Sets & Classes solutions

December 7, 2019

1 Classe et Set

Dans cet exercice nous allons travailler sur l'intégration des sets en python et de leur utilisation avec des objets que nous déclarons.

En mathématique un set (ou ensemble) est une collection d'objets distincts clairement définis.

En python, il existe deux classes pour créer des sets; l'une instancie un set muable et l'autre un set immuable, respectivement `set()` et `frozenset()`.

Pour assurer qu'un élément est unique dans un set, python va utiliser le hash de ce dernier à l'aide de la fonction `hash()`. Les listes, dictionnaires et les sets ne sont pas hashable en python, il est donc impossible de les contenir dans un set. Il est possible que deux éléments differents aient le même hash, ceci s'appelle une collision. Pour s'assurer que deux éléments sont identiques quand ils ont le même hash, leurs valeurs sont comparées. Il est primordale que les objets dans un set soient immuables car si le hash d'un élément change, sa présence dans le set pourait être remise en question et python ne sera plus en mesure de le retrouver.

Ce qui nous intéresse maintenant est de rendre une classe que l'on définie compatible avec les sets. Par défaut python utilisera comme input du hash d'un objet son adresse mémoire, ce qui veut dire que les hashs de deux instances ne seront jamais égaux même si les instances sont égales de par leurs attributs.

Pour rendre cela possible, nous pouvons redéfinir les méthodes `__hash__(self)`et `__eq__(self)`. Attention il est obligatoire d'utiliser des attributs immuables. L'une des façons de faire cela est d'utiliser des attributs privés sans définir de methode setter pour les modifier.

1.1 En pratique:

Ci-dessous une classe `Person` avec deux instances ayant la même valeur pour chaque attribut. Retournez leur hash respectif avec la methode `__hash__()` et testez si les objets sont égaux. Ensuite inserez-les dans un set. Que remarquez-vous lorsque vous imprimez le set ?

[1]:
```
class Person():
    def __init__(self,name,surname):
        self.__name = name #on utilise __ devant self.__name pour le l'attribut␣
↪name soit privé
        self.__surname = surname

    def __repr__(self):
        return str(self.__name + " " + self.__surname)
```

```
p1 = Person("Mathieu", "Leroy")
p2 = Person("Mathieu", "Leroy")

p1_hash = p1.__hash__()
p2_hash = p2.__hash__()

print(p1_hash, p2_hash, p1_hash == p2_hash)

set1 = {p1,p2}

print(set1)
```

```
-9223363256743997212 8780110778586 False
{Mathieu Leroy, Mathieu Leroy}
```

Maintenant vous allez redefinir les methodes `__hash__()` et `__eq__()`dans la classe `Person()`ci-dessous et repetez les étapes ci-dessus:

[2]:
```
class Person():
    def __init__(self,name,surname):
        self.__name = name #on utilise __ dans self.__name pour le l'attribut␣
↪name soit privé
        self.__surname = surname

    def __repr__(self):
        return str(self.__name + " " + self.__surname)

    def __eq__(self,other):
        return (
            self.__class__ == other.__class__ and
            self.__name == other.__name and self.__surname == other.__surname )


    def __hash__(self):
        return hash((self.__name,self.__surname))


p3 = Person("Mathieu", "Leroy")
p4 = Person("Mathieu", "Leroy")

p3_hash = p3.__hash__()
p4_hash = p4.__hash__()

print(p3_hash, p4_hash, p3_hash == p4_hash)

set2 = {p3,p4}
```

```
print(set2)
```

```
-1566094862208225612 -1566094862208225612 True
{Mathieu Leroy}
```

[ ]:

heritage_des_constructeurs_answers

December 7, 2019

0.1 L'héritage des constructeurs

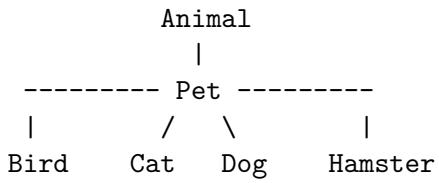
Nous créons une instance d'une classe en appelant son constructeur.
AClass a = new **AClass()**;

Si la classe a une classe parente, le constructeur de la classe appelle toujours le constructeur de la classe parente. Aussi, nous pouvons appeler le constructeur de la classe parente en utilisant **super()**.
(**this()** est utilisé pour appeler la classe elle-même)

0.1.1 L'implementation d'Animal et de Pet

Nous avons implémenté la classe "Animal" et "Pet" pour vous.

Voici une représentation en arbre des relations d'héritage.



La classe "Animal" domine les autres.

Examinez son implémentation.

[0]:
```java
// la classe s'appelle "Animal" et elle est publique
public class Animal {

    // les variables d'instance ; toutes deux sont privées et sont
    // de type int, elles s'appellent respectivement weight et height
    private int weight;
    private int height;

    // le constructeur ; il est public et prend deux paramètres qui
    // sont tous deux de type int, le constructeur définit les variables
    // d'instance et écrit un message à l'utilisateur.
    public Animal(int weight, int height) {
        this.weight = weight;
        this.height = height;
        System.out.println("We're in : public Animal(int weight, int height)");
```

```
    }

    // un accesseur ; il s'appelle getWeight, ne prend aucun paramètre et
    // retourne la valeur du poids
    public int getWeight() {
        return this.weight;
    }

    // un accesseur ; il s'appelle getHeight, ne prend aucun paramètre et
    // retourne la valeur du poids
    public int getHeight() {
        return this.height;
    }

    // un modificateur ; il s'appelle setWeight, prend une valeur int et
    // fixe la valeur du poids
    public void setWeight(int weight) {
        this.weight = weight;
    }

    // un modificateur ; il s'appelle setWeight, prend une valeur int et
    // fixe la valeur du poids
    public void setHeight(int height) {
        this.height = height;
    }

    // implémentation de la méthode "equals()"
    @Override
    public boolean equals(Object o) {
        if (o instanceof Animal) {
            Animal a = (Animal) o;
            return a.getWeight() == this.getWeight() && a.getHeight() == this.
↪getHeight();
        }
        return false;
    }

    // implémentation de la méthode "toString()"
    @Override
    public String toString() {
        return "Weight: " + this.getWeight() + " Height: " + this.getHeight();
    }
}
```

[0]: com.twosigma.beaker.javash.bkrb994444b.Animal

[0]:
```
public class Pet extends Animal {

    private String name;

    public Pet(int weight, int height) {
        super(weight, height);
        System.out.println("We're in : public Pet(int weight, int height)");
    }

    public Pet(int weight, int height, String name) {
        this(weight, height);
        this.name = name;
        System.out.println("We're in : public Pet(int weight, int height,␣
 ↪String name)");
    }

    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }


    // Ici nous utilisons le mot-clé "super" pour appeler la méthode
    // "equals" de la classe Animal au lieu de l'écrire à nouveau
    @Override
    public boolean equals(Object o) {
        if (o instanceof Pet) {
            Pet p = (Pet) o;
            return super.equals(p) && this.getName().equals(p.getName());
        }
        return false;
    }

    // Ici nous utilisons le mot-clé "super" pour appeler la méthode
    // "toString" de la classe Animal au lieu de l'écrire à nouveau.
    @Override
    public String toString() {
        return super.toString() + " Name: " + this.getName();
    }
}
```

[0]:
```
com.twosigma.beaker.javash.bkrb994444b.Pet
```

Ici, nous créons une instance de la classe Pet. Remarquez comme elle est créée en appelant le

constructeur de la classe-même ainsi que le constructeur de la classe parente.

[0]:
```
Pet myPet = new Pet(4, 30, "My Pet's Name");
```

```
We're in : public Animal(int weight, int height)
We're in : public Pet(int weight, int height)
We're in : public Pet(int weight, int height, String name)
```

[0]:
```
null
```

0.1.2 L'implementation de la classe Cat

Mettez en œuvre l'un des enfants de la classe "Pet" (oiseau, chat, chien, hamster).

La classe doit avoir:
1. deux variables d'instance privées appelées "breed" et "favoriteToy" (de type String)
2. deux constructeurs ;
2.1. le premier prend weight, height, name et breed comme paramètres, définit les variables et écrit un message à l'utilisateur
2.2. le second prend weight, height, name, breed et favoriteToy comme paramètres, définit les variables et écrit également un message à l'utilisateur
3. méthodes d'accesseurs et de modificateurs des variables breed et favoriteToy
4. remplace les méthodes equals et toString

[0]:
```
// VOTRE CODE ICI
public class Cat extends Pet {

    private String breed;
    private String favoriteToy;

    public Cat(int weight, int height, String name, String breed) {
        super(weight, height, name);
        this.breed = breed;
        System.out.println("We're in : public Cat(int weight, int height,␣
↪String name, String breed)");
    }

    public Cat(int weight, int height, String name, String breed, String␣
↪favoriteToy) {
        this(weight, height, name, breed);
        this.favoriteToy = favoriteToy;
        System.out.println("We're in : public Cat(int weight, int height,␣
↪String name, String breed, String favoriteToy");
    }

    public String getBreed() {
        return this.breed;
    }
```

```
    public String getFavoriteToy() {
        return this.favoriteToy;
    }

    public void setBreed(String breed) {
        this.breed = breed;
    }

    public void setFavoriteToy(String favoriteToy) {
        this.favoriteToy = favoriteToy;
    }

    @Override
    public boolean equals(Object o) {
        if (o instanceof Cat) {
            Cat c = (Cat) o;
            return super.equals(c) && this.getBreed().equals(c.getBreed()) &&␣
 ↪this.getFavoriteToy().equals(c.getFavoriteToy());
        }
        return false;
    }

    @Override
    public String toString() {
        return super.toString() + " Breed: " + this.getBreed() + " Favorite Toy:
 ↪ " + this.getFavoriteToy();
    }
}
```

[0]: com.twosigma.beaker.javash.bkrb994444b.Cat

Créez une instance de la classe Cat et observez le résultat.

[0]:
```
// VOTRE CODE ICI
Cat myCat = new Cat(3, 40, "Pharaoh", "sphynx", "laser pointer");
```

```
We're in : public Animal(int weight, int height)
We're in : public Pet(int weight, int height)
We're in : public Pet(int weight, int height, String name)
We're in : public Cat(int weight, int height, String name, String breed)
We're in : public Cat(int weight, int height, String name, String breed, String
favoriteToy
```

[0]: null

[0]:

Class_student_java_sol

December 7, 2019

1 Syntaxe Java

Pour plus d'informations sur la syntaxe de Java, veuillez vous référer à la documentation de wikiversity, w3schools ou geeksforgeeks:

https://fr.wikiversity.org/wiki/Java/Classes_et_objets

https://www.w3schools.com/java/default.asp

https://www.geeksforgeeks.org/java-tutorials/

Dans la cellule suivante, vous trouverez un exemple de déclaration d'une classe `Dog`. Veuillez y jeter un oeil et l'exécuter.

[4]:
```java
// source: https://www.geeksforgeeks.org/classes-objects-java/
// Class Declaration
public class Dog
{
    // Instance Variables
    String name;
    String breed;
    int age;
    String color;

    // Constructor Declaration of Class
    public Dog(String name, String breed,
                   int age, String color)
    {
        this.name = name;
        this.breed = breed;
        this.age = age;
        this.color = color;
    }

    // method 1
    public String getName()
    {
        return name;
    }
```

```
    // method 2
    public String getBreed()
    {
        return breed;
    }

    // method 3
    public int getAge()
    {
        return age;
    }

    // method 4
    public String getColor()
    {
        return color;
    }

    @Override
    public String toString()
    {
        return("Hi my name is "+ this.getName()+
               ".\nMy breed,age and color are " +
               this.getBreed()+"," + this.getAge()+
               ","+ this.getColor());
    }

    public static void main(String[] args)
    {
        Dog tuffy = new Dog("tuffy","papillon", 5, "white");
        System.out.println(tuffy.toString());
    }
}
```

[4]: com.twosigma.beaker.javash.bkr246d5b64.Dog

[5]:
```
Dog.main(null);
```

```
Hi my name is tuffy.
My breed,age and color are papillon,5,white
```

[5]: null

1.0.1 Exercise

La semaine passée vous avez eu l'occasion de créer une classe nommée `Student`. En python, la solution se présentait comme ci-dessous.

‘class Student:

```
def __init__(self, name:str, year:int): #constructeur
    self.name = name
    self.year = year
    self.grades = []

def get_name(self):
    return self.name

def get_year(self):
    return self.year

def add_grade(self, grade:int):
    self.grades.append(grade)

def final_grade(self):
    if len (self.grades)>0:
        return sum(self.grades)/len(self.grades)
    else:
        return 0

def evaluate_year(self):
    if self.final_grade() >= 4:
        self.year += 1
        self.grades = []

def __str__(self):
    return self.name + " is in year " + str(self.year) + ("  with average grade " + str(self.fi
```

Tanguy = Student(“Tanguy”, 1)

Tanguy.add_grade(4)

Tanguy.evaluate_year()

print(Tanguy.__str__())‘

A faire:

Reprenez les informations importantes de cette solution pour réecrire la même classe en java. Inspirez vous également de la classe Dog donnée ci-dessus.

[7]:
```
// svp écrivez tout votre code dans la même cellule code

// imports
import java.util.ArrayList; // for ArrayList functions
import java.util.*;

// Class Declaration
```

```java
public class Student{

    // Instance Variables
    String name;
    int year;
    ArrayList<Integer> grades = new ArrayList<>();

    // Constructor Declaration of Class
    public Student(String name, int year){
        this.name = name;
        this.year = year;
    }

    // method 1
    public String getname(){
        return name;
    }

    // method 2
    public int getyear(){
        return year;
    }

    // method 3
    public void add_grade(int grade){
        grades.add(grade); // using add() to initialize values
    }

    // method 4
    public int final_grade(){
        int sum=0;
        for (int i=0; i< grades.size();i++){
            sum += grades.get(i); // get the sum of the elements in the list
        }
        // double avg = sum/grades.size();
        if (grades.size() > 0){
             return sum/grades.size();
        } else{
            return 0;
        }
    }

    // method 5
    public void evaluate_year(){
        if (this.final_grade() >= 4)
            this.year += 1;
            grades.clear(); // clear fucntion used
```

```
    }

    // method 6
     @Override
    public String toString(){
        return(this.getname()+ " is in year " +
               this.getyear() + "  with average grade " +
               grades + this.final_grade());

    }

     // method  main
    public static void main(String[] args){
        Student t = new Student("Tanguy",1);
        t.add_grade(4);
        t.evaluate_year();
        System.out.println(t.toString());

    }

}
```

[7]: com.twosigma.beaker.javash.bkr246d5b64.Student

[8]:
```
Student.main(null);
```

```
Tanguy is in year 2  with average grade []0
```

[8]: null

[ ]:

Class_basics_sol

December 7, 2019

0.1 Exercise de base

-Déclarez une classe publique `JavaAppbasics`

-Déclarez une méthode publique statique `main`

-Affichez:

```
`Hello!
 My name is  Webskater! What is yours?`
```

-Demandez à l'utilisateur d'introduire son prénom

-Affichez:

```
`Nice to meet you, prenom
 Goodbye ...`
```

[4]:
```java
//package javaappbasics;

import java.util.Scanner;

/*
 * @author tanjamarkotic inspired by openclassroom
*/

public class JavaAppbasics {
// nom de classe avec majuscule par convention
// au minimum une classe pour programme java

      public static void main(String[] args) {
        // TODO code application logic here
        // point de départ avec méthode main
        // une seule méthode main par projet
        System.out.println("Hello! ");
        // saut de ligne à la fin avec println
        System.out.print("My name is ");
        System.out.print(" Webskater!");
        // objet out de classe System
        // JDK pour compilation de programme
```

```
        // avec précompilation en byte code
        System.out.println(" What is yours? ");
        Scanner sc = new Scanner(System.in);
        // comme fonction input en python
        String str = sc.nextLine();
        // ou .nextInt()
        System.out.println("Nice to meet you, " + str);
        System.out.println("Goodbye …");
        //fin de la boucle si réponse Non
    }

}
```

[4]: com.twosigma.beaker.javash.bkr90234c16.JavaAppbasics

[5]:
```
JavaAppbasics.main(null);
```

```
Hello!
My name is  Webskater! What is yours?
Tanja
Nice to meet you, Tanja
Goodbye …
```

[5]: null

[ ]: