wrappers

# But du laboratoire

* Découvrir et utiliser de nouveaux modules.
* Initiation aux fonctions et aux modules personnalisés.
* Jeux d’essais.

# Préalables

Créez un dossier nommé **wrappers**.

Dans ce dossier créez 4 fichiers Python :

* **mathematique.py**
* **statistiques.py**
* **manipulation\_string.py**
* **main.py**

# Wrappers

Un *wrapper,* est un terme utilisé en programmation qui signifie de réécrire des modules existants pour les rendre plus conviviaux et/ou leur ajouter des fonctionnalités. C’est ce que vous allez faire avec différents modules.

À la fin du laboratoire toute l’exécution devra être faite **à** **partir du fichier main.py**. À l’intérieur de celui-ci vous devez importer vos 3 autres modules et en démontrer l’utilisation.

Avant de vous lancer dans l’écriture du code, il est important de lire un peu sur les fonctions que vous allez *wrapper* et de savoir les utiliser.

J’ai fait le premier pour vous.

**Attention :** Les meilleures pratiques de programmation sont importantes. Utilisez le *type hinting* et les *doctrings* pour créez des fonctions qui vont rendre votre maman fière de vous.

# Module math

Dans votre fichier **mathematique.py** écrivez 3 nouvelles fonctions pour simplifier l’utilisation des trois fonctions suivantes : ceil(), floor() et prod().

Réponse :

import math

def arrondir\_haut(nombre: float) -> int:

*"""Arrondit au plus haut entier.*

*Paramètre: Un nombre réel.*

*Retourne: L'entier supérieur.*

*Exception: TypeError si le nombre n'est pas valide."""*

if not isinstance(nombre, (int, float)):

raise TypeError("Votre nombre n'est pas valide.")

return math.ceil(nombre)

def arrondir\_bas(nombre: float) -> int:

*"""Arrondit au plus bas entier.*

*Paramètre: Un nombre réel.*

*Retourne: L'entier inférieur.*

*Exception: TypeError si le nombre n'est pas valide."""*

if not isinstance(nombre, (int, float)):

raise TypeError("Votre nombre n'est pas valide.")

return math.floor(nombre)

def calculer\_produit(ls\_nombres: list) -> float:

*"""Multiplie entre eux les nombres d'une liste.*

*Paramètre: Une liste de nombres réels.*

*Retourne: Le produit de tous les nombres.*

*Exception: TypeError si la liste n'est pas valide."""*

for nombre in ls\_nombres:

if type(nombre) not in (float, int):

raise TypeError(f"Votre liste n'est pas valide. {nombre} n'est pas un nombre.")

return math.prod(ls\_nombres)

# Module statistics

Dans votre fichier **statistiques.py** écrivez 2 nouvelles fonctions pour **ajouter de la validation** dans les deux fonctions suivantes : **mean()** et **median()**.

* **calculer\_moyenne()**, va calculer la moyenne et lever une erreur si un élément de la liste de nombre n’est pas valide.
* **calculer\_mediane()**, va calculer la médiane et lever une erreur si un élément de la liste de nombre n’est pas valide.

Entrer votre code Python ici

# Module unidecode

Dans votre fichier **manipulation\_string.py** écrivez 2 nouvelles fonctions :

* **standard\_string()**, qui va remplacer les caractères accentués par leur semblable sans accent, enlever les espaces en début et fin de chaîne puis mettre la chaîne en minuscule.
* **validation\_nom()**, qui va vérifier que les caractères contenus dans la chaîne sont uniquement alphabétique et que les seuls symboles acceptés sont l’espace, l’apostrophe et le tiret.

Entrer votre code Python ici