Python
Les Dataclasses

Nicolas Delestre

INSA
ROUEN NORMANDIE

Contexte

La représentation des données

- Très souvent les objets ont besoin de s'échanger des données qui forment un tout mais qui n'ont pas nécessairement des messages à traiter
- Dans les langages tels que C, Pascal, etc. on utilise dans ce cas des **struct**, **record**, etc. pour organiser ces données
- En python, jusqu'à la version 3.7, on utilisait soit :
  - des tuples, mais dans ce cas le code était moins lisible
  - des classes, mais dans ce cas le concepteur devait coder un initialiseur, des transtypages en **str**, etc.

INSA ROUEN NORMANDIE

Module *dataclasses*

- Disponible à partir de la version 3.7 de Python
- Module proposant un décorateur et des fonctions permettant de générer automatiquement du code comme par exemple `__init__` et `__repr__` (cf. la PEP 557) facilitant le développement de classes qui représentent des données

Avantages

- Développement accéléré
- Lisibilité du code

Inconvénient

- Pas d'encapsulation

INSA ROUEN NORMANDIE

Un exemple (tiré de la documentation de python)

Conception

```python
from dataclasses import dataclass

@dataclass
class ElementDInventaire:
    """Classe représentant un élément d'inventaire en stock."""
    nom: str
    prix_unitaire: float
    quantite_en_stock: int = 0

    def cout_total(self) -> float:
        return self.prix_unitaire * self.quantite_en_stock
```

Utilisation

```
>>> from inventaire import ElementDInventaire
>>> ecrous = ElementDInventaire("Ecrou de diametre 15", 0.1,
      100)
>>> ecrous
ElementDInventaire(nom='Ecrou de diametre 15', prix_unitaire
      =0.1, quantite_en_stock=100)
>>> ecrous.nom
'Ecrou de diametre 15'
>>> ecrous.prix_unitaire = 0.2
>>> ecrous.cout_total()
20.0
```

Principaux paramètres du décorateur `dataclass`

À partir de la 3.7

- `init` (valeur par défaut `True`) pour générer la méthode `__init__`
- `repr` (valeur par défaut `True`) pour générer la méthode `__repr__`
- `eq` (valeur par défaut `True`) pour générer la méthode `__eq__`
- `frozen` (valeur par défaut `False`) pour définir les objets immuables
- `unsafe_hash` (valeur par défaut `False`) pour générer la méthode `__hash__` (à n'utiliser que si les objets sont immuables)
- `order` (valeur par défaut `False`) pour générer les méthodes de comparaison (`__lt__`, `__le__`, etc.) en considérant les objets comme des tuples

À partir de la 3.10

- `kw_only` (valeur par défaut `False`) pour obliger à utiliser une initialisation des objets à l'aide d'un passage de paramètre nommé

Quelques fonctions du module dataclasses

```
>>> import dataclasses
>>> dataclasses.asdict(ecrous)
{'nom': 'Ecrou de diametre 15', 'prix_unitaire': 0.1, 'quantite_en_stock':
    100}
>>> dataclasses.astuple(ecrous)
('Ecrou de diametre 15', 0.1, 100)
>>> dataclasses.is_dataclass(ecrous)
True
```

INSA ROUEN NORMANDIE

Fonction field

- Elle permet de paramétrer les valeurs par défaut et le comportement des champs

Exemple tiré de https://www.youtube.com/watch?v=CvQ7e6yUtnw

```python
#!/usr/bin/env python3
import random
import string
from dataclasses import dataclass, field

def id_aleatoire() -> str:
    return "".join(random.choices(string.ascii_uppercase, k=12))

@dataclass
class Personne:
    """Classe représentant une personne"""
    nom: str
    prenom: str
    majeur: bool = True
    emails: list[str] = field(default_factory=list)
    identifiant: str = field(init=False, default_factory=id_aleatoire)
    _champ_recherche: str = field(init=False, repr=False)

    def __post_init__(self):
        self._champ_recherche = f"{self.nom} {self.prenom}"
```

```
>>> from personne import Personne
>>> p = Personne("Delestre","Nicolas")
>>> p
Personne(nom='Delestre', prenom='Nicolas', majeur=True,
      emails=[], identifiant='YZOYLUVTDRSA')
>>>
```

Conclusion

- Les dataclasses en Python, introduites à partir de la version 3.7, simplifient considérablement la création de classes destinées principalement à stocker des données
- Grâce au décorateur `@dataclass`, les méthodes spéciales telles que `__init__`, `__repr__`, et `__eq__` peuvent être générées automatiquement, réduisant ainsi la quantité de code à écrire
- Cela conduit à un développement plus rapide, un code plus lisible, et une meilleure gestion des classes utilisées pour représenter des données
- Le module `dataclasses` offre des fonctionnalités telles que `asdict`, `astuple`, et `field` pour une personnalisation accrue
- Il est essentiel de comprendre les paramètres du décorateur `@dataclass` et la fonction `field` pour tirer le meilleur parti de cette fonctionnalité

INSA ROUEN NORMANDIE