Durée du CC : 1h15

Prénom ..... Nom : .....
Toutes les méthodes demandées sont testées dans le code.

1 Union européenne

Dans cette première partie, on va représenter des pays d'Europe comme instances de la classe `Pays`.

Question 1 Compléter la première partie du corps du constructeur de la classe `Pays` (c'est-à-dire la boîte ligne 4 – on ne remplira pas la boîte ligne 6 pour le moment) pour que chaque pays ait comme attribut

- un `nom`,
- une `capitale` (sous la forme d'une chaîne de caractères),
- `habitants`, le nombre (en millions) d'habitants du pays,
- un drapeau booléen `dansUE` précisant si le pays est membre de l'UE.

Question 2 Remplir la boîte ligne 9 de sorte qu'un pays s'affiche selon le format indiqué entre les lignes 15 et 17.

Question 3 Implémenter la méthode `__eq__()` pour que deux `Pays` soient considérés comme égaux dès qu'ils ont le même `nom` (cf. ligne 20).

On s'intéresse plus particulièrement aux pays de l'Union européenne, qui seront stockés dans le dictionnaire `ue` (défini comme vide à la ligne 1). Ce dictionnaire suivra le format `clef:valeur` où `valeur` est un `Pays` ayant comme `clef` son `nom`.

Question 4 Remplir la deuxième partie du constructeur de `Pays` (ligne 6) de sorte que lorsqu'on crée un pays membre de l'UE, ce `Pays` soit ajouté dans le dictionnaire `ue`, avec comme clef son `nom` (cf. lignes 23 à 30).

Question 5 Implémenter la fonction `habitantsUE()` (ligne 32) qui renvoie le nombre total d'habitants de l'Union européenne (en millions).

Question 6 Ligne 37, définir une variable `l` qui contienne la liste des pays de l'UE dont la capitale termine par la lettre `'s'`. On utilisera une définition par compréhension.

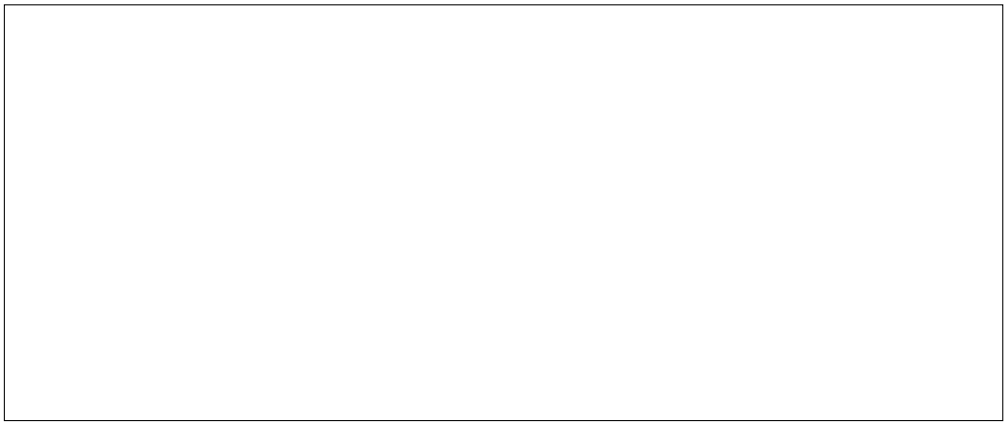
2 Entiers pairs

Question 7 Remplir les boîtes ligne 46 et 49 de sorte qu'un objet de la classe `EntierPair` attende à la création un itérable `iterable` (qui pourra être n'importe quel itérateur, et pas forcément une liste) énumérant des entiers. Cet `EntierPair` devra lui aussi être un itérable, qui énumérera seulement les entiers pairs parmi ceux de l'itérable `iterable`.

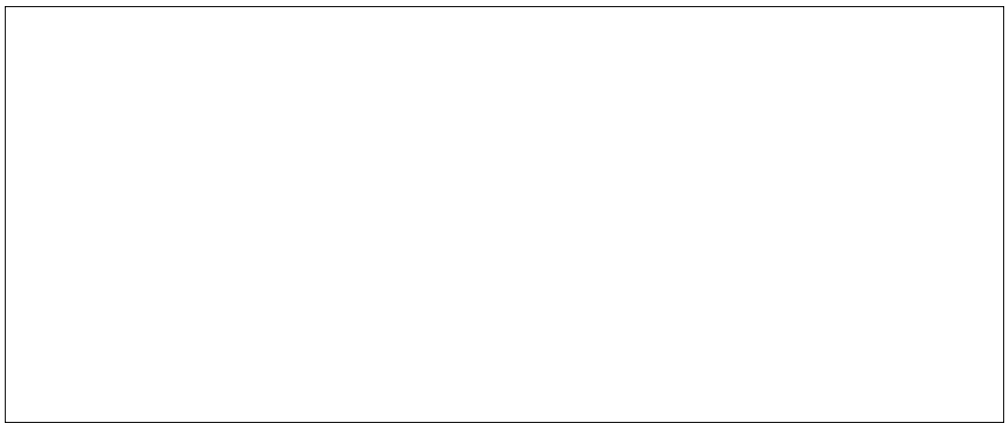
Le comportement attendu est illustré entre les lignes 52 et 60.

```
1   ue = {}
2
3   class Pays:
4       def __init__(self,nom,capitale,habitants,dansUE):
5
6
7
8
9
10
11
```

```
12        def __eq__(self,other):
```



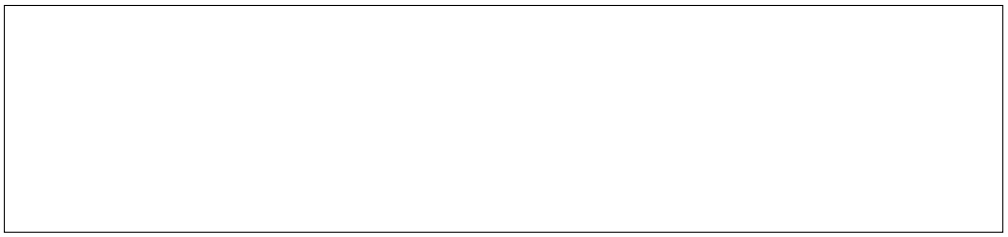
```
13
14
15    gb1=Pays("Royaume-Uni","Londres",68,False)
16    print(gb1)
17        # Royaume-Uni, capitale Londres, 68 millions d'habitants
18
19    gb2=Pays("Royaume-Uni","London",68,False)
20    print(gb1==gb2)
21        # True
22
23    f=Pays("France","Paris",68,True)
24    n=Pays("Norvège","Oslo",5,False)
25    rh=Pays("Croatie","Zagreb",4,True)
26    gr=Pays("Grèce","Athènes",10,True)
27    print(ue)
28      # {'France': France, capitale Paris, 68 millions d'habitants,
29      # 'Croatie': Croatie, capitale Zagreb, 4 millions d'habitants,
30      # 'Grèce': Grèce, capitale Athènes, 10 millions d'habitants}
31
32    def habitantsUE():
```



```
33
34
```

```
print(habitantsUE())
    # 82 (=68+4+10) ; l'UE n'est pas au complet

```

```

print(l)
    #['France', 'Grèce']

class EntiersPairs:
    def __init__(self,iterable):
        self.iterable=iterable

    def __iter__(self):
```



```

class EntiersPairsIt:
```

```
52  ep=EntiersPairs([3,1,4,1,5,9,2])
53
54  for n in ep:
55      print(n)
56      # affiche 4 puis 2
57
58  for n in ep:
59      print(n)
60      # affiche 4 puis 2
```