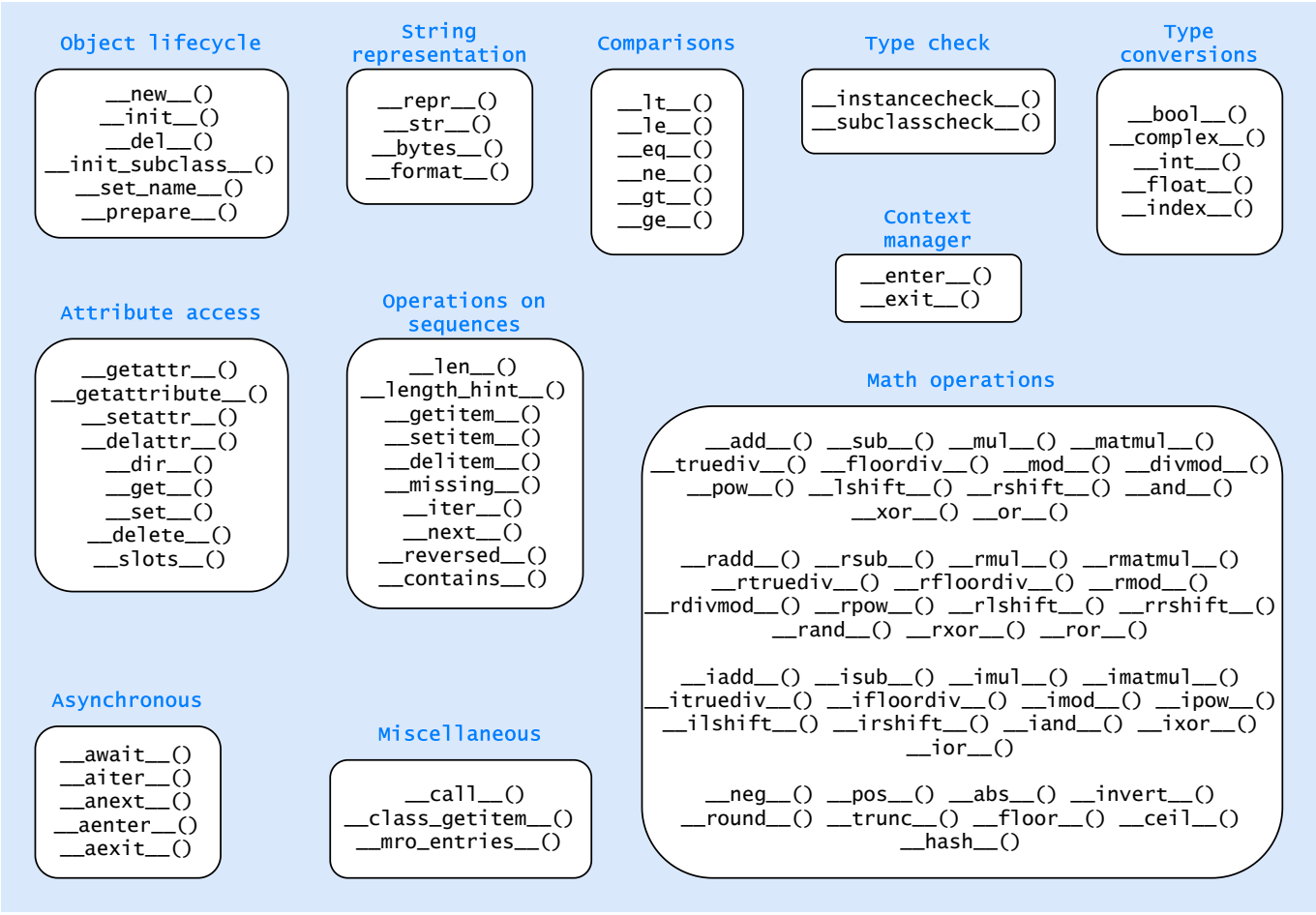

> [!info] Méthodes magiques Liste exhaustive des méthodes magiques en Python ici.

Nom de la Fonction	Exemple Simple	Description

`__init__(self, ...)`	`def __init__(self, param1, param2):`
	`self.param1 = param1`
	`self.param2 = param2`
		Méthode d'initialisation appelée lors de la création d'une instance de la classe.

`__del__(self)`	`def __del__(self):` `print("L'objet a été détruit.")`	Méthode appelée lorsqu'un objet est sur le point d'être détruit (garbage collected).
`__str__(self)`	`def __str__(self):` `return "Ceci est un exemple."`	Retourne une représentation textuelle de l'objet lorsqu'il est converti en chaîne de caractères.
`__repr__(self)`	`def __repr__(self):` `return "Exemple(objet)"`	Retourne une représentation détaillée de l'objet, souvent utilisée à des fins de débogage.
`__len__(self)`	`def __len__(self):` `return self.taille`	Permet de définir la longueur de l'objet lorsque `len(objet)` est appelé.

Nom de la Fonction	Exemple Simple	Description
`__iter__(self)`	`def __iter__(self):` `return iter(self.elements)`	Permet d'itérer sur les éléments de l'objet en utilisant une boucle `for`.

`__next__(self)`	`def __next__(self):`
	`if self.index <`
	`len(self.elements):`
	`result = self.elements[self.index]`
	`self.index += 1`
	`return result`
	`raise StopIteration`
		Utilisée avec `__iter__` pour définir un itérateur sur l'objet.

`__contains__(self, item)`	`def __contains__(self, item):` `return item in self.elements`	Vérifie si un élément est présent dans l'objet en utilisant l'opérateur `in`.
`__eq__(self, other)`	`def __eq__(self, other):` `return self.valeur == other.valeur`	Définit l'égalité (`==`) entre des objets de la classe.
`__ne__(self, other)`	`def __ne__(self, other):` `return self.valeur != other.valeur`	Définit l'inégalité (`!=`) entre des objets de la classe.
`__lt__(self, other)`	`def __lt__(self, other):` `return self.valeur < other.valeur`	Définit l'infériorité stricte (`<`) entre des objets de la classe.
`__le__(self, other)`	`def __le__(self, other):` `return self.valeur <= other.valeur`	Définit l'infériorité ou l'égalité (`<=`) entre des objets de la classe.
`__gt__(self, other)`	`def __gt__(self, other):` `return self.valeur > other.valeur`	Définit la supériorité stricte (`>`) entre des objets de la classe.
`__ge__(self, other)`	`def __ge__(self, other):` `return self.valeur >= other.valeur`	Définit la supériorité ou l'égalité (`>=`) entre des objets de la classe.
`__add__(self, other)`	`def __add__(self, other):` `return self.valeur + other.valeur`	Définit l'addition (`+`) entre des objets de la classe.
`__sub__(self, other)`	`def __sub__(self, other):` `return self.valeur - other.valeur`	Définit la soustraction (`-`) entre des objets de la classe.
`__mul__(self, other)`	`def __mul__(self, other):` `return self.valeur * other.valeur`	Définit la multiplication (`*`) entre des objets de la classe.
`__truediv__(self, other)`	`def __truediv__(self, other):` `return self.valeur / other.valeur`	Définit la division (`/`) entre des objets de la classe (division réelle).
`__floordiv__(self, other)`	`def __floordiv__(self, other):` `return self.valeur // other.valeur`	Définit la division entière (`//`) entre des objets de la classe.

Nom de la Fonction	Exemple Simple	Description
`__mod__(self, other)`	`def __mod__(self, other):` `return self.valeur % other.valeur`	Définit le reste de la division (`%`) entre des objets de la classe.
`__pow__(self, other)`	`def __pow__(self, other):` `return self.valeur ** other.valeur`	Définit l'exponentiation (`**`) entre des objets de la classe.
`__and__(self, other)`	`def __and__(self, other):` `return self.valeur & other.valeur`	Définit l'opération logique "et" bit à bit (`&`) entre des objets de la classe.

`__or__(self, other)`	`def __or__(self, other):`
	`return self.valeur ||`
	`other.valeur`
		Définit l'opération logique "ou" bit à bit () entre des objets de la classe.

`__xor__(self, other)`	`def __xor__(self, other):` `return self.valeur ^ other.valeur`	Définit l'opération logique "ou exclusif" bit à bit (`^`) entre des objets de la classe.
`__invert__(self)`	`def __invert__(self):` `return ~self.valeur`	Définit l'opération de complément à un bit (`~`) sur l'objet.
`__lshift__(self, other)`	`def __lshift__(self, other):` `return self.valeur << other.valeur`	Définit le décalage à gauche (`<<`) sur l'objet.
`__rshift__(self, other)`	`def __rshift__(self, other):` `return self.valeur >> other.valeur`	Définit le décalage à droite (`>>`) sur l'objet.
`__contains__(self, item)`	`def __contains__(self, item):` `return item in self.elements`	Vérifie si un élément est présent dans l'objet en utilisant l'opérateur `in`.

`__enter__(self)`	`def __enter__(self):`
	`print("Entrée dans le contexte")`
	`return self`
		Utilisée dans le contexte d'une instruction `with`, pour définir les actions lors de l'entrée dans le contexte.

`__exit__(self, exc_type, exc_value, traceback)`	`def __exit__(self, exc_type, exc_value, traceback):`
	`print("Sortie du contexte")`
	`return False`
		Utilisée dans le contexte d'une instruction `with`, pour définir les actions lors de la sortie du contexte.

`__getattr__(self, name)`	`def __getattr__(self, name):`
	`if name == 'attribut_special':`
	`return "C'est spécial!"`
	`raise AttributeError`
		Appelée lorsque l'objet tente d'accéder à un attribut inexistant.

Nom de la Fonction	Exemple Simple	Description

`__setattr__(self, name, value)`	`def __setattr__(self, name, value):`
	`if name == 'attribut_special':`
	`raise AttributeError("L'attribut spécial est en lecture seule")`
	`self.__dict__[name] = value`
		Appelée lors de la tentative de définition d'un attribut de l'objet.

`__delattr__(self, name)`	`def __delattr__(self, name):`
	`if name == 'attribut_special':`
	`raise AttributeError("L'attribut spécial ne peut pas être supprimé")`
	`del self.__dict__[name]`
		Appelée lors de la tentative de suppression d'un attribut de l'objet.

`__dir__(self)`	`def __dir__(self):` `return ['attribut1', 'attribut2']`	Utilisée pour personnaliser la liste des attributs accessibles avec la fonction `dir(objet)`.
`__hash__(self)`	`def __hash__(self):` `return hash(self.valeur)`	Retourne la valeur de hachage de l'objet pour son utilisation dans des structures de données telles que les dictionnaires.
`__bool__(self)`	`def __bool__(self):` `return self.valeur != 0`	Permet de définir si un objet est considéré comme vrai ou faux (utilisé par `bool(objet)`).
`__copy__(self)`	`def __copy__(self):` `return Exemple(self.param1, self.param2)`	Définit une copie superficielle de l'objet lorsque la fonction `copy.copy(objet)` est utilisée.
`__deepcopy__(self, memo)`	`def __deepcopy__(self, memo):` `return Exemple(copy.deepcopy(self.param1, memo), copy.deepcopy(self.param2, memo))`	Définit une copie profonde de l'objet lorsque la fonction `copy.deepcopy(objet)` est utilisée.
`__format__(self, format_spec)`	`def __format__(self, format_spec):` `return f"Formaté selon {format_spec}"`	Permet de personnaliser le format d'une chaîne lorsque la fonction `format(objet, format_spec)` est utilisée.

Nom de la Fonction	Exemple Simple	Description

`__getattribute__(self, name)`	`def __getattribute__(self, name):`
	`if name == 'attribut_special':`
	`return "C'est spécial!"`
	`return object.__getattribute__(self, name)`
		Appelée lors de la tentative d'accès à un attribut, même si l'attribut est déjà défini.

`__setstate__(self, state)`	`def __setstate__(self, state):` `self.param1, self.param2 = state`	Utilisée pour personnaliser la désérialisation d'un objet lorsqu'il est récupéré à partir d'une sauvegarde ou d'une sérialisation.
`__reduce__(self)`	`def __reduce__(self):` `return (self.__class__, (self.param1, self.param2))`	Utilisée pour personnaliser la sérialisation d'un objet.
`__reduce_ex__(self, protocol)`	`def __reduce_ex__(self, protocol):` `return self.__reduce__()`	Version améliorée de `__reduce__` prenant en compte le protocole de sérialisation.
`__setstate__(self, state)`	`def __setstate__(self, state):` `self.param1, self.param2 = state`	Utilisée pour personnaliser la désérialisation d'un objet lorsqu'il est récupéré à partir d'une sauvegarde ou d'une sérialisation.
`__sizeof__(self)`	`def __sizeof__(self):` `return object.__sizeof__(self) + sys.getsizeof(self.valeur)`	Retourne la taille de l'objet en octets, utilisée pour l'optimisation de la gestion de la mémoire.
`__getnewargs__(self)`	`def __getnewargs__(self):` `return (self.param1, self.param2)`	Utilisée pour obtenir les arguments nécessaires pour recréer l'objet lors de la désérialisation.
`__setstate__(self, state)`	`def __setstate__(self, state):` `self.param1, self.param2 = state`	Utilisée pour personnaliser la désérialisation d'un objet lorsqu'il est récupéré à partir d'une sauvegarde ou d'une sérialisation.
`__dir__(self)`	`def __dir__(self):` `return ['attribut1', 'attribut2']`	Utilisée pour personnaliser la liste des attributs accessibles avec la fonction `dir(objet)`.