TD Python 3 : programmation orientée objet

« Everything in Python is an object »

Une petite initiation à la programmation orientée objet en langage Python.

1. Classe Rectangle

Écrire le code de la classe Rectangle.

Cette classe possède :

3 attributs d'instance :

- *longueur* (type float)
- *largeur* (type float)
- *nombreCotes* (nombre de côtés) (type int)

3 méthodes d'instance :

- Un constructeur : méthode spéciale *__init__()*
  Le constructeur a deux paramètres : *longueur* et *largeur*

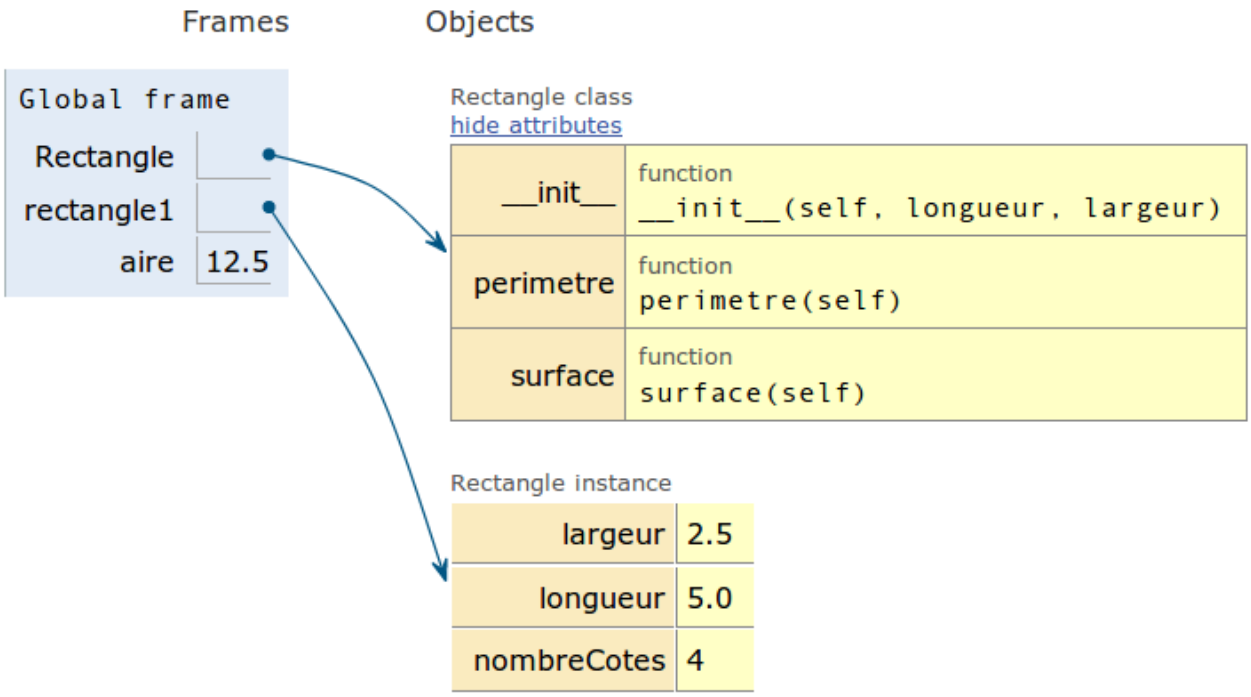
- Une méthode *surface()* qui retourne la surface (type float)
- Une méthode *perimetre()* qui retourne le périmètre (type float)

Exemple d'utilisation :

```
>>> rectangle1 = Rectangle(5, 2.5)
>>> aire = rectangle1.surface()
>>> print(aire)
>>> print(rectangle1.largeur)
>>> help(rectangle1)
```

Bibliographie : https://docs.python.org/3/tutorial/classes.html

Un outil intéressant pour suivre l'évolution des variables : http://www.pythontutor.com/



2. Classe Carre (carré)

Par rapport à la classe Rectangle, cette classe possède un attribut d'instance *cote* (type float) en lieu et place des attributs *longueur* et *largeur*.

Exemple d'utilisation :

```
>>> carre1 = Carre(3.8)
>>> print(carre1.perimetre())
```

3. Héritage de classes

Un carré étant un cas particulier d'un rectangle, réécrire la classe Carre en s'appuyant sur la classe mère Rectangle.

Bibliographie : https://docs.python.org/3/library/functions.html#super

4. Diagramme de classes

Dessiner le diagrammes de classes.
On pourra utiliser l'application Umbrello (https://umbrello.kde.org).

5. Attribut en lecture seule

Tester et commenter le code suivant :

```
>>> rectangle2 = Rectangle(3, 2)
>>> print(rectangle2.nombreCotes)
>>> rectangle2.nombreCotes = 5
>>> print(rectangle2.nombreCotes)
```

On veut avoir maintenant accès à l'attribut *nombreCotes* en lecture mais sans possibilité de modification :

```
>>> print(rectangle2.nombreCotes)
```

doit donner 4

```
>>> rectangle2.nombreCotes = 5
```

doit provoquer une exception.

Modifier le code.

Quelques indications :

En python, tout est « public ».
La notion de variable « privée » n'existe pas.
Une (mauvaise ?) manière d'obtenir le comportement d'une variable privée est de préfixer son nom avec un double underscore :

```
__monattribut
```

Puis d'utiliser un « getter » pour un accès en lecture (voir la fonction property() ou le décorateur @property).

Bibliographie : https://docs.python.org/3/library/functions.html#property

6. Attribut de classe

Un attribut de classe est un attribut partagé par toutes les instances de la classe.

Créer un attribut de classe *compteur* qui donne le nombre d'instances de la classe Rectangle.

```
>>> rectangle1 = Rectangle(2.5, 5.0)
>>> rectangle2 = Rectangle(2, 10)
>>> rectangle3 = Rectangle(5, 8)
>>> print(rectangle1.compteur)
```

doit donner 3

```
>>> print(Rectangle.compteur)
```

doit donner 3

```
>>> del(rectangle1)
>>> print(Rectangle.compteur)
```

doit donner 2

```
>>> carre1 = Carre(4.5)
>>> print(Rectangle.compteur)
```

>>> print(Carre.compteur)

Remarque : il faudra définir un destructeur (méthode spéciale *__del__())*

Créer un attribut de classe *compteurCarre* qui donne le nombre d'instances de la classe Carre (sans double comptage !).