b7bd-3fb3ee3ef7a59795b024020b09c3ce2ac60c4ca5', '15ale08705b121c1b7bd-3fb3ee3ef7a59795b024020b09c3ce2ac60c4ca5.zip', '1698519.pdf', '19Q4361662_19Q04361662_5_09_05_2022.pdf', '19Q4361662_19Q04361662_5_15_11_2021 (1).pdf', '19Q4361662_19Q04361662_5_15_11_2021 (2).pdf', '19Q4361662_19Q04361662_5_15_11_2021 (3).pdf', '19Q4361662_19Q04361662_5_15_11_2021 (4).pdf', '19Q4361662_19Q04361662_5_15_11_2021 (5).pdf',
```

# StdLib

## OS



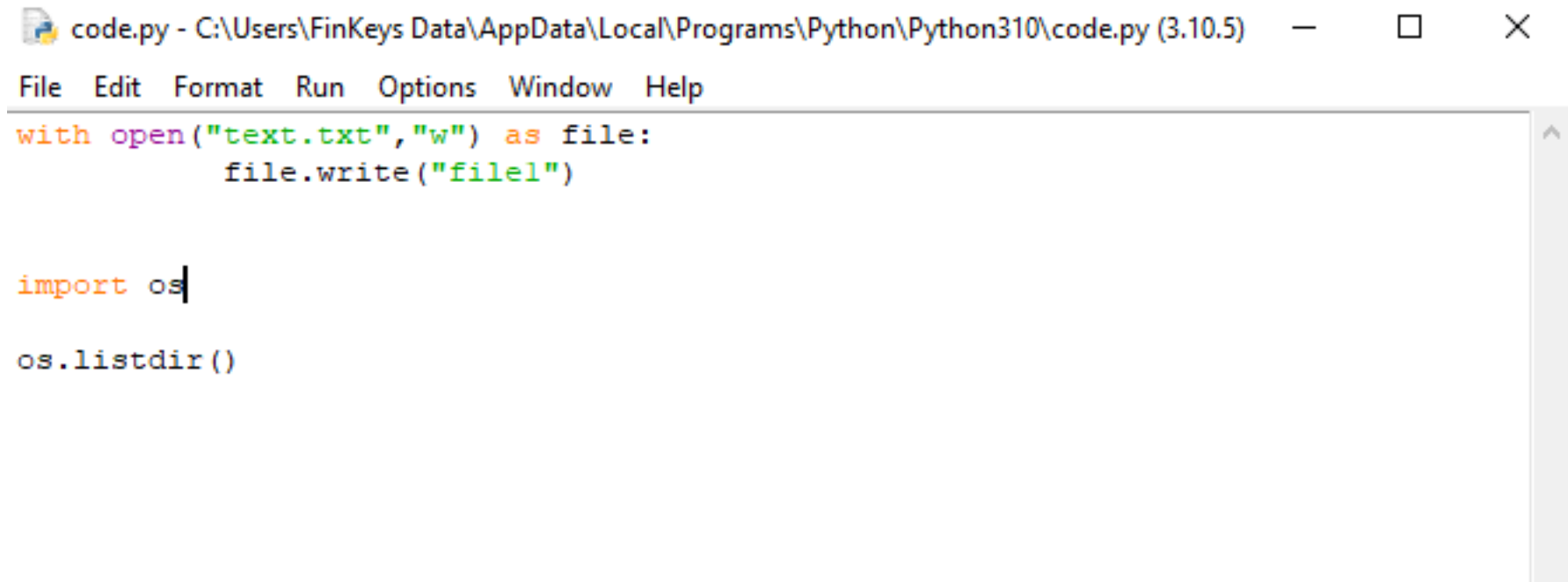
```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> os.listdir("C:\\Users\\FinKeys Data\\Desktop\\FinKey2")
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    os.listdir("C:\\Users\\FinKeys Data\\Desktop\\FinKey2")
NameError: name 'os' is not defined
>>> import os
>>> os.listdir("C:\\Users\\FinKeys Data\\Desktop\\FinKey2")
['agrégateur bancaire .pdf', 'Algorithme conseil financier_V_0.docx', 'Benchmark Chatbot&Framework (1).xlsx', 'BI Back office.drawio', 'Brainstorming sur les fonctionnalités du bot.pdf', 'CDD', 'chatbot diagram-Page-2.png', 'chatbot diagram.drawio', 'chatbot diagram.pdf', 'Compte-rendu-réunion- 23 Avril 2021.pdf', 'Compte-rendu-réunion-Expert Big Data 21 Mai 2021.pdf', 'Data-Sandbox-FAQ.pdf', 'Diagrame total du chatbot.pdf', 'Diagramme sans nom.png', 'Diagramme simplifié du bot.drawio', 'Diagramme simplifié du bot.pdf', 'Etat actuel & Etude de priorité.pdf', 'Etude de faisabilité des services du bot-Khadija EL AZHARI.xlsx', 'Fiche de synthèse_Partie IA.pdf', 'Fiche des fonctionnalités du c hatbot.docx', 'Fiche des fonctionnalités du chatbot.pdf', 'Fiche descriptive des experts_Partie IA.pdf', 'Fiche descriptive des experts_Partie IA.xlsx', 'h.drawio', 'IA pour l'analyse des données clients.docx', 'IA pour l'analyse de s données clients.pdf', 'la répartition de la catégorie essentiel.pdf', 'la répartitionla répartition de la catégorie epargne.pdf', 'la répartitionla répartition de la catégorie essentiel.drawio', 'la répartitionla répartition de la catégorie plaisir-Page-1.png', 'la répartitionla répartition de la catégorie plaisir-Page-2.png', 'la répartitionla répartition de la catégorie plaisir-Page-3.png', 'la répartitionla répartition de la catégorie plaisir.drawio', 'la répartitionla répartition de la catégorie plaisir.pdf', 'layers.drawio', 'NOTE STAGE.xlsx', 'Nouveau dossier', 'Présentation Bot.pptx', 'règles.drawio', 'Récapitulatif des pateformes.xlsx', 'Scénario du bot.pdf', 'Scénario du bot.docx', 'Scénarios du bot.docx', 'Scénarios du bot.pdf', 'Startup program - Questions.xlsx', 'Test de réflexion.docx', '~$Fiche descriptive des experts_Partie IA.xlsx']
>>> os.mkdir("Temp")

>>> os.getcwd()
'C:\\Users\\FinKeys Data\\AppData\\Local\\Programs\\Python\\Python310'
>>> os.listdir("C:\\Users\\FinKeys Data\\AppData\\Local\\Programs\\Python\\Python310")
['code.py', 'DLLs', 'Doc', 'include', 'Lib', 'libs', 'LICENSE.txt', 'NEWS.txt', 'python.exe', 'python3.dll', 'python310.dll', 'pythonw.exe', 'Scripts', 'tcl', 'Temp', 'test.py', 'Tools', 'vcruntime140.dll', 'vcruntime140.pycache__']
>>>
```



# StdLib

## OS



```
code.py - C:\Users\FinKeys Data\AppData\Local\Programs\Python\Python310\code.py (3.10.5)
File Edit Format Run Options Window Help
with open("text.txt","w") as file:
    file.write("file1")

import os

os.listdir()
```

# StdLib

## shutil

```
>>> os.listdir()
['code.py', 'DLLs', 'Doc', 'include', 'Lib', 'libs', 'LICENSE.txt', 'NEWS.txt', 'python.exe', 'python3.dll', 'python310.dll', 'pythonw.exe', 'Scripts', 'tcl', 'Temp', 'test.py', 'text.txt',
_l.dll', '__pycache__']
>>> import shutil
      .
      .
      .

>>> shutil.copyfile("text.txt", "test_copy.txt")
'test_copy.txt'
>>> os.listdir()
['code.py', 'DLLs', 'Doc', 'include', 'Lib', 'libs', 'LICENSE.txt', 'NEWS.txt', 'python.exe', 'python3.dll', 'python310.dll', 'pythonw.exe', 'Scripts', 'tcl', 'Temp', 'test.py', 'test_copy.txt', 'text.txt',
l', 'vcruntime140_1.dll', '__pycache__']

>>> shutil.rmtree('TEMP')
>>> os.listdir()
['code.py', 'DLLs', 'Doc', 'include', 'Lib', 'libs', 'LICENSE.txt', 'NEWS.txt', 'python.exe', 'python3.dll', 'python310.dll', 'pythonw.exe', 'Scripts', 'tcl', 'test.py', 'test_copy.txt', 'text.txt', 'Tools',
untimel40_1.dll', '__pycache__']
>>> os.remove("test_copy.txt")
>>> os.listdir()
['code.py', 'DLLs', 'Doc', 'include', 'Lib', 'libs', 'LICENSE.txt', 'NEWS.txt', 'python.exe', 'python3.dll', 'python310.dll', 'pythonw.exe', 'Scripts', 'tcl', 'test.py', 'text.txt', 'Tools', 'vcruntime140.d
__pycache__']
>>> |
```

# StdLib

## os.path

### os.path pour les chemins de fichiers

\* Faire `import os.path`

\* Fonctions de manipulation de fichiers :

- `os.path.sep` : le séparateur dans un path ('/' sous linux par exemple).
- `os.path.basename('/myDir/myDir2/myFile')` donne `myFile`, même si le fichier n'existe pas sur le disque.
- `os.path.dirname('/myDir/myDir2/myFile')` donne `/myDir/myDir2`, même si le fichier n'existe pas sur le disque.
- `os.path.splitext('myFile.txt')` : renvoie un tuple `('myFile', '.txt')` :
  - si le fichier n'a pas d'extension, le 2ème élément est vide.
  - si le fichier a plusieurs points dans son nom, seul le dernier est considéré comme étant l'extension
- `os.path.split('/myDir/myDir2/myFile')` renvoie la paire `('myDir/myDir2', 'myFile')`.
- `os.path.exists('/myDir/myFile')` : renvoie True si le fichier ou directory existe. Si c'est un lien symbolique qui pointe vers un fichier qui n'existe pas, renvoie False.
- `os.path.lexists('/myDir/myFile')` : comme `os.path.exists`, sauf que si c'est un lien symbolique qui pointe vers un fichier qui n'existe pas, ça renvoie True au lieu de False avec `os.path.exists`.
- `os.path.expanduser('~myUser/myFile')` remplace le `~myUser` ou le `~` par le chemin vers le home et renvoie le résultat..
- `os.path.expandvars('/homez.489/duclert/myFile')` remplace les variables d'environnement par leur valeur et renvoie le resultat.
- `os.path.getmtime('myFile')` : renvoie la date de modification du fichier (exception si fichier n'existe pas), sous forme de timestamp (utiliser `datetime.datetime.fromtimestamp` pour le convertir en date et temps). Utiliser `os.path.getatime()` pour la date de dernier accès.
- `os.path.getctime()` : date de dernière modification du fichier ou de l'inode. Par exemple, si on change l'owner ou les droits sans changer le contenu du fichier, `getctime()` est mis à jour alors que `getmtime()` reste inchangé.
- `os.path.getsize('myFile')` : renvoie la taille (exception si n'existe pas)
- `os.path.isfile`, `os.path.isdir`, `os.path.islink` : renvoie True si c'est le cas.
- `os.path.join('/myDir/myDir2', 'myDir3/', 'myFile')` : renvoie `/myDir/myDir2/myDir3/myFile` (sans vérifier si le path existe).
- `os.path.normpath('/myDir//myDir2../myFile')` : normalise le path, par exemple, donne ici `/myDir/myFile` (sans vérifier si le path existe).
- `os.path.realpath` : renvoie le chemin complet en déréférençant les liens symboliques.

# StdLib

os.path

[https://python.developpez.com/cours/DiveIntoPython/php/frdiveintopython/file\\_handling/os\\_module.php](https://python.developpez.com/cours/DiveIntoPython/php/frdiveintopython/file_handling/os_module.php)

# StdLib

os.path

[https://python.developpez.com/cours/DiveIntoPython/php/frdiveintopython/file\\_handling/os\\_module.php](https://python.developpez.com/cours/DiveIntoPython/php/frdiveintopython/file_handling/os_module.php)

# Expressions régulières

## Définition : search

Une expression régulière est une suite de caractères qui a pour but de décrire un fragment de texte. Cette suite de caractères est encore appelée motif (en anglais pattern).

Le module re permet d'utiliser des expressions régulières avec Python. Les expressions régulières sont aussi appelées en anglais regular expressions ou en plus court regex.

```
>>> cours="Machine Learning big data python"
>>> import re
>>> re.search("Learning",cours)
<re.Match object; span=(8, 16), match='Learning'>
>>> if re.search("Learning",cours):
...     print("ok")
...
...
ok
>>> |
```

# Expressions régulières

## Match() & fullmatch()

La fonction *match()* dans le module *re* qui fonctionne sur le modèle de *search()*. La différence est qu'elle renvoie un objet du type *SRE\_Match* seulement lorsque la regex correspond au début de la chaîne de caractères (à partir du premier caractère).

La fonction *fullmatch()* qui renvoie un objet du type *SRE\_Match* si et seulement si l'expression régulière correspond exactement à la chaîne de caractères.

```
>>> re.match("Learning",cours)
>>> re.match("Machine",cours)
<re.Match object; span=(0, 7), match='Machine'>
>>> cours="Big Data  "
>>> re.fullmatch("Big Data  ",cours)
<re.Match object; span=(0, 10), match='Big Data  ' >
>>> re.fullmatch("Big Data",cours)
>>> |
```

# Expressions régulières

## Match() & fullmatch()

De manière générale, la fonction search() peut être utilisé :

1. Si on cherche une correspondance avec le début de la chaîne de caractères comme dans la fonction match(), on peut utiliser l'accroche de début de ligne ^.
2. Si on cherche une correspondance exacte comme dans la fonction fullmatch(), on peut utiliser les métacaractères ^ et \$, par exemple ^Big\$.

```
>>> cours="Big Data  "
>>> re.fullmatch("Big Data  ",cours)
<re.Match object; span=(0, 10), match='Big Data  '>
>>> re.fullmatch("Big Data",cours)
>>> re.search("^Big",cours)
<re.Match object; span=(0, 3), match='Big'>
>>> re.search("^Big$",cours)
>>> re.search("^Big Data  $",cours)
<re.Match object; span=(0, 10), match='Big Data  '>
>>> |
```



# Expressions régulières

## Compilation d'expressions régulières

Lorsqu'on a besoin de tester la même expression régulière sur plusieurs milliers de chaînes de caractères, il est pratique de compiler préalablement la regex à l'aide de la fonction `compile()` qui renvoie un objet de type `SRE_Pattern` :

```
>>> import re
>>> cours="Machine Learning"
>>> regex=re.compile("^Machine")
>>> regex.search(cours)
<re.Match object; span=(0, 7), match='Machine'>
```

# Expressions régulières

## Compilation d'expressions régulières

```
>>> import re
>>> cours="Machine Learning"
>>> regex=re.compile("^Machine")
>>> regex.search(cours)
<re.Match object; span=(0, 7), match='Machine'>
>>> regex.search("learning")
>>> regex=re.compile("([0-9]+)\.([0-9]+)")
>>> resultat=regex.search("pi vaut 3.14 et e vaut 7.8")
>>> resultat.group(0)
'3.14'
>>> resultat.group(1)
'3'
>>> resultat.group(2)
'14'
>>> resultat.start()
8
>>> resultat.end()
12
>>> |
```

# Expressions régulières

## Findall()

```
>>> resultat=regex.findall("pi vaut 3.14 et e vaut 7.8")
>>> resultat
[('3', '14'), ('7', '8')]
>>> regex=re.compile("[0-9]+\.[0-9]+")
>>> resultat=regex.findall("pi vaut 3.14 et e vaut 7.8")
>>> resultat
['3.14', '7.8']
>>> |
```

# Expressions régulières

## .sub()

La méthode `.sub()` permet d'effectuer des remplacements assez puissants. Par défaut la méthode `.sub (chaine1, chaine2)` remplace toutes les occurrences trouvées par l'expression régulière dans `chaine2` par `chaine1`. (On peut remplacer que les `n` premières occurrences, en utilisant l'argument `count=n`)

```
>>> regex=re.compile("[0-9]+\.[0-9]+")
>>> resultat=regex.sub("qlq chose","pi vaut 3.14 et e vaut 7.8")
>>> resultat
'pi vaut qlq chose et e vaut qlq chose'
>>> resultat=regex.sub("qlq chose","pi vaut 3.14 et e vaut 7.8",count=1)
>>> resultat
'pi vaut qlq chose et e vaut 7.8'
>>> |

>>> regex=re.compile('([0-9]+)\.([0-9]+)')
>>> resultat=regex.sub("un peu près \1","pi vaut 3.14 et e vaut 7.8",count=1)
>>> resultat
'pi vaut un peu près 3 et e vaut 7.8'
>>> |
```

# Utilisation xml

## Xml: extensible markup language

XML est un langage lisible par l'homme et par la machine. C'est un **un langage de balisage** qui permet de structurer, stocker et transférer des données. Python fournit un **grand nombre de bibliothèques** afin de lire, écrire, modifier et d'analyser des fichiers XML.

XML crée une structure arborescente selon une hiérarchie **facile à interpréter**. Un document XML est divisé en **sections, appelées éléments**, contenus entre **une balise de début et de fin**.

Une balise est une construction de balisage qui commence par **le signe <** et se termine par **le signe >**. De plus, les éléments peuvent contenir d'autres éléments, appelés **des éléments enfants**.

# Utilisation xml

## Xml: extensible markup language


L'élément de niveau supérieur s'appelle **la racine**, qui contient tous les autres éléments. Les attributs sont des paires **nom-valeur dans un élément**.

```
<?xml version="1.0" encoding="UTF-8"?>
<employe>
  <matricule>1234</matricule>
  <prenom>Jean</prenom>
  <nom>Tremblay</nom>
</employe>
```

# Utilisation xml

Xml: extensible markup language

XML

 Copier

```
<?xml version="1.0"?>
<Catalog>
  <Book id="bk101">
    <Author>Garghenti, Davide</Author>
    <Title>XML Developer's Guide</Title>
    <Genre>Computer</Genre>
    <Price>44.95</Price>
    <PublishDate>2000-10-01</PublishDate>
    <Description>An in-depth look at creating applications
    with XML.</Description>
  </Book>
  <Book id="bk102">
    <Author>Garcia, Debra</Author>
    <Title>Midnight Rain</Title>
    <Genre>Fantasy</Genre>
    <Price>5.95</Price>
    <PublishDate>2000-12-16</PublishDate>
    <Description>A former architect battles corporate zombies,
    an evil sorceress, and her own childhood to become queen
    of the world.</Description>
  </Book>
</Catalog>
```

# Utilisation xml

## Minidom & ElementTree

Le module XML Python prend en charge deux **sous-modules minidom** et **ElementTree** pour analyser un fichier XML.

Le module **minidom** ou **Minimal DOM** fournit une structure de type **DOM (Document Object Model)** pour analyser un fichier XML.

DOM : modèle d'objet de document, c'est le principe d'arborescence adopté par les documents xml avec une racine et des éléments, se trouvant à l'intérieur de balises.



# Utilisation xml

## Minidom & ElementTree

Le module XML Python prend en charge deux **sous-modules minidom** et **ElementTree** pour analyser un fichier XML.


Le module **minidom** ou **Minimal DOM** fournit une structure de type **DOM (Document Object Model)** pour analyser un fichier XML.

DOM : modèle d'objet de document, c'est le principe d'arborescence adopté par les documents xml avec une racine et des éléments, se trouvant à l'intérieur de balises.

**minidom** possède une fonction **parse()** qui permet de lire un fichier XML.

# Utilisation xml

## Minidom & ElementTree

 data - Bloc-notes

Fichier Edition Format Affichage Aide

---

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<info>
```

```
  <items>
```

```
    <item name="item1">1</item>
```

```
    <item name="item2">2</item>
```

```
    <item name="item3">3</item>
```

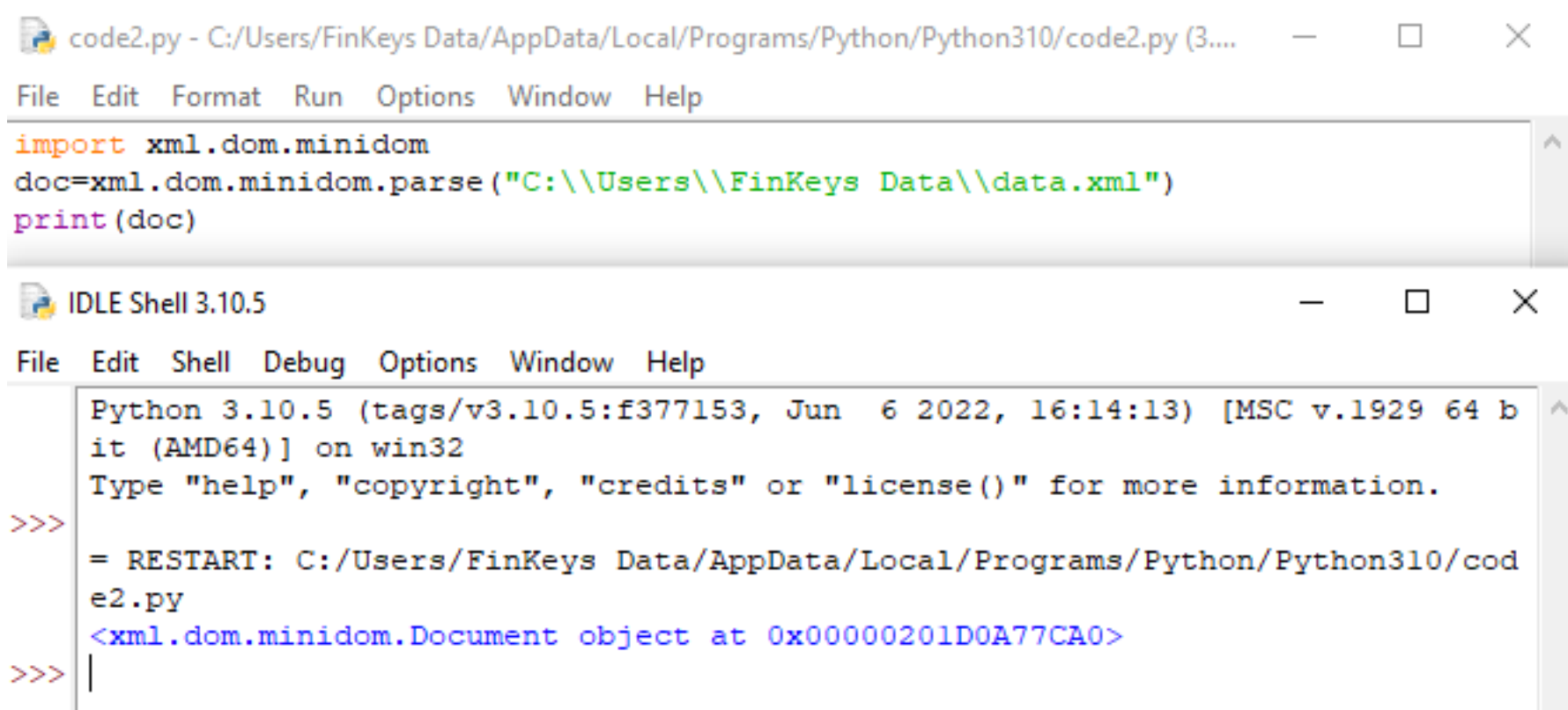
```
    <item name="item4">4</item>
```

```
  </items>
```

```
</info>|
```

# Utilisation xml

## Minidom & ElementTree



The image shows a screenshot of a Python IDE with two windows. The top window, titled 'code2.py', contains the following Python code:

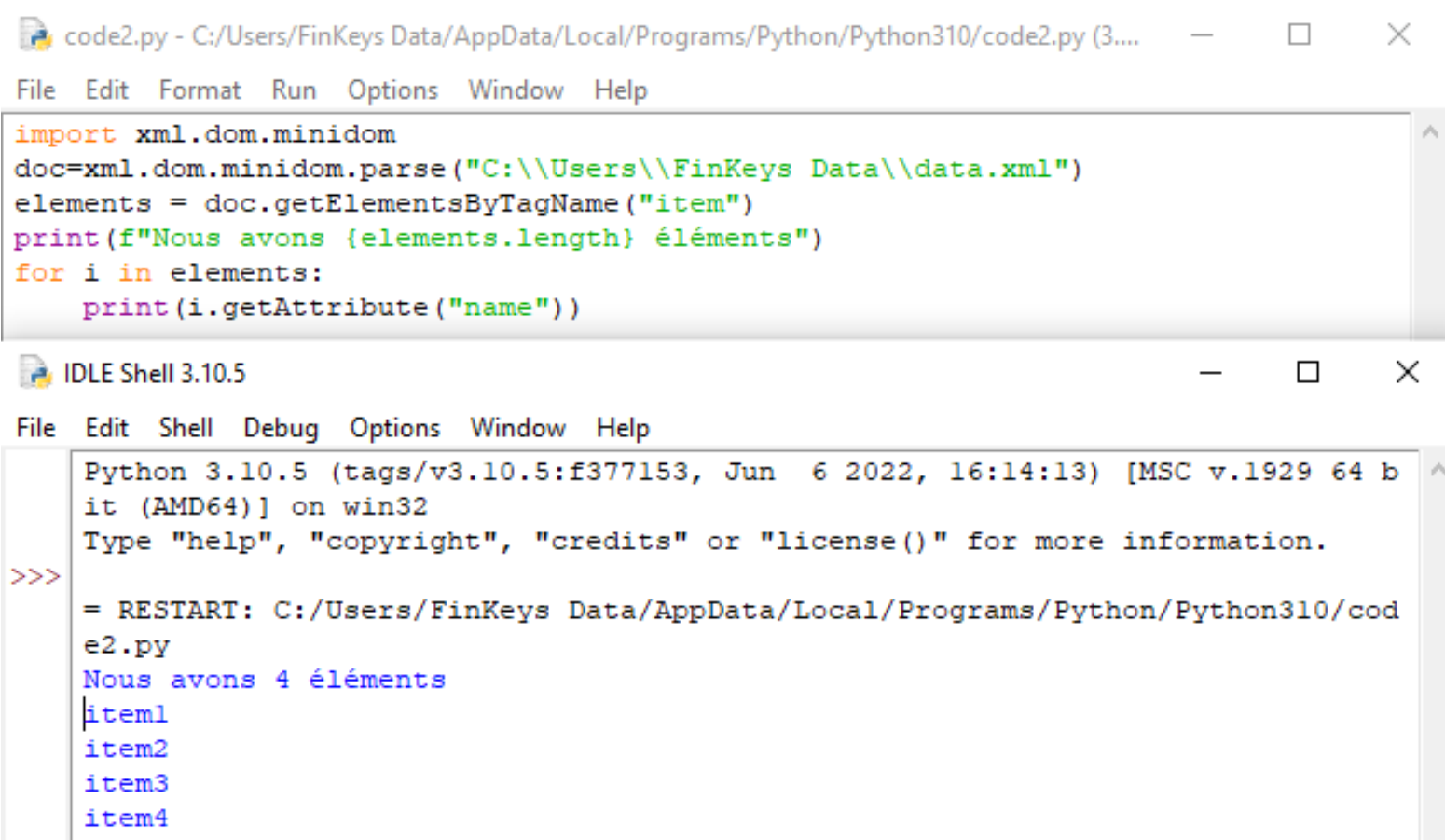
```
import xml.dom.minidom
doc=xml.dom.minidom.parse("C:\\Users\\FinKeys Data\\data.xml")
print(doc)
```

The bottom window, titled 'IDLE Shell 3.10.5', shows the output of the code execution:

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/cod
e2.py
<xml.dom.minidom.Document object at 0x00000201D0A77CA0>
>>> |
```

# Utilisation xml

## Minidom & ElementTree



```
code2.py - C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/code2.py (3...
File Edit Format Run Options Window Help
import xml.dom.minidom
doc=xml.dom.minidom.parse("C:\\Users\\FinKeys Data\\data.xml")
elements = doc.getElementsByTagName("item")
print(f"Nous avons {elements.length} éléments")
for i in elements:
    print(i.getAttribute("name"))

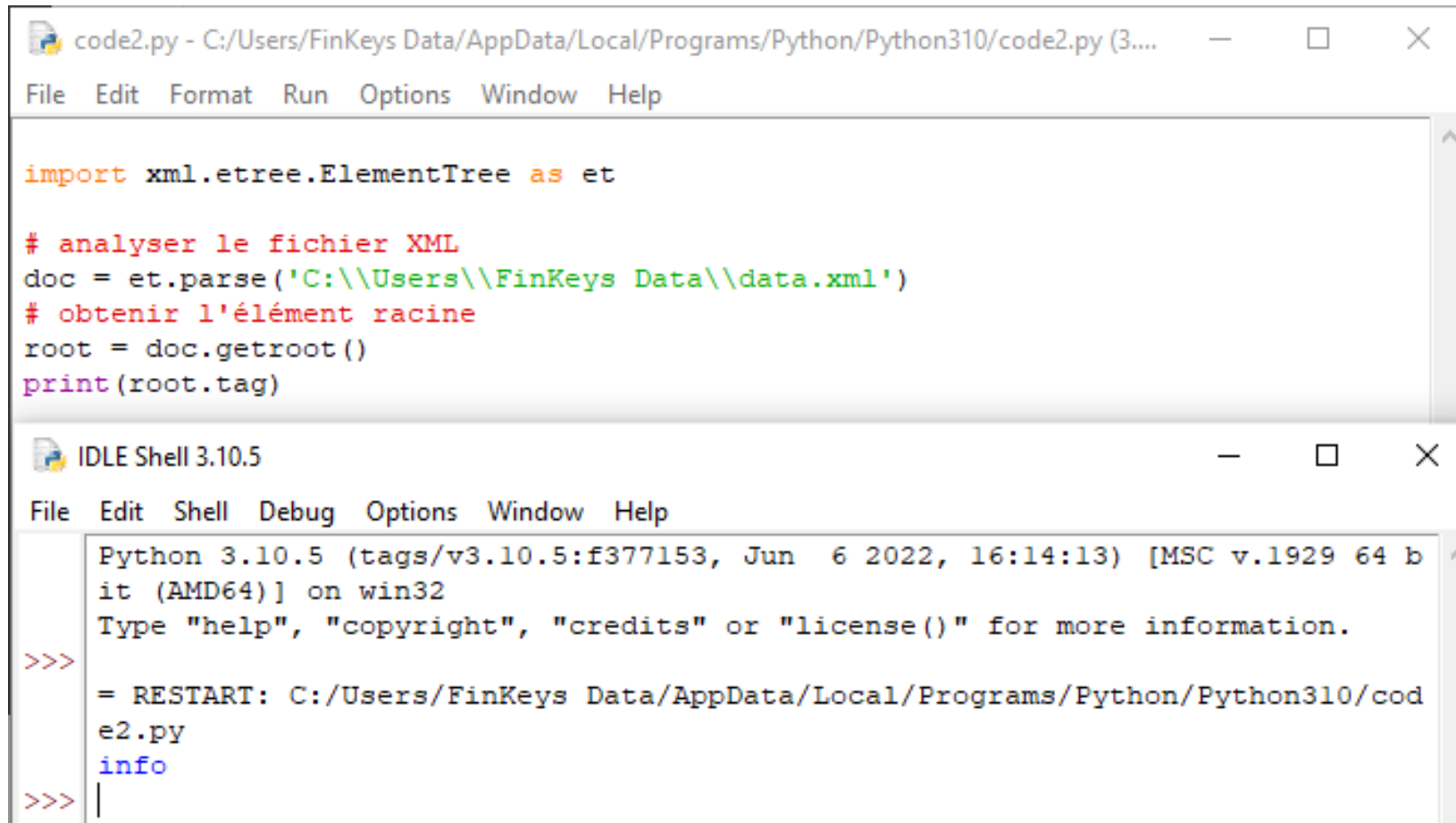
IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/cod
e2.py
Nous avons 4 éléments
item1
item2
item3
item4
```

Récupérer le nom de la balise avec la fonction *getElementsByTagName* et les attributs avec *getAttribute*

# Utilisation xml

## Minidom & ElementTree

*ElementTree* fournit une façon Pythonic pour analyser un fichier XML



```
code2.py - C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/code2.py (3....
File Edit Format Run Options Window Help

import xml.etree.ElementTree as et

# analyser le fichier XML
doc = et.parse('C:\\Users\\FinKeys Data\\data.xml')
# obtenir l'élément racine
root = doc.getroot()
print(root.tag)

IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help

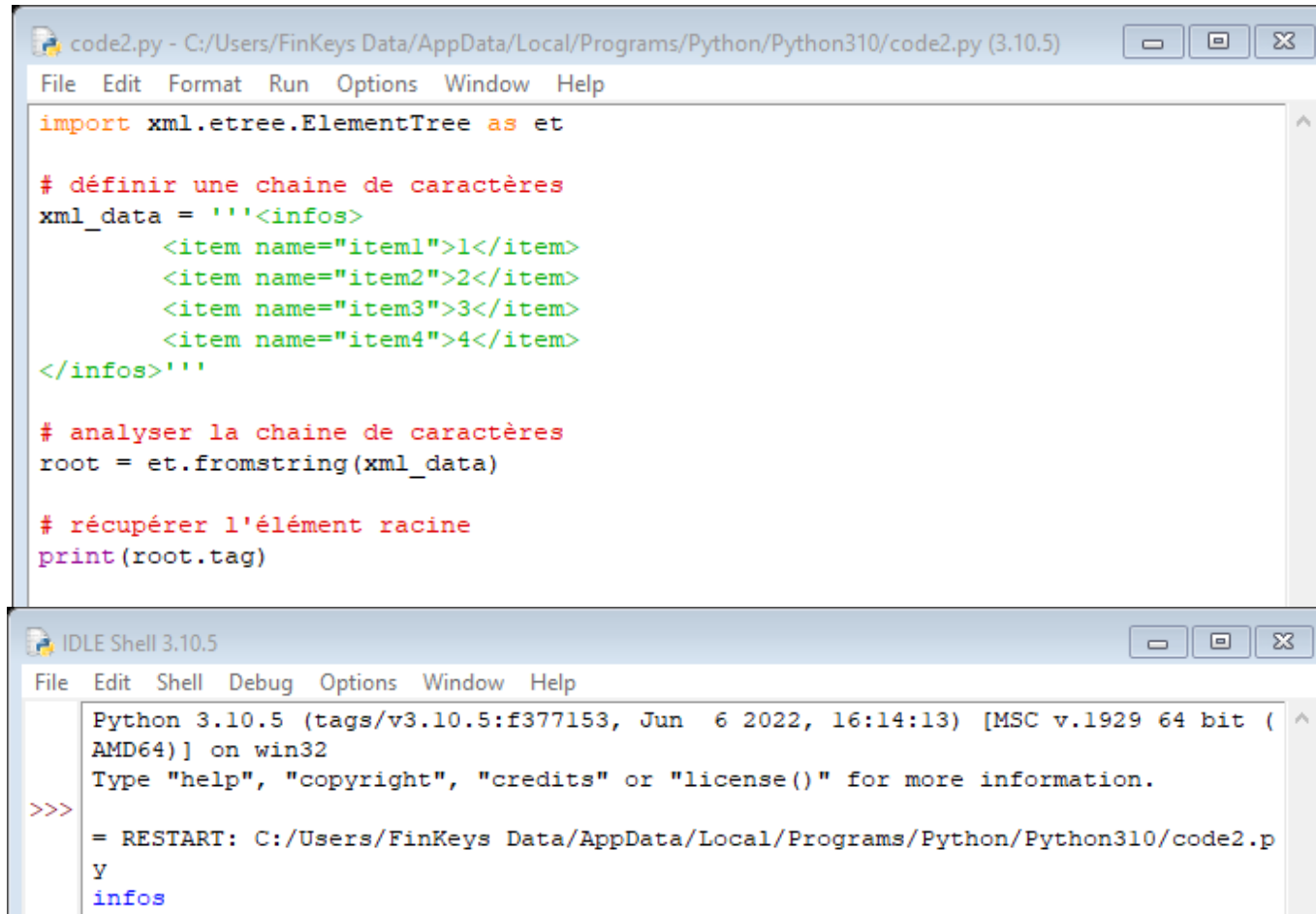
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/cod
e2.py
info
>>> |
```

La méthode *getroot()* est utilisée pour récupérer l'élément racine du fichier XML.

# Utilisation xml

## Minidom & ElementTree

*ElementTree* fournit une façon Pythonic pour analyser un fichier XML



The screenshot shows a Python IDE window titled 'code2.py - C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/code2.py (3.10.5)'. The code defines an XML string, parses it using `et.fromstring()`, and prints the root tag. Below the code editor is an 'IDLE Shell 3.10.5' window showing the execution output.

```
code2.py - C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/code2.py (3.10.5)
File Edit Format Run Options Window Help

import xml.etree.ElementTree as et

# définir une chaîne de caractères
xml_data = '''<infos>
    <item name="item1">1</item>
    <item name="item2">2</item>
    <item name="item3">3</item>
    <item name="item4">4</item>
</infos>'''

# analyser la chaîne de caractères
root = et.fromstring(xml_data)

# récupérer l'élément racine
print(root.tag)
```

```
IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help

Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/code2.p
y
infos
```

La méthode *fromstring()* est utilisée pour récupérer l'élément racine du fichier XML.

# Utilisation xml

## Minidom & ElementTree

Rechercher des éléments à partir de documents XML

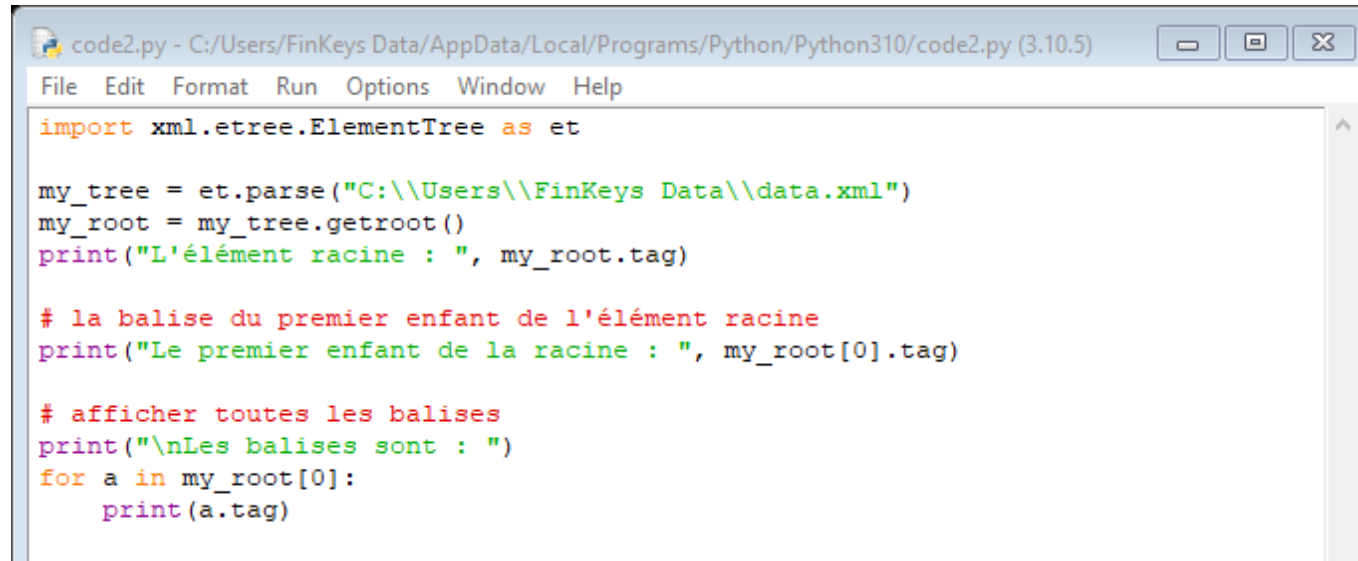
```
data - Bloc-notes
Fichier  Edition  Format  Affichage  Aide
<?xml version="1.0" encoding="UTF-8"?>

<produit>
  <Software >
    <item nom="ABC">Convertisseur PDF</item>
    <prix>100</prix>
    <version>1.2</version>
  </Software>
  <Hardware >
    <item nom="XYZ">Clavier</item>
    <prix>20</prix>
    <garantie>2 ans</garantie>
  </Hardware>
</produit>
```

# Utilisation xml

## Minidom & ElementTree

Rechercher des éléments à partir de documents XML

A screenshot of a Python IDE window titled 'code2.py - C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/code2.py (3.10.5)'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains the following Python code:

```
import xml.etree.ElementTree as et

my_tree = et.parse("C:\\Users\\FinKeys Data\\data.xml")
my_root = my_tree.getroot()
print("L'élément racine : ", my_root.tag)

# la balise du premier enfant de l'élément racine
print("Le premier enfant de la racine : ", my_root[0].tag)

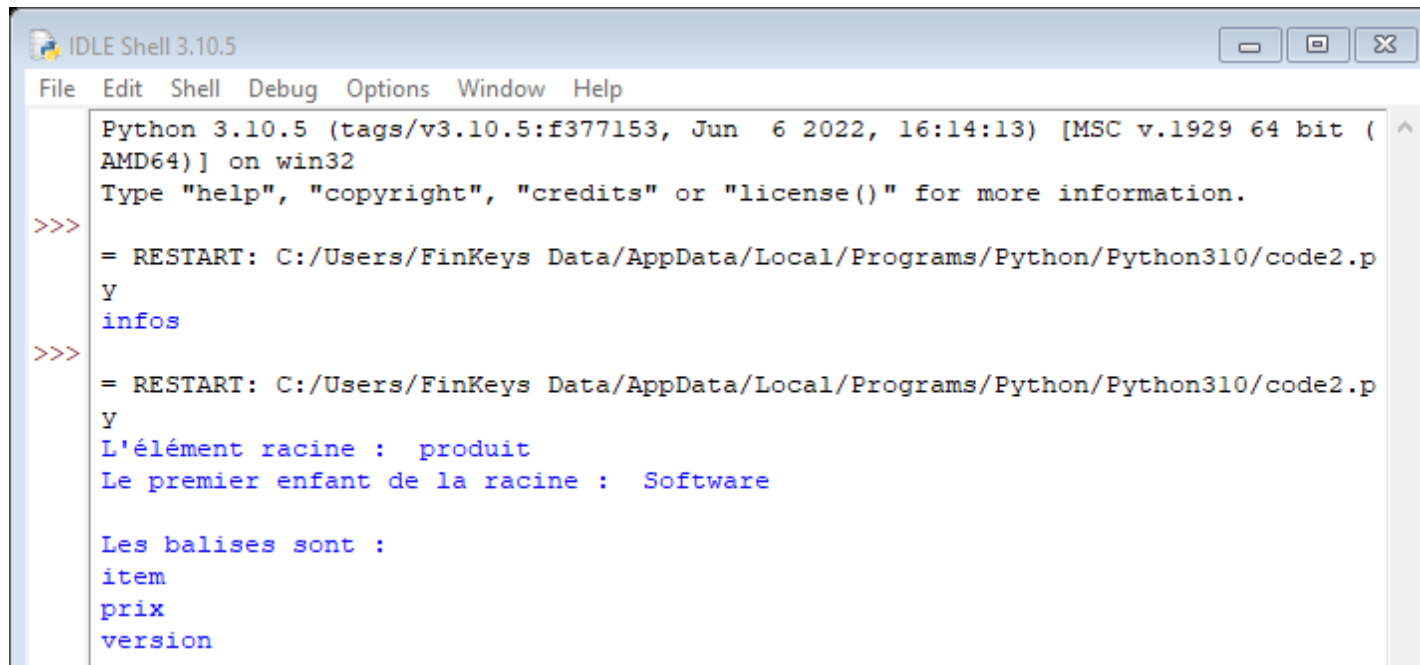
# afficher toutes les balises
print("\nLes balises sont : ")
for a in my_root[0]:
    print(a.tag)
```



# Utilisation xml

## Minidom & ElementTree

Rechercher des éléments à partir de documents XML



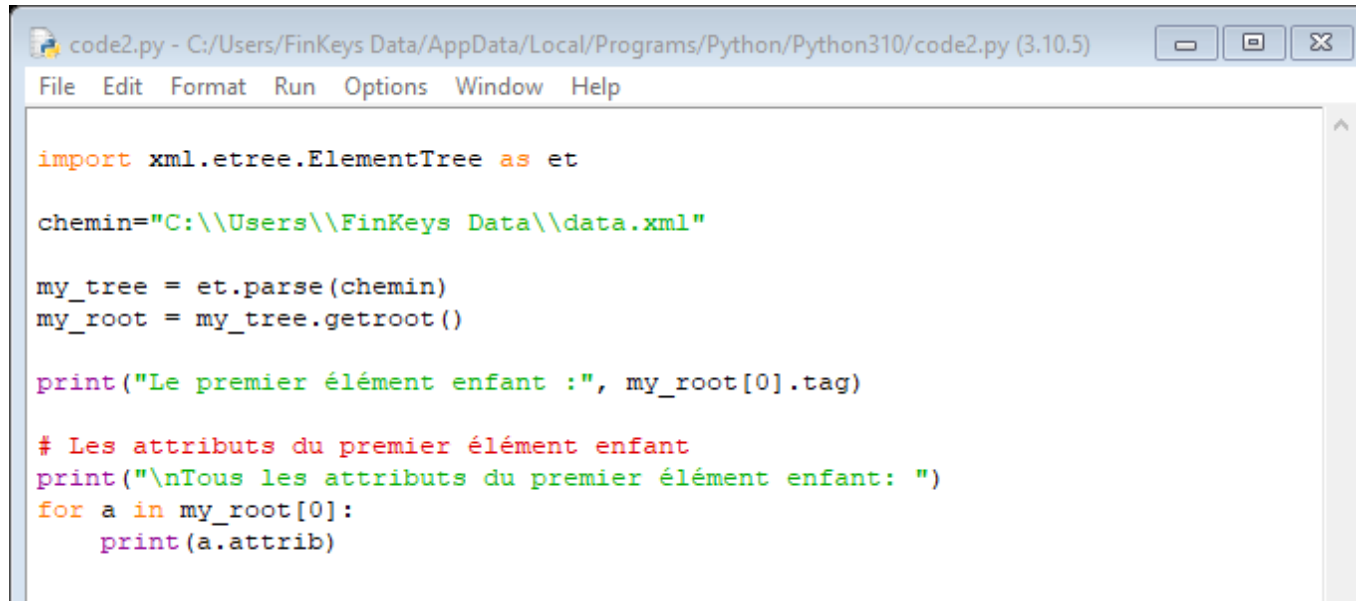
```
IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/code2.p
Y
infos
>>>
= RESTART: C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/code2.p
Y
L'élément racine : produit
Le premier enfant de la racine : Software

Les balises sont :
item
prix
version
```

# Utilisation xml

## Minidom & ElementTree

Rechercher des éléments à partir de documents XML

A screenshot of a Python IDE window titled 'code2.py - C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/code2.py (3.10.5)'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains the following Python code:

```
import xml.etree.ElementTree as et

chemin="C:\\Users\\FinKeys Data\\data.xml"

my_tree = et.parse(chemin)
my_root = my_tree.getroot()

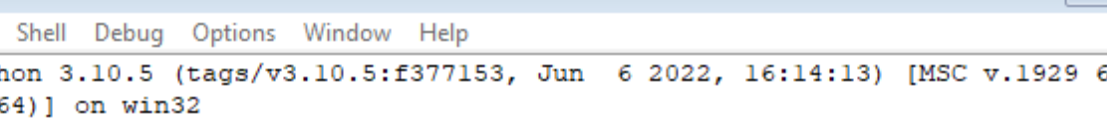
print("Le premier élément enfant :", my_root[0].tag)

# Les attributs du premier élément enfant
print("\nTous les attributs du premier élément enfant: ")
for a in my_root[0]:
    print(a.attrib)
```

# Utilisation xml

# Minidom & ElementTree

## Rechercher des éléments à partir de documents XML



```
IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/code2.p
y
Le premier élément enfant : Software

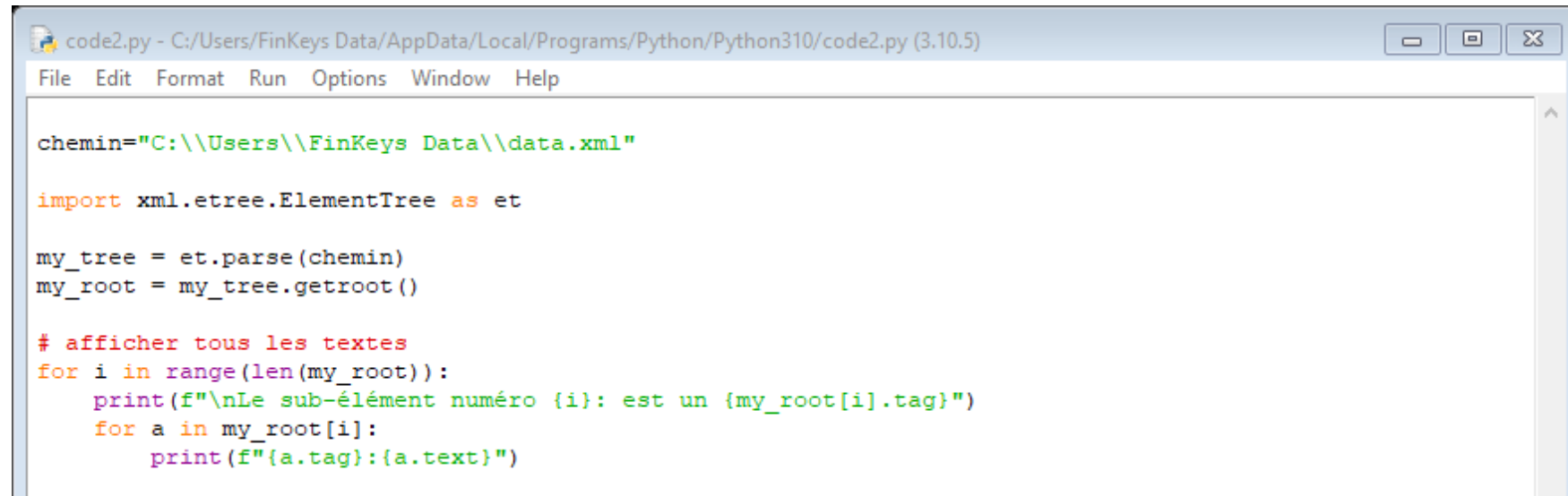
Tous les attributs du premier élément enfant:
{'nom': 'ABC'}
{}
{}

```

# Utilisation xml

## Minidom & ElementTree

Rechercher des éléments à partir de documents XML

A screenshot of a Python IDE window titled 'code2.py - C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/code2.py (3.10.5)'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains the following Python code:

```
chemin="C:\\Users\\FinKeys Data\\data.xml"

import xml.etree.ElementTree as et

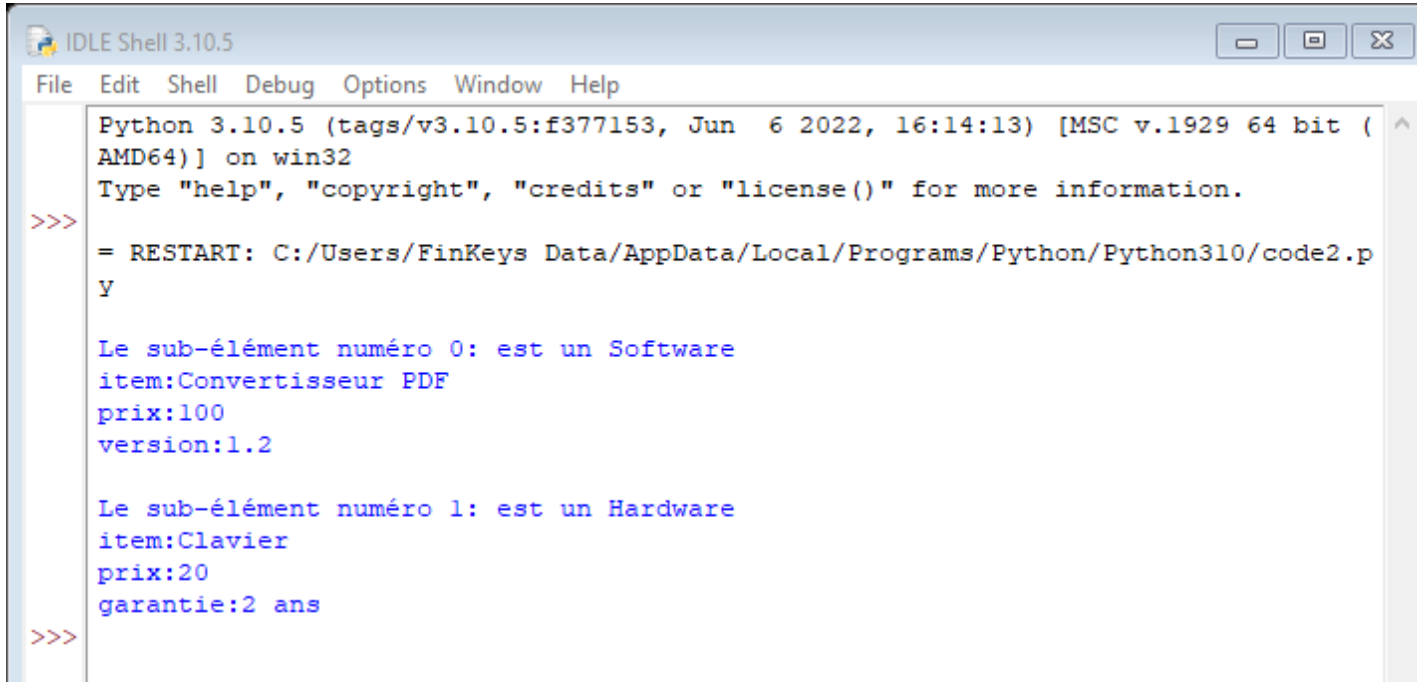
my_tree = et.parse(chemin)
my_root = my_tree.getroot()

# afficher tous les textes
for i in range(len(my_root)):
    print(f"\nLe sub-élément numéro {i}: est un {my_root[i].tag}")
    for a in my_root[i]:
        print(f"{a.tag}:{a.text}")
```

# Utilisation xml

## Minidom & ElementTree

Rechercher des éléments à partir de documents XML



```
IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/FinKeys Data/AppData/Local/Programs/Python/Python310/code2.py

Le sub-élément numéro 0: est un Software
item:Convertisseur PDF
prix:100
version:1.2

Le sub-élément numéro 1: est un Hardware
item:Clavier
prix:20
garantie:2 ans
>>>
```

# Accès aux bases de données relationnelles

## Définition : DB API

Le standard Python pour les interfaces de base de données est l'interface Python DB-API. La plupart des interfaces de base de données Python adhèrent à ce standard.

On peut choisir la base de données qui convient à notre application. L'API de base de données Python prend en charge un large éventail de serveurs de bases de données tels que:

- GadFly
- mSQL
- MySQL
- PostgreSQL
- Microsoft SQL Server 2000
- Informix
- Interbase
- Oracle
- Sybase

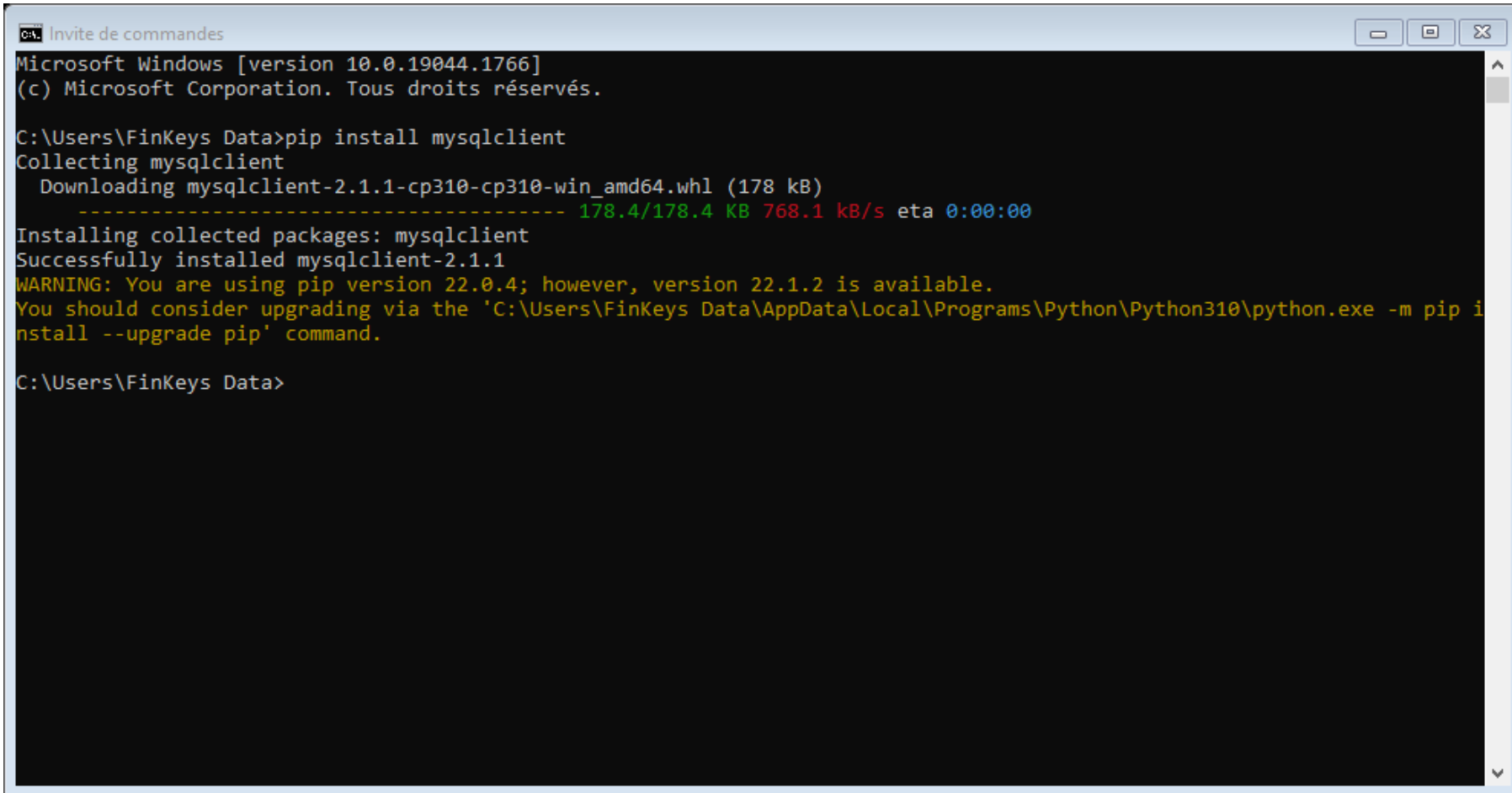
# Accès aux bases de données relationnelles

## MySQLdb

MySQL dB est une interface permettant de se connecter à un serveur de base de données MySQL depuis Python. Elle met en œuvre l'API de base de données Python v2.0 et s'appuie sur l'API C de MySQL.

# Accès aux bases de données relationnelles

## Installer mysqlldb



```

C:\> Invite de commandes

Microsoft Windows [version 10.0.19044.1766]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\FinKeys Data>pip install mysqlclient
Collecting mysqlclient
  Downloading mysqlclient-2.1.1-cp310-cp310-win_amd64.whl (178 kB)
    ----- 178.4/178.4 KB 768.1 kB/s eta 0:00:00
Installing collected packages: mysqlclient
Successfully installed mysqlclient-2.1.1
WARNING: You are using pip version 22.0.4; however, version 22.1.2 is available.
You should consider upgrading via the 'C:\Users\FinKeys Data\AppData\Local\Programs\Python\Python310\python.exe -m pip i
nstall --upgrade pip' command.

C:\Users\FinKeys Data>
```

pip install MySQL-python



# Accès aux bases de données relationnelles

## Etapes de connexion

Les scripts qui accèdent à MySQL au travers de DB-API en utilisant MySQLdb réalisent en général les étapes suivantes :

1. Import du module MySQLdb.
2. Ouverture d'une connexion au serveur MySQL.
3. Exécution de requêtes et récupération des résultats.
4. Fermeture de la connexion au serveur.

# Accès aux bases de données relationnelles

## Etapes de connexion

```
*test_module_import.py - C:/Users/FinKeys Data/MODULE TP/test_module_import.py (3.10.5)*
File Edit Format Run Options Window Help
import sys
import MySQLdb

# connexion au serveur MySQL

try:
    conn = MySQLdb.connect (host = "localhost",
                           user = "testuser",
                           passwd = "testpass",
                           db = "test")

except MySQLdb.Error, e:
    print "Erreur %d : %s" % (e.args[0], e.args[1])
    sys.exit (1)
```

Importer les librairies

Connecter à la base de données MySQL

# Accès aux bases de données relationnelles

## Etapes de connexion

```
# crée la table animal et la peuple
try:
    cursor = conn.cursor ()
    cursor.execute ("DROP TABLE IF EXISTS animal")
    cursor.execute ("""
        CREATE TABLE animal
        (
            name      CHAR(40),
            category  CHAR(40)
        )
    """)
    cursor.execute ("""
        INSERT INTO animal (name, category)
        VALUES
            ('serpent', 'reptile'),
            ('grenouille', 'amphibien'),
            ('thon', 'poisson'),
            ('raton laveur', 'mammifère')
    """)
    print "Nombre de lignes insérées : %d" % cursor.rowcount
```

Curseur pour exécuter des requêtes

Exemple des requêtes

# Accès aux bases de données relationnelles

## Etapes de connexion

```
*test_module_import.py - C:/Users/FinKeys Data/MODULE TP/test_module_import.py (3.10.5)*
File Edit Format Run Options Window Help

# boucle de récupération utilisant fetchall()

cursor.execute ("SELECT name, category FROM animal")
rows = cursor.fetchall ()
for row in rows:
    print "%s, %s" % (row[0], row[1])
print "Nombre de lignes renvoyées : %d" % cursor.rowcount

# exécution d'une requête qui change le nom en incluant les données
# littérales dans la chaîne de la requête, puis remplace le nom
# en utilisant les paramètres

cursor.execute ("""
    UPDATE animal SET name = 'tortue'
    WHERE name = 'serpent'
""")
print "Nombre de lignes mises à jour : %d" % cursor.rowcount

cursor.execute ("""
    UPDATE animal SET name = %s
    WHERE name = %s
""", ("serpent", "tortue"))
print "Nombre de lignes mises à jour : %d" % cursor.rowcount
```

Requête Select

Mise à jour

# Accès aux bases de données relationnelles

## Etapes de connexion

```
cursor.close ()  
conn.commit ()  
conn.close ()
```

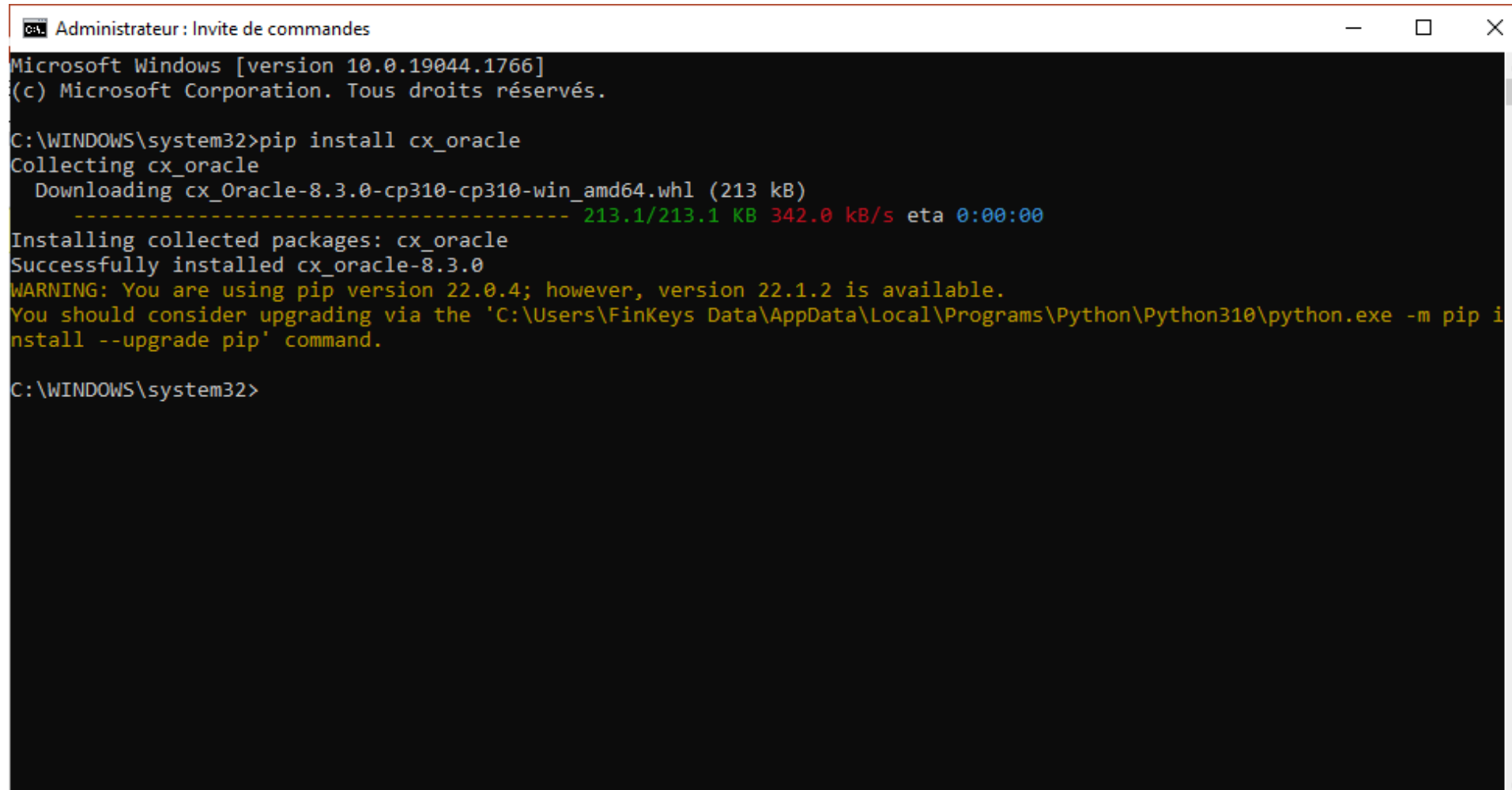
Arrêter le curseur

Enregistrer les modifications

Terminer la connexion

# Accès aux bases de données relationnelles

## Etapes de connexion



```
C:\WINDOWS\system32>pip install cx_oracle
Microsoft Windows [version 10.0.19044.1766]
(c) Microsoft Corporation. Tous droits réservés.

C:\WINDOWS\system32>pip install cx_oracle
Collecting cx_oracle
  Downloading cx_Oracle-8.3.0-cp310-cp310-win_amd64.whl (213 kB)
    ----- 213.1/213.1 KB 342.0 kB/s eta 0:00:00
Installing collected packages: cx_oracle
Successfully installed cx_oracle-8.3.0
WARNING: You are using pip version 22.0.4; however, version 22.1.2 is available.
You should consider upgrading via the 'C:\Users\FinKeys Data\AppData\Local\Programs\Python\Python310\python.exe -m pip i
nstall --upgrade pip' command.

C:\WINDOWS\system32>
```

# Accès aux BD relationnelles

## Etapes de connexion

### Connexion avec une base de données Oracle

```
import cx_Oracle

conn = cx_Oracle.connect("root", "root", "localhost/xe")

cur = conn.cursor()

cur.execute('select * from client')

for line in cur:
    print(line)
```

# Empaquetage & Installation

## Pip

- Il s'agit **d'un système de gestion de paquets** utilisé pour installer et gérer les **paquets/librairies logiciels** écrits en Python.
- On peut alors se demander où sont stockés tous ces paquets/bibliothèques ? Il est évident qu'il doit y avoir un grand "dépôt en ligne" qui stocke tout ce code. La réponse à cette question est le **Python Package Index (PyPI)**.
- PyPI est le dépôt officiel de logiciels tiers pour Python. PyPI hébergeait déjà plus de 95971 paquets !
- Pip utilise PyPI comme source par défaut pour les paquets et leurs dépendances. Ainsi, chaque fois que vous tapez :
- `pip` recherchera ce paquet sur PyPI et s'il est trouvé, il le téléchargera et l'installera sur votre système local.

*`pip install package_name`*



# Empaquetage & Installation

## Étape 1 : Préparer les scripts python

La première étape consiste, bien sûr, à préparer votre programme python (que vous souhaitez publier sur PyPI) !

Il peut s'agir de n'importe quel script python.

# Empaquetage & Installation

## Étape 2 : préparer la structure des répertoires de paquets

C'est l'étape la plus importante. Nous devons maintenant suivre une structure prédéfinie pour le répertoire de notre paquet.

La structure du répertoire doit être la suivante :

```
mygmap/  
  setup.py  
  DESCRIPTION.txt  
  LICENSE.txt  
  README.md  
  .gitignore  
  geo/  
    __init__.py  
    locator.py
```

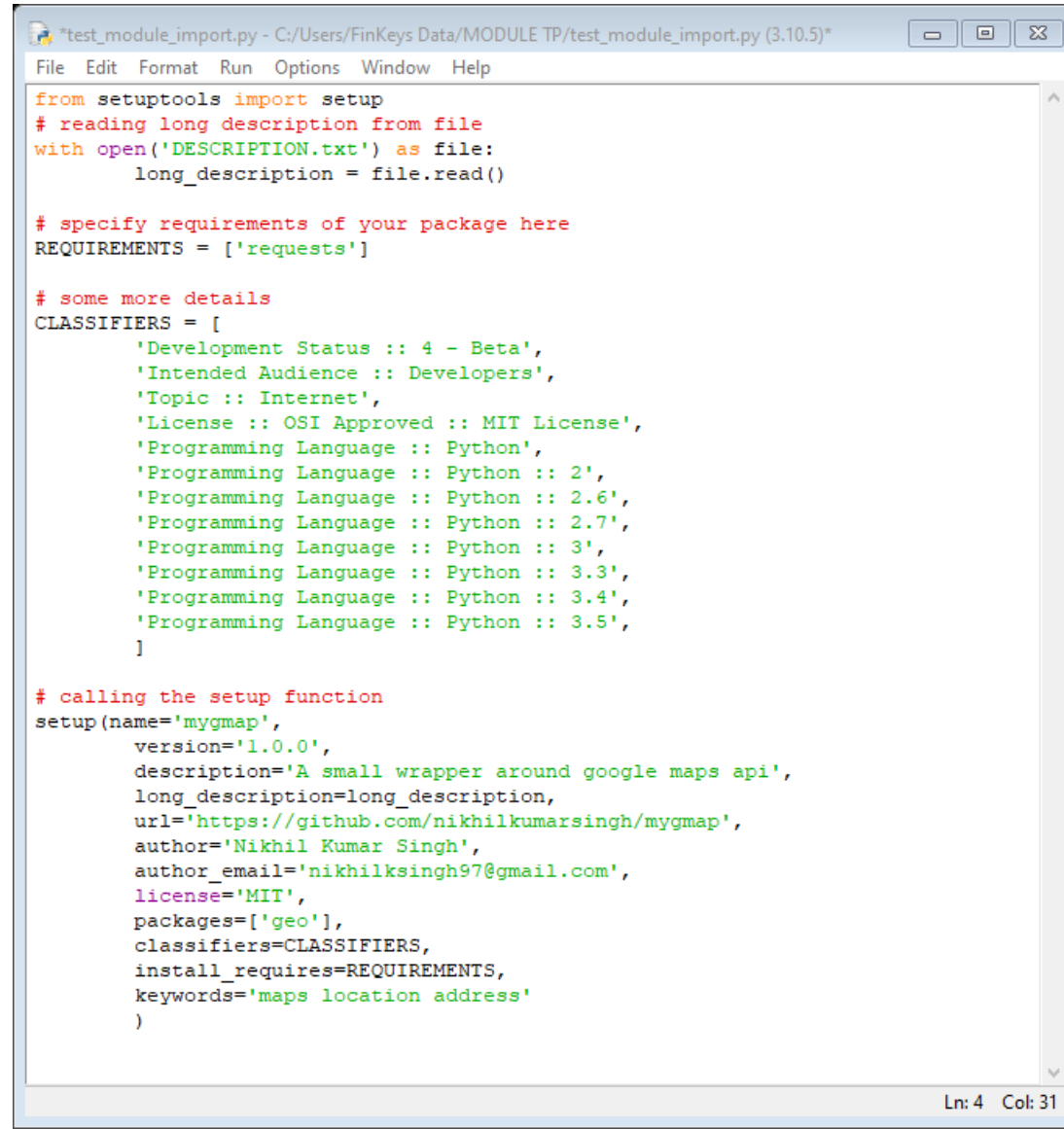
C'est le fichier le plus important. C'est le fichier où les différents aspects de votre projet sont configurés. La principale caractéristique de setup.py est qu'il contient une fonction globale setup(). Les arguments des mots-clés de cette fonction sont la façon dont les détails spécifiques de votre projet sont définis.

Vous devrez installer la bibliothèque setuptools à l'aide de pip :

*pip install setuptools*

# Empaquetage & Installation

## Étape 2 : préparer la structure des répertoires de paquets



```
*test_module_import.py - C:/Users/FinKeys Data/MODULE TP/test_module_import.py (3.10.5)*
File Edit Format Run Options Window Help

from setuptools import setup
# reading long description from file
with open('DESCRIPTION.txt') as file:
    long_description = file.read()

# specify requirements of your package here
REQUIREMENTS = ['requests']

# some more details
CLASSIFIERS = [
    'Development Status :: 4 - Beta',
    'Intended Audience :: Developers',
    'Topic :: Internet',
    'License :: OSI Approved :: MIT License',
    'Programming Language :: Python',
    'Programming Language :: Python :: 2',
    'Programming Language :: Python :: 2.6',
    'Programming Language :: Python :: 2.7',
    'Programming Language :: Python :: 3',
    'Programming Language :: Python :: 3.3',
    'Programming Language :: Python :: 3.4',
    'Programming Language :: Python :: 3.5',
]

# calling the setup function
setup(name='mygmap',
      version='1.0.0',
      description='A small wrapper around google maps api',
      long_description=long_description,
      url='https://github.com/nikhilkumarsingh/mygmap',
      author='Nikhil Kumar Singh',
      author_email='nikhilksingh97@gmail.com',
      license='MIT',
      packages=['geo'],
      classifiers=CLASSIFIERS,
      install_requires=REQUIREMENTS,
      keywords='maps location address'
)
```

Ln: 4 Col: 31

# Empaquetage & Installation

## Étape 2 : préparer la structure des répertoires de paquets

**name** : C'est le nom de votre projet. Votre paquet sera listé sous ce nom sur PyPI.

**version** : C'est une chaîne dans laquelle vous pouvez spécifier la version actuelle de votre projet. La manière dont vous souhaitez définir le schéma de la série de versions est totalement libre (vous pouvez utiliser '1.0' ou '0.1' ou même '0.0.1').

Cette version est affichée sur PyPI pour chaque version si vous publiez votre projet. Chaque fois que vous téléchargez une nouvelle version, vous devrez également modifier cet argument.

**description** : Une courte description du paquet. Vous pouvez utiliser l'argument `long_description` pour écrire des descriptions longues.

**long description** : Nous pouvons utiliser du texte riche pour une meilleure description de nos fichiers. Le format de fichier par défaut est reStructuredText . Vous pouvez jeter un coup d'œil à **DESCRIPTION.txt** pour vous faire une idée de la syntaxe.

**url** : L'URL de la page d'accueil de votre projet. Cela permet aux gens de suivre ou de contribuer plus facilement à votre projet.

**author, author\_email** : Détails sur l'auteur

# Empaquetage & Installation

## Étape 2 : préparer la structure des répertoires de paquets

**license** : spécifie le type de licence que vous utilisez.

**classificateurs** : Il s'agit d'une liste de chaînes dans lesquelles nous pouvons spécifier plus de détails sur notre projet comme son statut de développement, son sujet, sa licence et les versions de python supportées pour votre projet. Vous pouvez voir plus de classificateurs ici.

**install requires** : Il peut être utilisé pour spécifier les bibliothèques tierces dont votre paquet a besoin pour fonctionner. Ces dépendances seront installées par pip lorsque quelqu'un installera votre paquet.

**keywords** : Liste de mots-clés pour décrire votre projet.

**DESCRIPTION.txt** : Ce fichier contient la description longue de notre paquet à afficher sur la page PyPI. Nous utilisons le format de fichier reStructuredText ici. Vérifiez le fichier utilisé dans notre paquet ici.

**LICENSE.txt** : C'est une bonne pratique de définir une licence pour l'utilisation de votre projet. Vous pouvez utiliser l'un des modèles disponibles gratuitement. La plus communément utilisée est la licence MIT.

# Empaquetage & Installation

## Étape 2 : préparer la structure des répertoires de paquets

[README.md](#) : Ce fichier n'a rien à voir avec notre paquet PyPI. Il contient la description à afficher sur la page Github.

[\\_\\_init\\_\\_.py](#) : L'utilisation principale de `__init__.py` est d'initialiser un paquet python.

L'inclusion de ce fichier dans un répertoire indique à l'interpréteur Python que le répertoire doit être traité comme un paquetage Python.

Vous pouvez laisser ce fichier vide.

# Empaquetage & Installation

## Étape 3 : créer un compte sur PyPI et Test PyPI

Test PyPI est juste un site de test où nous allons d'abord télécharger notre code pour voir si tout fonctionne correctement ou non.

Une fois les comptes créés, créez ce fichier `.pypirc` dans le répertoire personnel de votre système et entrez les détails du compte.

```
[distutils]
index-servers =
    pypi
    pypitest

[pypi]
repository=https://pypi.python.org/pypi
username= your_username
password= your_password

[pypitest]
repository=https://testpypi.python.org/pypi
username= your_username
password= your_password
```

Note : Si vous êtes sur un système Windows, tapez simplement `echo %USERPROFILE%` dans l'invite de commande pour connaître le répertoire personnel de votre PC. Placez-y le fichier `.pypirc`.

# Emballage & Installation

## Étape 4 : Télécharger le paquet

Enfin, nous sommes prêts à télécharger notre paquet sur PyPI !

Tout d'abord, nous allons vérifier si notre paquet s'installe correctement sur Test PyPI. Ouvrez une invite de commande/terminal dans le répertoire racine de votre paquet. Exécutez ceci dans le terminal :

```
python setup.py register -r pypitest
```

Ceci va tenter d'enregistrer votre paquet sur le serveur de test PyPI, juste pour s'assurer que vous avez tout configuré correctement.



# Empaquetage & Installation

## Étape 4 : Télécharger le paquet

Maintenant, exécutez ceci :

```
python setup.py sdist upload -r pypitest
```

Vous ne devriez pas obtenir d'erreur, et vous devriez également être en mesure de voir votre bibliothèque dans le dépôt PyPI de test.

Une fois que vous avez téléchargé avec succès vers PyPI Test, effectuez les mêmes étapes mais pointez vers le serveur PyPI live à la place. Pour vous enregistrer sur PyPI, exécutez :

```
python setup.py register -r pypi
```

# Empaquetage & Installation

## Étape 4 : Télécharger le paquet

Ensuite, exécutez :

```
python setup.py sdist upload -r pypi
```

Et voilà, vous avez terminé ! Votre paquet est maintenant disponible publiquement sur PyPI et peut être facilement installé par une simple commande pip !

Le paquet que nous avons créé en utilisant ce tutoriel est disponible [ici](#).

Il suffit de taper dans le terminal

```
pip install votre_nom_de_package
```

pour vérifier si le processus d'installation est terminé avec succès.