PYTHON INITIATION

Présenté par :

Xavier TABUTEAU

Modules

Les modules utiles

Il existe un grand nombre de modules préprogrammés qui sont fournis d'office avec Python. C'est ce qu'on appel la bibliothèque standard. Vous pouvez en trouver d'autres chez divers fournisseurs. Souvent on essaie de regrouper dans un même module des ensembles de fonctions apparentées, que l'on appelle des bibliothèques.

Math

from math import *
sqrt()
sin()
cos()
pi
…

DateTime

from datetime import date
from datetime import time
from datetime import datetime

module_math.py
module_dates.py

Les modules utiles

Modules

Subprocess

```
import subprocess

# run() Exécute un programme.
subprocess.run(["notepad.exe"])

# Exécute une commande en arrière plan
processus = subprocess.Popen(["cmd", "/c", "ping google.com"],
                             stdout=subprocess.PIPE, text=True)

# Affiche le résultat
for ligne in processus.stdout:
    print(ligne.strip())  # Affiche la sortie ligne par ligne
```

module_subprocess.py

Les modules utiles

Modules

Sys

Le module sys en Python fournit des fonctions et variables permettant d'interagir avec l'interpréteur Python. Voici ses principales fonctions et variables :

- Gestion des Arguments de la Ligne de Commande

  sys.argv : Liste des arguments passés au script (le premier élément est le nom du script).
  sys.exit([code]) : Termine le programme avec un code de sortie optionnel.

- Flux d'Entrée et de Sortie

  sys.stdin : Gère l'entrée standard.
  sys.stdout : Gère la sortie standard.
  sys.stderr : Gère les erreurs et les messages d'alerte.

- Informations sur l'Interpréteur Python

  sys.version : Retourne la version de Python sous forme de chaîne.
  sys.platform : Indique le système d'exploitation.
  sys.executable : Chemin de l’exécutable Python utilisé.

Les modules utiles

Modules

Sys

- Gestion des Modules

sys.modules : Dictionnaire des modules chargés.
sys.path : Liste des chemins où Python recherche les modules.
sys.getrecursionlimit() : Obtient la profondeur maximale de récursion.
sys.setrecursionlimit(n) : Définit la profondeur maximale de récursion.

- Gestion de la Mémoire

sys.getsizeof(objet) : Retourne la taille d'un objet en mémoire.
sys.maxsize : Taille maximale d'un entier en Python.
sys.getrefcount(objet) : Retourne le nombre de références à un objet.

Ce module est utile pour interagir avec l'environnement d'exécution et manipuler les entrées/sorties ou les modules dynamiquement.

module_sys.py

Modules

Pathlib

Le module pathlib de Python v3.4 on supérieur fournit une interface orientée objet pour manipuler des chemins de fichiers et de répertoires. Voici ses principales fonctions et classes :

- Création et manipulation de chemins

Path("chemin/vers/fichier") : Crée un objet Path représentant un chemin (relatif ou absolu).
Path.cwd() : Retourne le chemin du répertoire de travail actuel.
Path.home() : Retourne le chemin du répertoire utilisateur.

- Opérations sur les chemins

name : Nom du fichier avec extension.
stem : Nom du fichier sans extension.
suffix : Extension du fichier.
parent : Répertoire parent du fichier/dossier.
parts : Renvoie les différentes parties du chemin sous forme de tuple.

- Navigation et vérifications

exists() : Vérifie si le chemin existe.
is_file() : Vérifie si le chemin est un fichier.
is_dir() : Vérifie si le chemin est un répertoire.

Modules

Les modules utiles

module_pathlib.py

Pathlib

• Lecture et écriture de fichiers

read_text() : Lit le contenu du fichier sous forme de texte.
read_bytes() : Lit le contenu du fichier en bytes.
write_text("contenu") : Écrit du texte dans le fichier.
write_bytes(b"contenu") : Écrit des bytes dans le fichier.

• Création et suppression

mkdir(parents=True, exist_ok=True) : Crée un répertoire (y compris les parents s'ils n'existent pas).
touch() : Crée un fichier vide.
unlink() : Supprime un fichier.
rmdir() : Supprime un répertoire vide.

• Gestion des fichiers et répertoires

iterdir() : Itère sur les éléments d'un répertoire.
glob("*.txt") : Recherche des fichiers correspondant à un motif.
rename("nouveau_nom") : Renomme le fichier ou le répertoire.
replace("nouveau_chemin") : Déplace le fichier en écrasant s'il existe déjà.

Ce module est particulièrement utile pour remplacer os.path avec une approche plus intuitive et plus puissante.

Les modules utiles

Modules

os

Le module os en Python permet d'interagir avec le système d'exploitation. Voici ses principales fonctions :

- Gestion des fichiers et répertoires

getcwd() : Retourne le répertoire de travail actuel.
chdir(path) : Change le répertoire de travail.
listdir(path) : Liste les fichiers et dossiers d’un répertoire.
mkdir(path) : Crée un dossier.
makedirs(path) : Crée un dossier et ses parents si nécessaires.
remove(path) : Supprime un fichier.
rmdir(path) : Supprime un dossier vide.
removedirs(path) : Supprime un dossier et ses parents s’ils sont vides.

- Gestion des processus

system(command) : Exécute une commande système.
popen(command) : Exécute une commande et retourne son résultat.
getpid() : Retourne l’ID du processus actuel.
getppid() : Retourne l’ID du processus parent.

Modules

Les modules utiles

os

• Informations système

name : Donne le nom du système d’exploitation ('posix', 'nt', etc.).
uname() : Retourne des informations sur le système (uniquement sous Unix).
environ : Dictionnaire contenant les variables d’environnement.
getlogin() : Retourne le nom de l’utilisateur connecté.

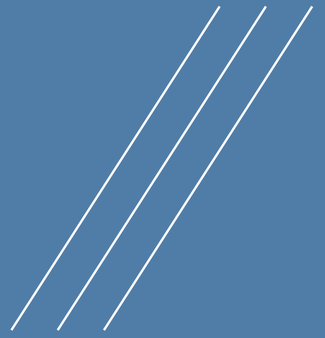
• Manipulation des chemins

path.join(path1, path2) : Construit un chemin valide.
path.exists(path) : Vérifie si un chemin existe.
path.isfile(path) : Vérifie si un chemin est un fichier.
path.isdir(path) : Vérifie si un chemin est un répertoire.
path.abspath(path) : Donne le chemin absolu d’un fichier.
path.basename(path) : Retourne le nom du fichier d’un chemin.
path.dirname(path) : Retourne le dossier d’un chemin.

module_os.py

Ce module est souvent utilisé avec shutil pour des manipulations avancées de fichiers et répertoires.

PYTHON INITIATION



Présenté par

Xavier TABUTEAU