

Etienne LANGLET.

9h00 - 12h30

14h00 - 17h00

CONNEXIONS :

- Poste de travail distant
- Retour du poste formateur
- Visio.

RESSOURCES PEDAGOGIQUES

- Support de cours (PPT)
- Support de référence (livre numérique)

Support de cours.

<https://github.com/elanglet/python>

Module 1 : Présentation de Python.

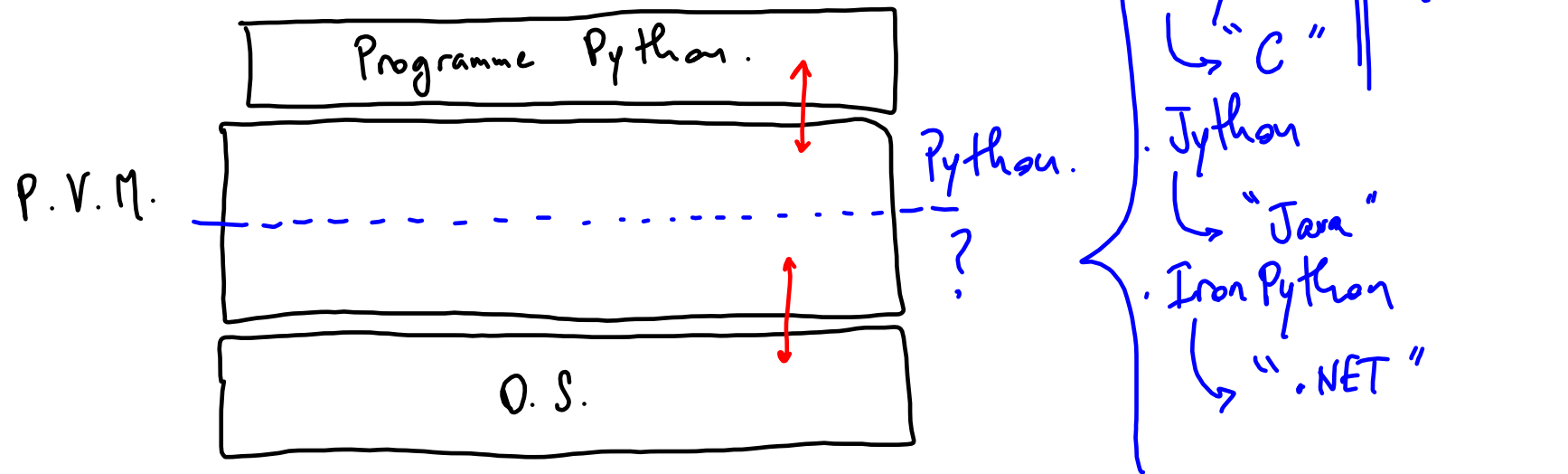
L. N. : Chap. "Présentation de Python"

Site Officiel : www.python.org

Implémentation Python

"Passerelle" entre les programmes et l'O.S.

↳ "Python Virtual Machine"



Lancement d'un programme Python

→ Démarrage de la PVM avec un fichier .py en paramètre
| l'interpréteur



→ Ex. :

python main.py

Module 2 : Mise en place d'une plateforme Python

L.N: Chap. "Installer son env. de travail".

Outils minimum pour Python

- Un éditeur de texte pour écrire les fichiers de code
Source  programme.py
- Une invite de commande / terminal pour lancer l'exécution
 python programme.py

Idéalement :

- Un IDE

Installation de Python

Vigilance !

- Ajout de Python au PATH
- Installation pour tous les utilisateurs
 - ↳ C:\Program Files\...

Par défaut !

[ou non

- ↳ Répertoire utilisateur courant !

Téléchargement & Installation de PyCharm

<http://www.jetbrains.com/pycharm>

Module 3 : Les bases du langage

L.N. : Chap. "Les premiers pas"
"Déclarations"

Exemple de bloc : if / else

⚠ Indentation du code

Bloc "if" }
if a == 1 :
 → _____
 → _____
 → _____
Bloc "else" }
else :
 → _____
 → _____
 → _____

Types de données :

Simple (1 seule valeur)

- . Entiers (int)
- . Réels (float)
- . Chaînes de caractères (str)
- . Booléen (bool)

Evolvés (+ieurs valeurs)

- . Ensembles (tuple)
- . Listes (list)
- . Dictionnaires (dict)

Spécial :

- . Représente l'absence de valeur (None)

Les collections

Ensemble (tuple) ()

Liste (list) []

Identificateur ↘ Paramètres

```
def addition(operande1, operande2):  
    resultat = operande1 + operande2  
    return resultat
```

↑
Retour de la fonction

Structure des collections

Ensembles & listes

\emptyset	"un"
1	"deux"
2	"trois"
3	"quatre"

Dictionnaires

"un"	1
"deux"	2
"trois"	3
"quatre"	4

Les opérateurs.

$x = x + 1$

$x += 1$

~~$x++$~~

Avant de manipuler une donnée, on peut s'assurer que cette donnée existe:

if données is not None:

====

Module 4 : Les fonctions

L.N. : Chap. "Fonctions et Modules"

Les fonctions

Identificateur → Paramètres (Facultatifs)
def addition(operande1, operande2):
 resultat = operande1 + operande2
 return resultat

Signature

↑
Retour de la fonction (Facultatif)

Module 5: Les modules

L.N. : Chap. "Fonctions et Modules"

Module 6 : La programmation Orientée Objet

L.N. : Chap. "Les classes"

Les concepts de la P. O. O.

- Objet
- Classe
- Encapsulation
- Héritage
- Polymorphisme

Standards de
l'O.M.G

"Object Management Group"

L'objet :

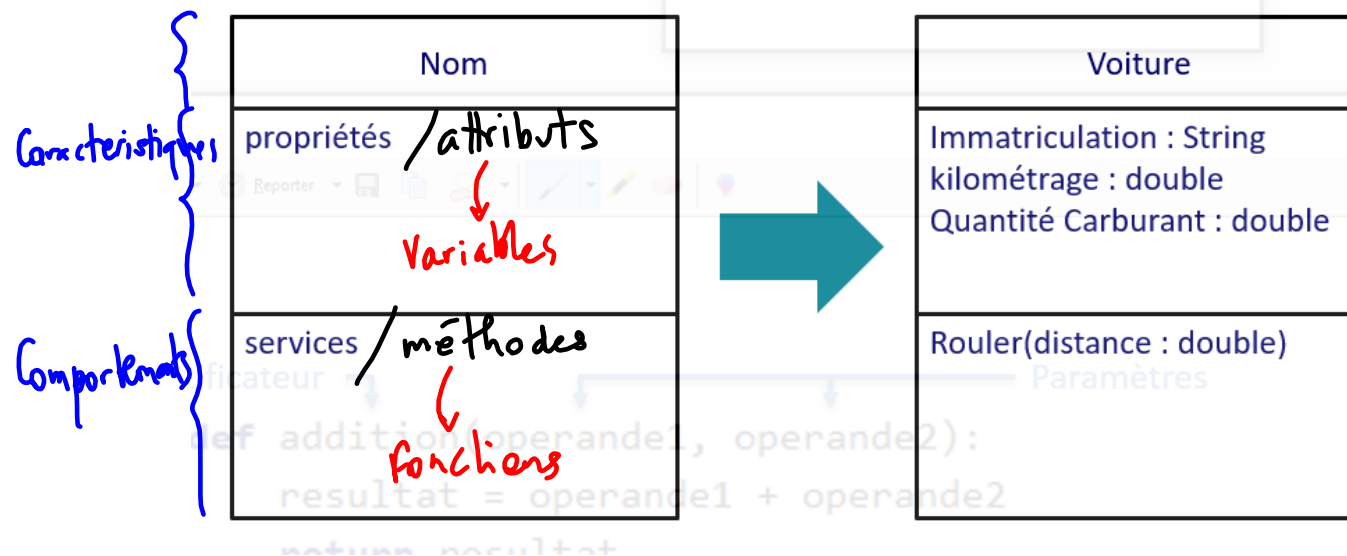
" Une entité identifiable du monde réel ayant des caractéristiques et des comportements ".

→ Faire preuve d'abstraction!

La classe :

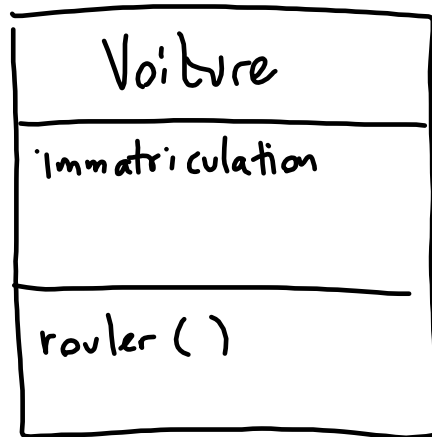
Modèle de construction des objets.

Représentation d'une classe en UML



Relation Classe / Objet

Classe = Modèle



} Attributs

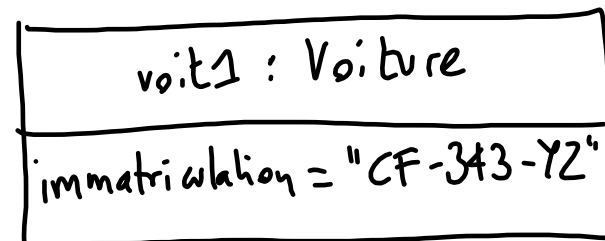
} Methodes

"Instanciation"



Instance

Objet = exemple



Vehicle
<pre>marque = "" modele = "" vitesse = 0 vitesse_max = 0</pre>
<pre>accelerer() freiner()</pre>

Le constructeur

→ Créer et initialiser les objets

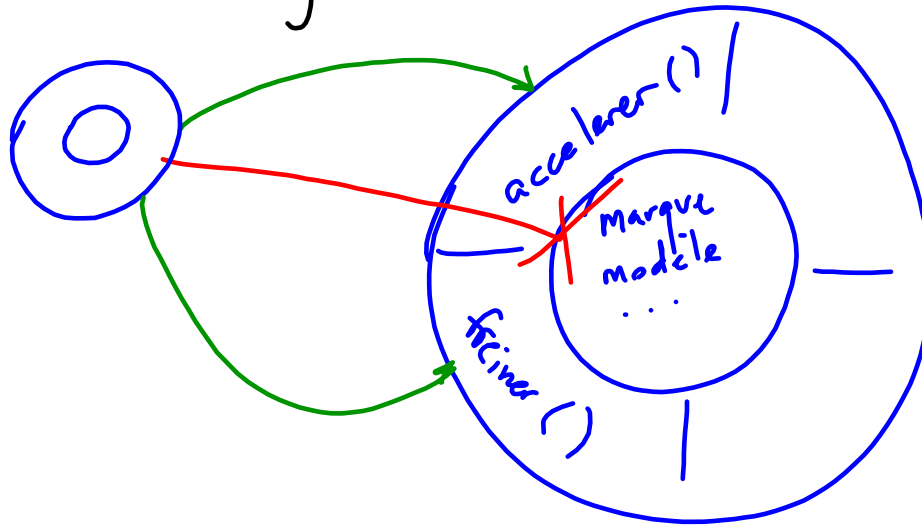
⊗ Dans la classe: `def __init__(self):`

⊗ Au moment de la création de l'objet: `Vehicule()`

L'encapsulation

Objectif: Assurer l'intégrité de l'objet !

- Interdire l'accès direct aux attributs. (Privés)
- L'objet ne sera plus manipulable que par ses méthodes (Publics)



Les accesseurs

→ En lecture (getter)

Pour un attribut `--nom`

↳ `def get_nom(self):`
`return self.--nom`

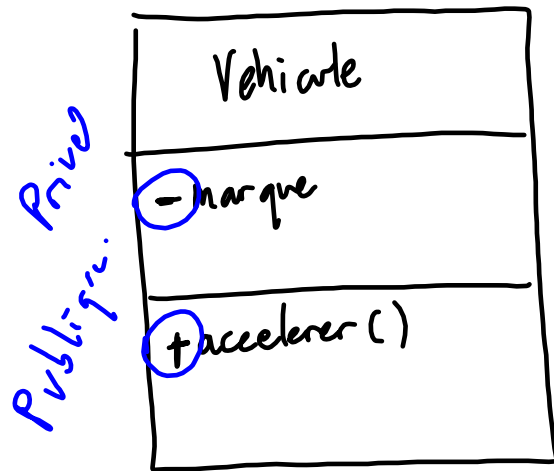
→ En écriture (setter)

↳ `def set_nom(self, nom):`
`# Contrôles sur 'nom'`
`self.--nom = nom`

L'encapsulation : Synthèse

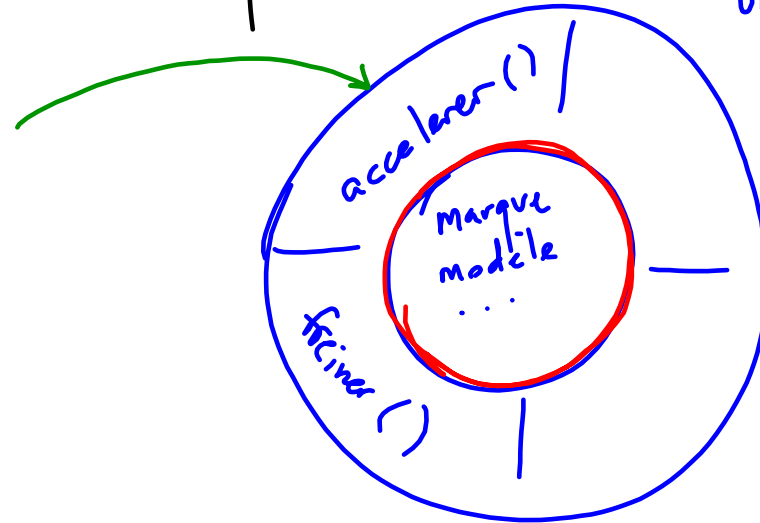
Attributs privés
Méthodes publiques

en UML:



AVANTAGES:

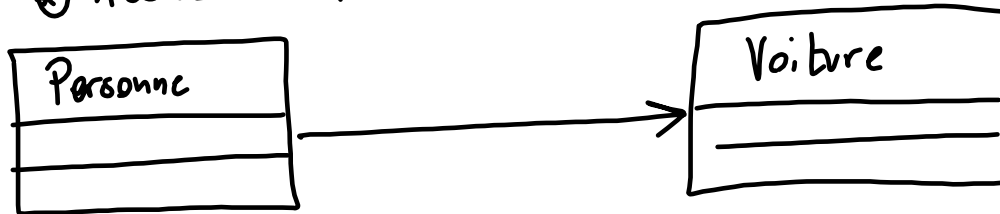
- ⊗ Garantir l'intégrité de l'objet
- ⊗ Simplification de la vue de l'objet d'un point de vue externe : Par les méthodes uniquement!



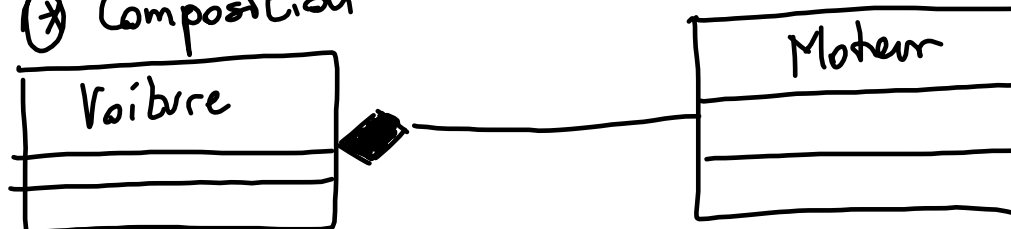
Les collaborations

Représentation UML :

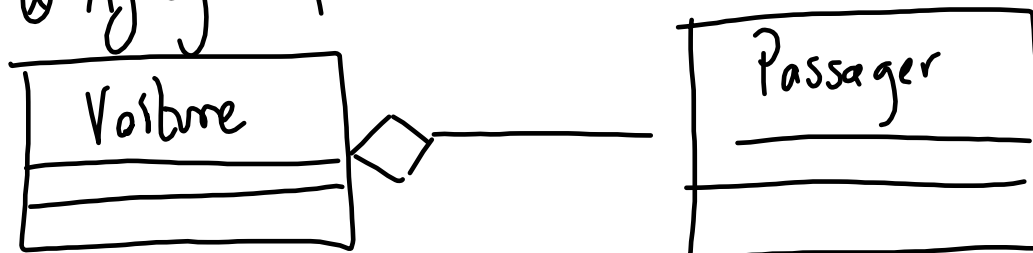
⊗ Association



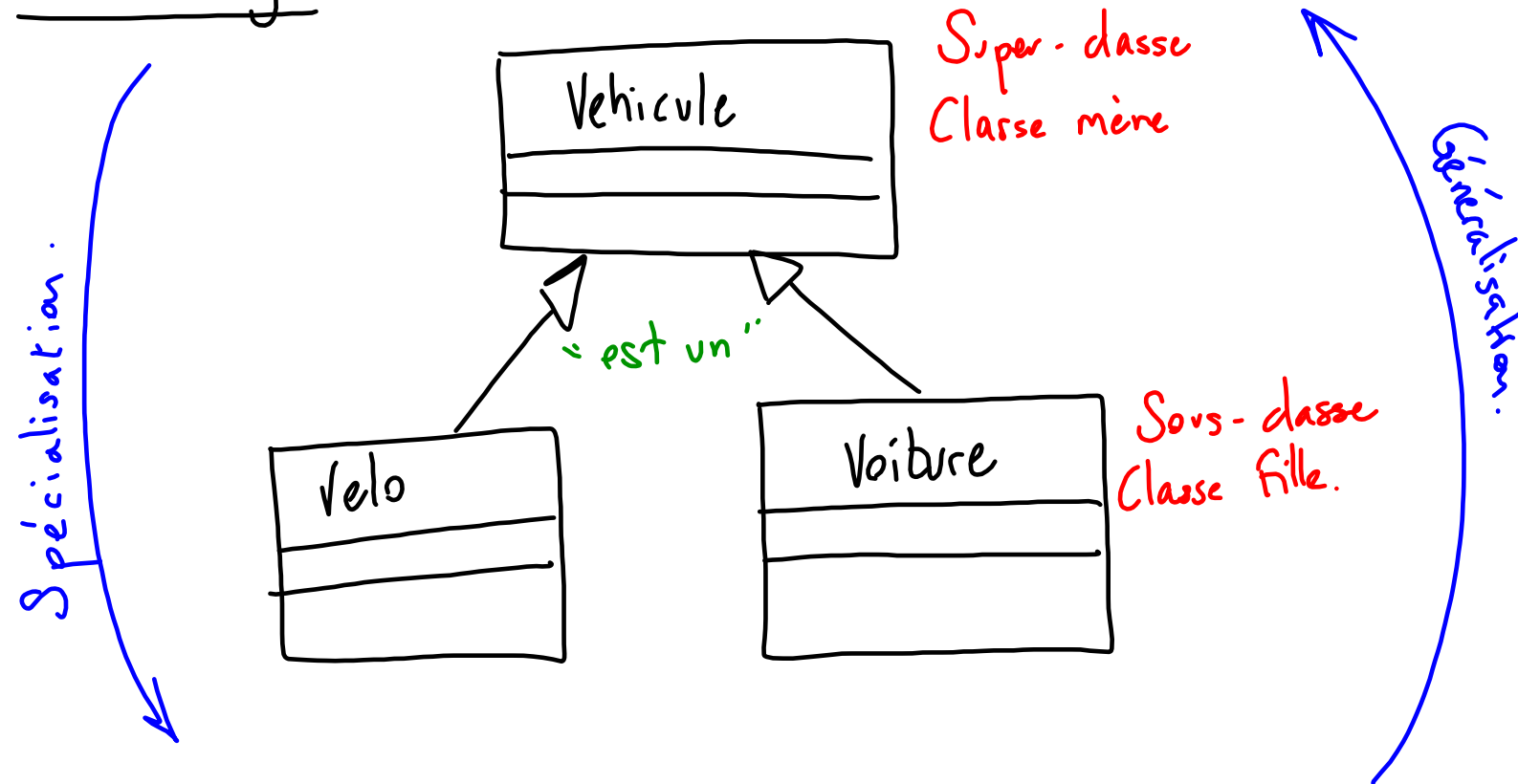
⊗ Composition



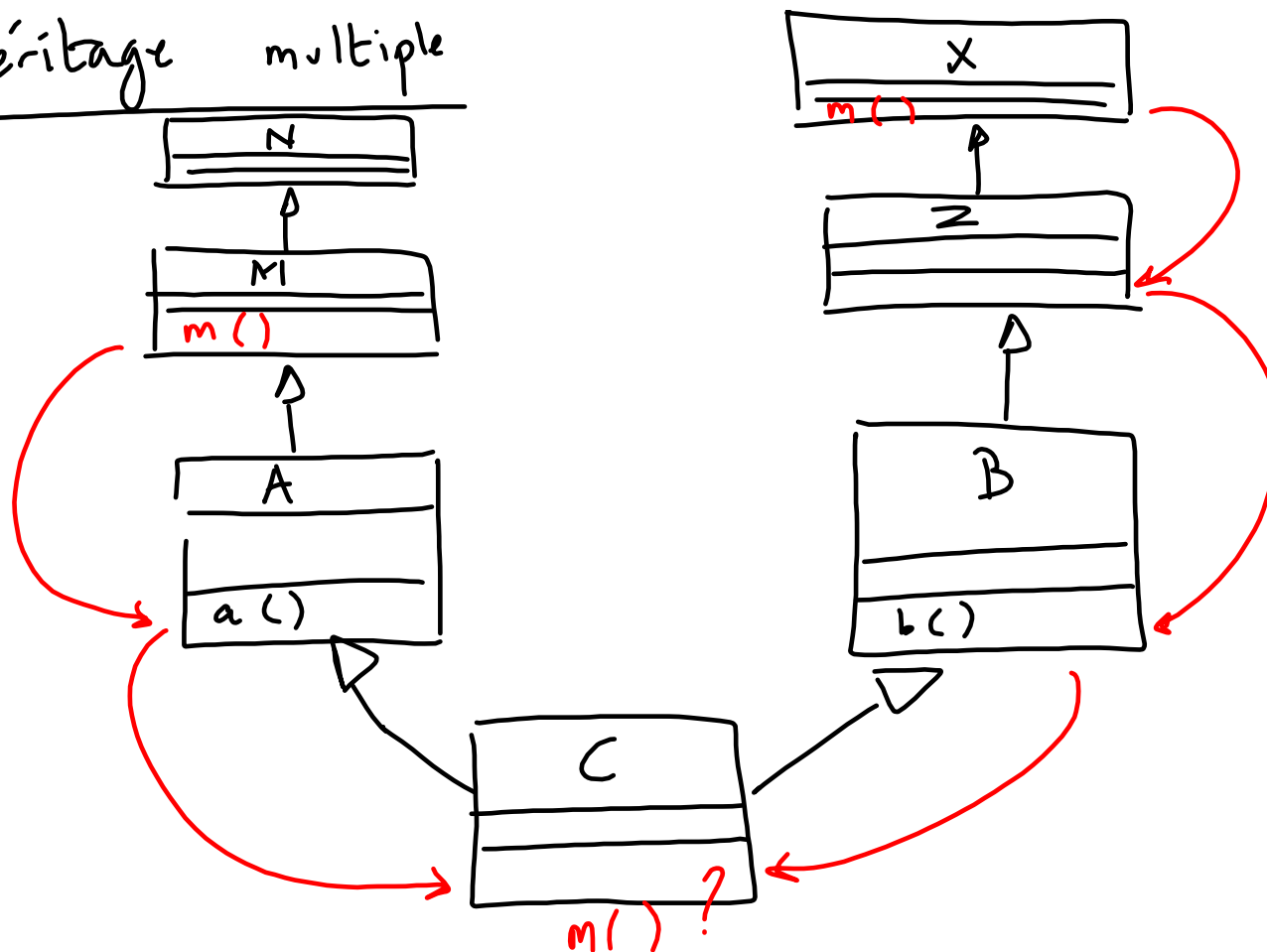
⊗ Agrégation

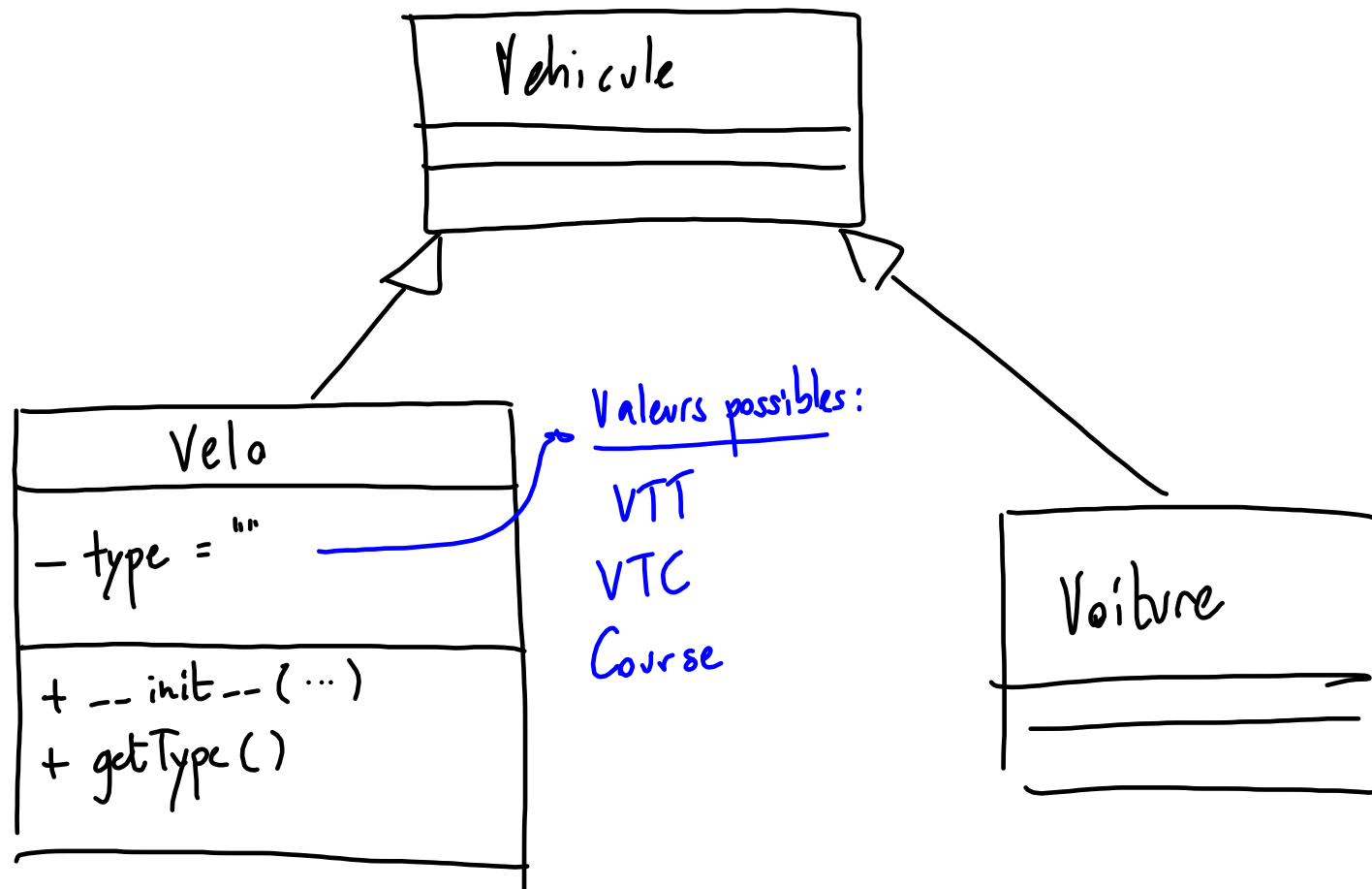


L'héritage



L'héritage multiple



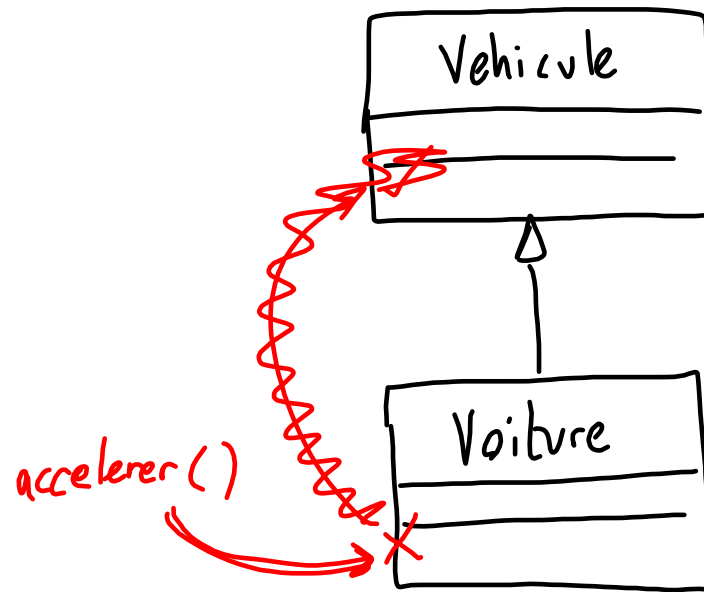


Problème : La voiture accélère sans consommer de carburant !

voit1.accelerer()

REDEFINITION DE METHODE!

↳ Spécialisation parfois
nécessaire dans l'héritage.



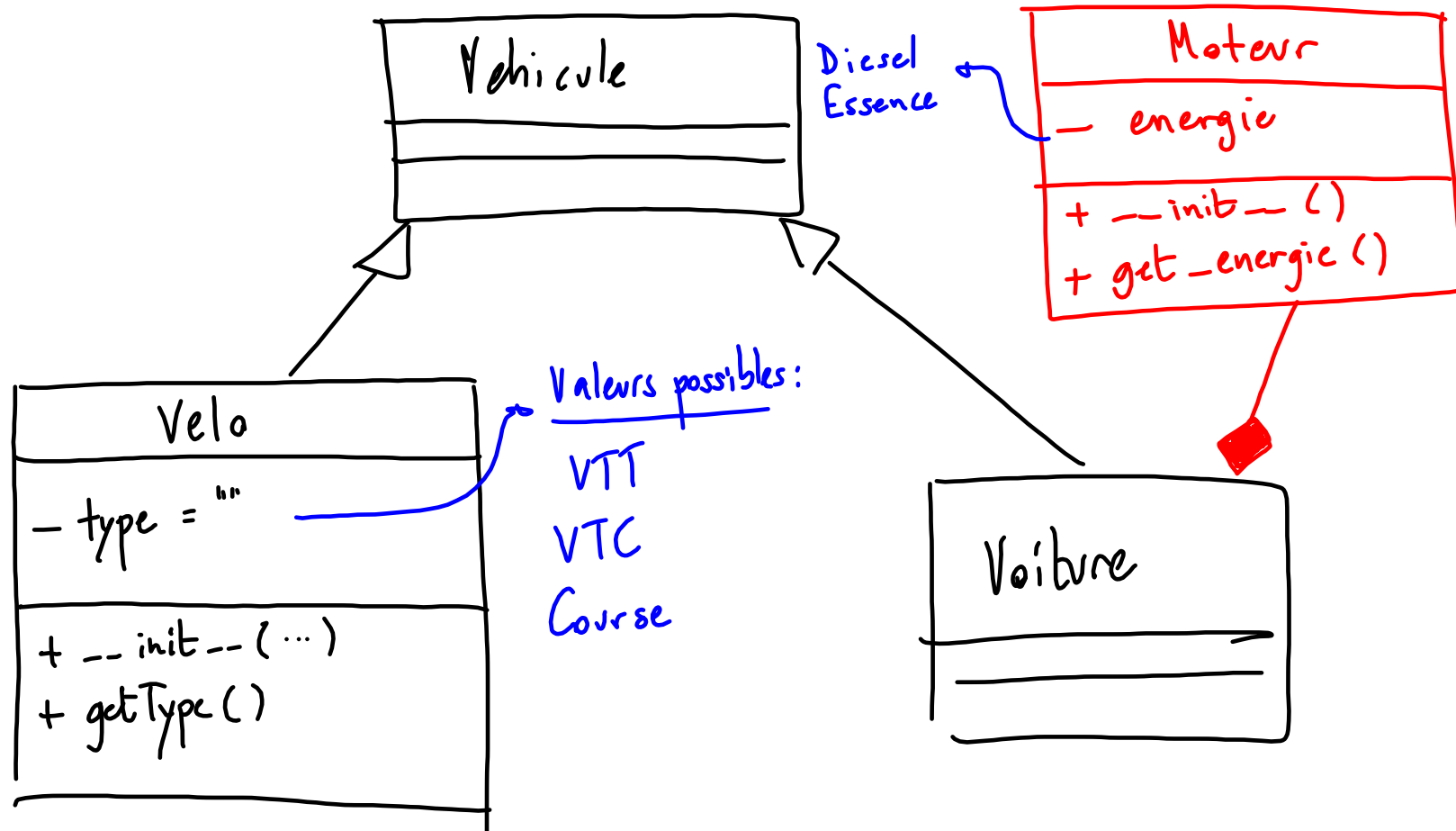
Polymorphisme

Une méthode avec des comportements différents selon la sous-classe !

↳ Redéfinition de méthodes.

Polymorphisme

↳ Forme
↳ Plusieurs.



La composition

```
voit1 = Voiture( . . . . . , "Diesel" )
```

référence au moteur

```
moteur = voit1.get_moteur()
```

Renvoi l'objet Moteur.

Energie

```
energie = moteur.get_energie()
```

ou

```
energie = voit1.get_moteur().get_energie()
```

Module 7 : Concepts Avancés.

L.N. : Chap. "Modèle Objet"

↳ Fonctions spéciales et primitives associées.

Chap. "Algorithmique de base"

↳ Instructions

↳ 7. Gestion des exceptions

Les méthodes spéciales de comparaison

On écrit :

voit1 == voit2

Python exécute :

voit1 . --eq-- (voit2)

On vérifie que 'other'
est bien un objet 'Voiture'

```
def __eq__(self, other):  
    if isinstance(other, Voiture) and self.__immatriculation == other.__immatriculation:  
        return True  
    return False
```

Les exceptions

Generation d'une exception

1 / Création de l'objet d'erreur.

Référence à l'objet d'err. $e = \text{Exception}(\text{"_____"})$

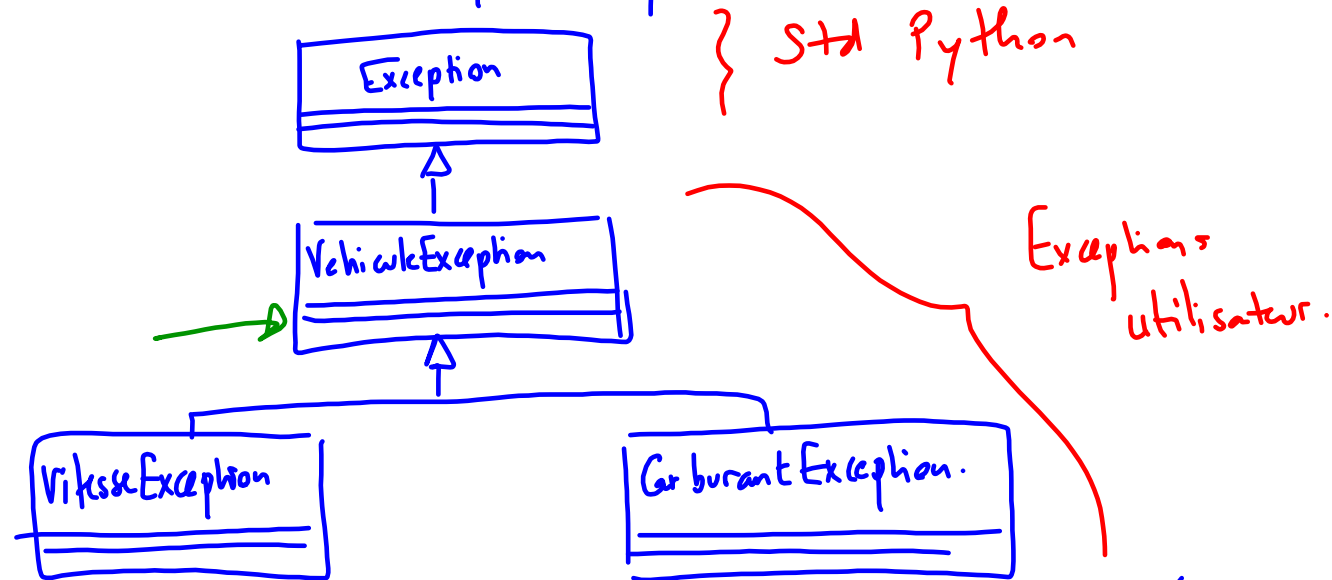
2 / Déclenchement de l'erreur.

$\text{raise } e$

Souvent synthétisé : $\text{raise Exception}(\text{"_____"})$

Mise en oeuvre de la gestion des exceptions dans le projet POO

1/ Création de classes d'exceptions personnalisées.



2/ Déclencher des exceptions pour les conditions non-vérifiées.

3/ Gestion des exceptions dans le module principal.

Gestion des exceptions avec try... except...

Q: Quel périmètre pour le bloc try ?

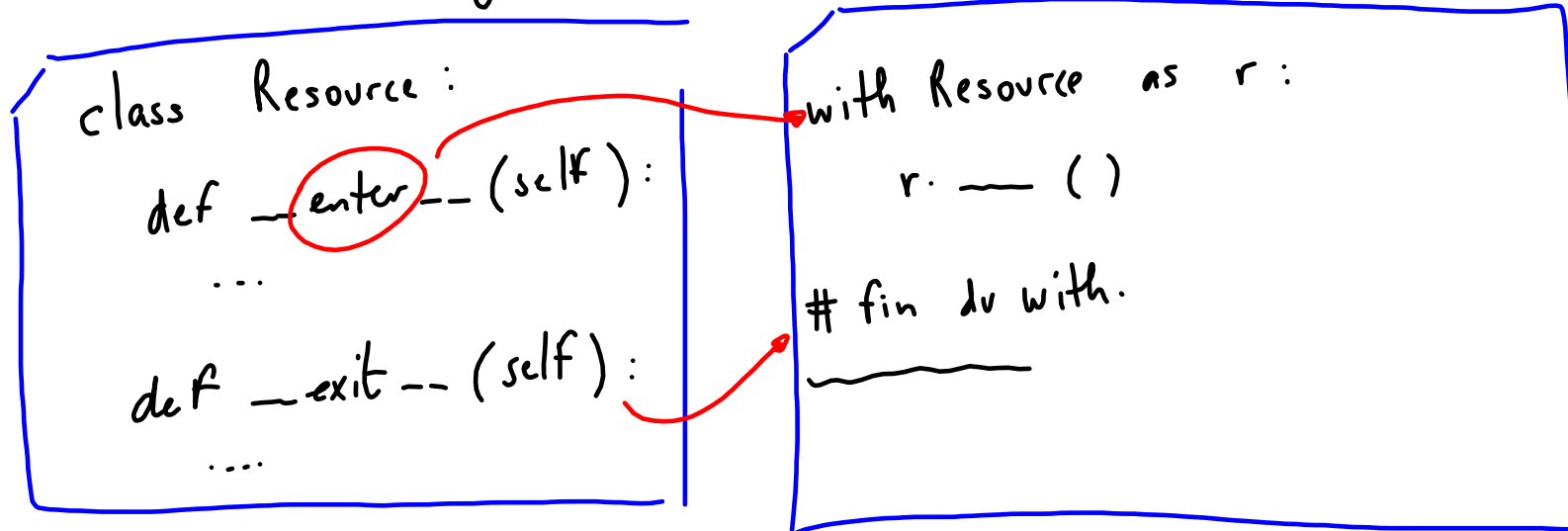
R: Penser au code susceptible de produire une exception et aux conséquences.

↳ L'objet peut-il être manipulé après l'exception ?

Utilisation du with.

With s'utilise avec les ressources !

Resource = Objet structuré comme un "Context Manager"



Module 8 : La bibliothèque standard.

L.N. :: Chap. "Manipulation de données"

Modules étudiés :

- . os
- . subprocess
- . sys
- . pathlib

Exemples de manipulations avec le module os

```

Invite de commandes - python
C:\Users\Administrateur>python
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import os
>>> dir(os)
['DirEntry', 'F_OK', 'MutableMapping', 'O_APPEND', 'O_BINARY', 'O_CREAT', 'O_EXCL', 'O_NOINHERIT', 'O_RANDOM', 'O_RDONLY', 'O_RDWR', 'O_SEQUENTIAL', 'O_SHORT_LIVED', 'O_TEMPORARY', 'O_TEXT', 'O_TRUNC', 'O_WRONLY', 'P_DETACH', 'P_NOWAIT', 'P_NOWAITO', 'P_OVERLAY', 'P_WAIT', 'PathLike', 'R_OK', 'SEEK_CUR', 'SEEK_END', 'SEEK_SET', 'TMP_MAX', 'W_OK', 'X_OK', '_AddedDllDirectory', '_Environ', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', '_check_methods', '_execvpe', '_exists', '_exit', '_fspath', '_get_exports_list', '_putenv', '_unsetenv', '_wrap_close', 'abc', 'abort', 'access', 'add_dll_directory', 'altsep', 'chdir', 'chmod', 'close', 'closerange', 'cpu_count', 'curdir', 'defpath', 'device_encoding', 'devnull', 'dup', 'dup2', 'environ', 'error', 'execl', 'execle', 'execlp', 'execlpe', 'execv', 'execve', 'execvp', 'execvpe', 'extsep', 'fdopen', 'fsdecode', 'fsencode', 'fspath', 'fstat', 'fsync', 'ftruncate', 'get_exec_path', 'get_handle_inheritable', 'get_inheritable', 'get_terminal_size', 'getcwd', 'getcwdb', 'getenv', 'getlogin', 'getpid', 'getppid', 'isatty', 'kill', 'linesep', 'link', 'listdir', 'lseek', 'lstat', 'makedirs', 'mkdir', 'name', 'open', 'pardir', 'path', 'pathsep', 'pipe', 'popen', 'putenv', 'read', 'readlink', 'remove', 'removedirs', 'rename', 'renames', 'replace', 'rmdir', 'scandir', 'sep', 'set_handle_inheritable', 'set_inheritable', 'spawnl', 'spawnle', 'spawnl', 'spawnve', 'st', 'startfile', 'stat', 'stat_result', 'statvfs_result', 'strerror', 'supports_bytes_environ', 'supports_dir_fd', 'supports_effective_ids', 'support_s_fd', 'supports_follow_symlinks', 'symlink', 'sys', 'system', 'terminal_size', 'times', 'times_result', 'truncate', 'umask', 'uname_result', 'unlink', 'urandom', 'utime', 'waitpid', 'walk', 'write']
>>>
>>>
>>> help(os.mkdir)
Help on built-in function mkdir in module nt:

mkdir(path, mode=511, *, dir_fd=None)
    Create a directory.

    If dir_fd is not None, it should be a file descriptor open to a directory,
    and path should be relative; path will then be relative to that directory.
    dir_fd may not be implemented on your platform.
    If it is unavailable, using it will raise a NotImplementedError.

    The mode argument is ignored on Windows.

>>> os.getcwd()
'C:\Users\Administrateur'
>>> os.mkdir('formation_python')
>>> os.chdir('formation_python')
>>> os.getcwd()
'C:\Users\Administrateur\formation_python'
>>>

```

Gestion des chemins avec os

rep1/

└ rep2 /

└ rep3 /

`chemin = "rep1" + os.sep + "rep2" + os.sep + "rep3"`

Exemples de manipulations avec subprocess

```

Invite de commandes - python

C:\Users\Administrateur>python
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import subprocess
>>>
>>> retour = subprocess.call('notepad.exe')
>>> print(retour)
0
>>>
>>> retour, resultat = subprocess.getstatusoutput(['cmd', '/?'])
>>> print(retour)
1
>>> print(resultat)
D,marre une nouvelle instance de l'interpr,teur de commandes de Windows

CMD [/A | /U] [/Q] [/D] [/E:ON | /E:OFF] [/F:ON | /F:OFF] [/V:ON | /V:OFF]
[[/S] [/C | /K] chaîne]

/C Ex,cute la commande donn,e par la chaîne de caractères puis se termine.
/K Ex,cute la commande donn,e par la chaîne de caractères et reste actif.
/S Modifie le traitement de la chaîne après /C ou /K (voir ci-dessous).
/Q Ex,cute (sans interactions) la commande donn,e puis reste actif.
/D D,sactive l'ex,cution d'AutoRun ... partir du Registre (voir ci-dessous).
/A Redirige la sortie de commandes internes vers un canal ou un fichier
  ANSI.
/U Redirige la sortie de commandes internes vers un canal ou un fichier

```

Lecture et ecriture dans un fichier

```
>>>
>>> import os
>>> os.getcwd()
'C:\\Users\\Administrateur'
>>> os.chdir('formation_python')
>>>
>>> lignes = ['Ligne1\n', 'Ligne2\n']
>>> with open('fichier.txt', 'w') as f:
...     f.writelines(lignes)
...
>>>
>>> with open('fichier.txt') as f:
...     lignes = f.readlines()
...
>>> print(lignes)
['Ligne1\n', 'Ligne2\n']
>>>
```

Manipulations de chemins avec pathlib (Python 3.4 +)

```
C:\Users\Administrateur>python
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> from pathlib import Path
>>>
>>> ici = Path()
>>> ici.absolute()
WindowsPath('C:/Users/Administrateur')
>>>
>>> ici.exists()
True
>>> ici.is_dir()
True
>>> ici.is_file()
False
>>>
>>> chemin = ici / 'formation_python'
>>> chemin.absolute()
WindowsPath('C:/Users/Administrateur/formation_python')
>>>
>>> autre_chemin = chemin / 'autre_répertoire' / 'un_dernier_répertoire'
>>>
>>> autre_chemin.absolute()
WindowsPath('C:/Users/Administrateur/formation_python/autre_répertoire/un_dernier_répertoire')
>>>
>>> autre_chemin.exists()
False
>>>
>>> autre_chemin.mkdir(parents=True)
>>> autre_chemin.exists()
True
>>> autre_chemin.is_dir()
True
>>>
```

L'opérateur / est
utilisé comme séparateur
universel de chemin !