 *Pypi est un site qui propose un catalogue exhaustif des modules créés par la communauté Python afin d'étendre ses fonctionnalités.*

Définition de Module

Un **module** en Python est un fichier contenant des déclarations et des définitions de fonctions, de classes, de variables et d'instructions. Ces éléments peuvent être réutilisés dans d'autres programmes Python. Les modules sont une manière efficace d'organiser et de structurer le code pour le rendre plus lisible et réutilisable.

Installer un package avec pip

Cf https://pypistats.org/top

Pour installer un package avec pip (Python Package Index) sur le terminal, il suffit de rentrer la commande suivante.

```
pip install nom_du_package
```

Pour obtenir la liste des modules installés :

```
pip list
```

Désinstaller un package

```
pip uninstall nom_du_package
```

Les Différents Types de Modules

Il existe deux types de modules en Python :

1. **Modules intégrés (built-in modules)** : Ce sont des modules intégrés dans Python et disponibles par défaut. Par exemple, le module `math` pour les opérations mathématiques.
2. **Modules personnalisés (user-defined modules)** : Ce sont des modules que vous créez vous-même pour organiser votre code en le divisant en fichiers séparés.

Importer un Module

Pour utiliser un module dans votre code, vous devez l'importer à l'aide de l'instruction `import`. Par exemple, pour importer le module `math`, vous pouvez faire ceci :

```
import math
```

Les Alias d'Import de Module

Vous pouvez importer un module sous un alias (un autre nom) en utilisant le mot-clé `as`. Par exemple, pour importer le module `math` sous l'alias `m`, vous pouvez faire ceci :

```
import math as m
```

Importer Uniquement Certains Éléments d'un Module

Si vous n'avez besoin que de certains éléments d'un module, vous pouvez les importer individuellement. Par exemple, pour importer uniquement la fonction `sqrt` du module `math`, vous pouvez faire ceci :

```
from math import sqrt
```

Obtenir la Liste des Éléments d'un Module

Pour obtenir la liste de tous les éléments d'un module, vous pouvez utiliser la fonction `dir()`. Par exemple, pour afficher la liste des éléments du module `math`, vous pouvez faire ceci :

```
import math
print(dir(math))
```

Présentation de Quelques Modules Courants

- Le module `math` : Fournit des fonctions mathématiques avancées pour effectuer des opérations mathématiques complexes.
- Le module `random` : Permet de générer des nombres aléatoires.
- Le module `re` (pour les expressions régulières) : Permet de travailler avec des expressions régulières pour la recherche et la manipulation de motifs de texte.

Des Exemples avec le Module `math`

```
import math

x = math.factorial(5) # factorielle vu plus tôt
y = math.sqrt(25)  # Racine carrée
z = math.sin(math.pi)  # Sinus

print(x, y, z)
```

Méthode	Description
`sqrt(x)`	Calcule la racine carrée de `x`.
`pow(x, y)`	Calcule `x` élevé à la puissance `y`.
`exp(x)`	Calcule l'exponentielle de `x`.
`log(x)`	Calcule le logarithme naturel de `x`.
`cos(x)`	Calcule le cosinus de `x` (en radians).
`sin(x)`	Calcule le sinus de `x` (en radians).
`tan(x)`	Calcule la tangente de `x` (en radians).

Des Exemples avec le Module `random`

```
import random

num = random.randint(1, 100)  # Génère un nombre aléatoire entre 1 et 100
lst = random.sample(range(1, 50), 5)  # Sélectionne 5 éléments aléatoires dans une liste

print(num, lst)
```

Méthode	Description
`random()`	Génère un nombre aléatoire à virgule flottante entre 0 (inclus) et 1 (exclus).
`randint(a, b)`	Génère un nombre aléatoire entier entre `a` (inclus) et `b` (inclus).
`choice(sequence)`	Sélectionne un élément aléatoire dans une séquence.
`shuffle(sequence)`	Mélange aléatoirement les éléments d'une séquence.
`sample(sequence, k)`	Sélectionne `k` éléments uniques aléatoires dans une séquence sans remplacement.
`random.seed()`	Initialise le générateur de nombres aléatoires avec une valeur de départ.

Présentation du Module `re`

Le module `re` est utilisé pour travailler avec des **expressions régulières** ou **expressions rationnelles**. Les expressions régulières sont des motifs de texte qui permettent de rechercher, de filtrer et de manipuler des chaînes de caractères de manière complexe.

Méthode	Description
`re.search(pattern, string)`	Recherche la première occurrence de `pattern` dans `string`.
`re.match(pattern, string)`	Vérifie si `string` commence par `pattern`.
`re.findall(pattern, string)`	Trouve toutes les occurrences de `pattern` dans `string`.
`re.sub(pattern, repl, string)`	Remplace les occurrences de `pattern` par `repl` dans `string`.
`re.compile(pattern)`	Compile un modèle d'expression régulière pour une utilisation ultérieure.
`re.split(pattern, string)`	Divise `string` en une liste en utilisant `pattern` comme séparateur.

Expressions Régulières

Les expressions régulières sont des motifs de texte qui permettent de décrire des modèles de caractères. Par exemple, l'expression régulière `\d` correspond à un chiffre. Les expressions régulières sont très puissantes pour rechercher et manipuler des chaînes de caractères.

Les expressions régulières utilisent des **métacaractères** pour définir des modèles. Voici quelques métacaractères courants :

Métacaractère	Description
`.`	Correspond à n'importe quel caractère, sauf un saut de ligne.
`*`	Correspond à zéro ou plusieurs occurrences du caractère précédent.
`+`	Correspond à une ou plusieurs occurrences du caractère précédent.
`?`	Correspond à zéro ou une occurrence du caractère précédent.
`\`	Permet d'échapper un métacaractère pour le traiter comme un caractère ordinaire.
`[]`	Permet de spécifier un ensemble de caractères valides.
`|`	Permet de spécifier une alternative.