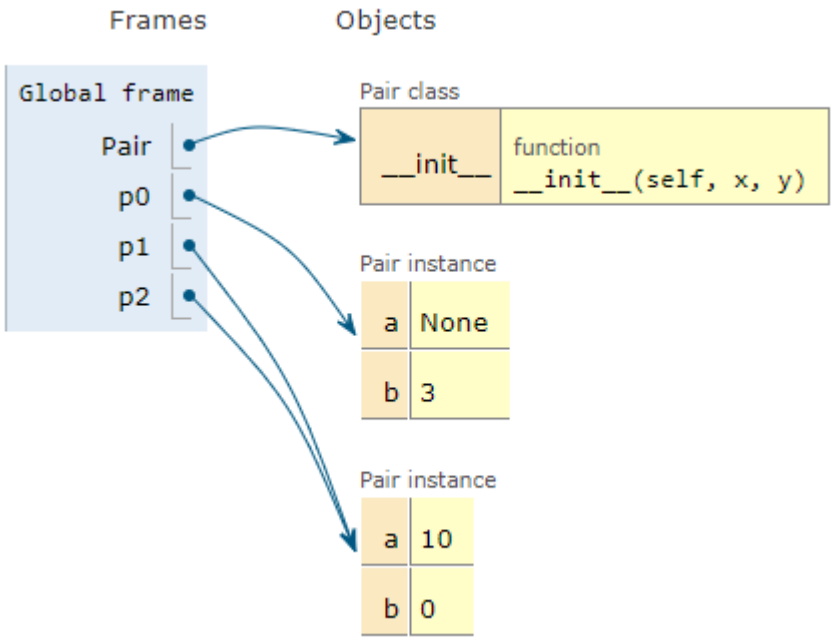
Auteur: GG, Ba38

1. Différences entre égalité et référence

- Pour ceux qui n'ont pas assisté à la fin de la séance avec l'explication sur l'exercice "Une classe simple", je vous invite à le compléter chez vous avant de démarrer la mission.
- Cet exercice est crucial pour comprendre la notion de référence et d'égalité
- Conseil: utiliser Python Tutor ou l'outil debug de Thonny pour visualiser ce qu'il se passe en mémoire



2. Mise au point sur `self`

A quoi sert-il ?

- Utilisé en premier paramètre de toute méthode d'instance, il fait référence à l'instance (= l'objet) en cours, celui qui l'appelle.
    - Il permet par exemple de faire la différence entre de simples paramètres (`a` et `b`) et des variables d'instance (ou attributs) `self.a` et `self.b`

```
def __init__(self,a=0,b=0):
    self.a = a
    self.b = b
```

    - Il permet de distinguer les fonctions, des méthodes d'instance:

```
class Student :
    """ NB. Description, Pre and Post should be completed """
    def __init__(self,n) :
        self.name = n
        self.test1 = None
```

```
        self.test2 = None

    # UNE METHODE D'INSTANCE
    def average_score(self) :
        return (self.test1 + self.test2) / 2

# UNE FONCTION, en dehors de la définition de la classe
def average_score_bis(student):
    return (student.test1 + student.test2) / 2

stud = Student("Kim")
stud.test1, stud.test2 = 4, 10
res1 = stud.average_score()
res2 = average_score_bis(stud)

print(res1==res2)
```

- Dans le cas des listes, il en va de même entre `sort` et `sorted` 😉

Oui, c'est une convention entre développeurs

- Le premier paramètre de toutes les méthodes est une instance, mais il n'a pas de nom obligatoire.
- Ce code marche parfaitement :

```
class Pair:
""" Une classe représentant une paire d'entiers """
def __init__(kreatur,a=0,b=0):
    kreatur.a = a
    kreatur.b = b

def __str__(dumbledore):
    return "{},{}".format(dumbledore.a,dumbledore.b)

p = Pair()
print(p)
```

- Il ne passera probablement pas une peer code review, mais il est valide.

MAIS soyez conventionnels, utilisez `self` 😉

3. Les méthodes spéciales

Sont une convention vis à vis de Python!

- Les méthodes spéciales (ou "magiques") prennent la forme `__methodespeciale__` en Python
- Les fonctionnalités "spéciales" peuvent être de nature diverse et agissent souvent de manière invisible :

  - `__file__` indique l'adresse d'un fichier Python
  - `__eq__` est exécuté quand l'expression `a == b` est exécutée
  - `__str__` est exécuté quand `print()` est exécuté
  - `__init__` est exécuté à la construction d'un objet, pour initialiser ses attributs, e.g. quand `s = Student("Kim")` est exécuté

- Dans le code suivant, qu'est-ce qui est affiché à la console ? 😉

```
class Pair:
""" Une classe représentant une paire d'entiers """

def init(self, a=42, b=42):
        self.a = a
        self.b = b

def __init__(kreatur,a=0,b=0):
        kreatur.a = a
        kreatur.b = b

def str(dumbledore):
        print("{},{}".format(dumbledore.a,dumbledore.b))

p = Pair()
print(p)
print(p.a)
```