Types abstraits de données

Compétences

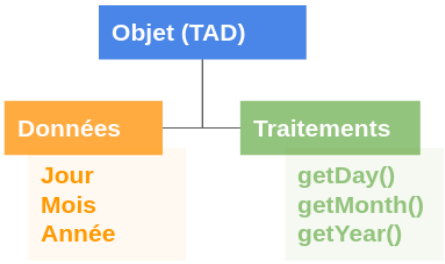
- Implémenter un type abstrait de données
- Effectuer des tests unitaires pour vérifier le bon comportement de votre implémentation.

Introduction

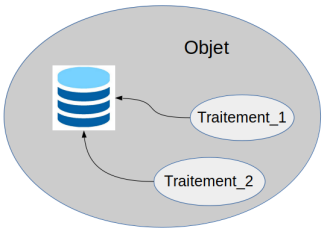
En Python, les données sont représentées sous forme d'objets. Le type de l'objet détermine les opérations que l'on peut appliquer à l'objet et définit aussi les valeurs possibles pour les objets de ce type. La fonction `type` renvoie le type de l'objet (qui est lui-même un objet). Par exemple l'instruction `type("")` renvoie `<class 'str'>`. En effet l'objet ici considéré est une chaîne de caractères vide. La classe correspond à l'implémentation des différents contrats sur chaque opération disponible pour l'objet. **Une classe est une structure de données abstraites**. Cette classe servira ensuite à fabriquer les objets correspodants. C'est en quelque sorte une usine à objets. Il est donc très important de bien définir l'objet à manipuler avant de se lancer dans son implémentation.

Type abstrait de données (TAD) : Date

Lors du dernier TD nous avons réfléchi à la conception d'un type abstrait de données dont quelques caractéristiques sont les suivantes :



Les **axiomes** liés au type `Date` ont également permis de définir la logique des traitements et leurs liens avec les données.



Implémentation

Rappels

L'implémentation de notre type Date passe par la création d'une classe Python. Une classe contient :

- **des propriétés** qui representent les types de données que les objets possèdent
- **un constructeur** qui est appelé lors de la création de l'objet
- **des méthodes** qui possèdent un nom de traitement ou d'opération, un ou plusieurs paramètres (ou aucun) et le code qui implémente le traitement.

Pour en savoir plus : Documentation Python

Exemple

Voici un début d'implémentation pour notre classe Date

```
1  class Date:
2
3      # Propriétés
4      DAYS = ["Sunday"] # À compléter
5      MONTHS = ["January"] # À compléter
6
7      # Constructeur
8      def __init__(self, year):
9          self.year = year
10
11     # Méthode
12     def getYear(self):
13         return self.Year
```

Création d'un objet

```
1  # Création de l'objet date
2  date = Date(2021)
3  # Afficher l'année
4  print(date.getYear())
```

La syntaxe pour accéder à une méthode de l'objet est la suivante : `nomObjet.methode(args)`, le nom de l'objet ou encore référence vers l'objet est suivi d'un point puis du nom de la méthode. Dans notre exemple le nom de l'objet est `date` et la méthode est `getYear()` ce qui donne `date.getYear()`. Cet appel renvoie 2021.
On peut également demander à Python le type de notre nouveau type abstrait de données à l'aide de l'instruction

```
1  print(type(date))
```

Cet appel renvoie `<class '__main__.Date'>`

À vous de jouer

1. Créer le type abstrait de données `Date` et implémenter les différentes opérations définies dans le précédent TD.
2. Écrire quelques tests unitaires permettant de vérifier le bon fonctionnement de votre implémentation.