NSI · TERMINALE / EXERCICES CORRIGÉS

La programmation orientée objet

Astrolab Cours · astrolabcours.fr

Ex. 1 Créer un objet ●○○ Application

Avec la classe `Compte` du cours, que vaut `c.solde` après ces lignes ?

```
c = Compte("Tom")
c.deposer(100)
c.retirer(30)
c.retirer(200)
```

Indice

Le solde par défaut vaut 0. Un retrait trop gros est refusé.

Ligne par ligne

Solde 0, puis 100, puis 70. Le retrait de 200 est refusé (renvoie False).

La réponse

`c.solde` vaut 70.

Ex. 2 Une classe Point ●●○ Méthode

Écrire une classe `Point` avec deux attributs `x` et `y`, et une méthode `distance_origine` qui renvoie la distance du point à l'origine.

Indice

Distance : $\sqrt{x^2 + y^2}$.

© Astrolab Cours · astrolabcours.fr/nsi/terminale/programmation-objet/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits

La classe

```
from math import sqrt

class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance_origine(self):
        return sqrt(self.x ** 2 + self.y ** 2)
```

Ex. 3 Le self oublié ●●○ Piège

Un élève écrit :

```
class Chat:
    def __init__(self, nom):
        nom = nom
    def miauler(self):
        return self.nom + " miaule"
```

`Chat("Félix").miauler()` provoque une erreur. Pourquoi ?

Indice

Où l'attribut est-il créé ?

L'erreur

`nom = nom` crée une variable locale, pas un attribut : l'objet n'a pas d'attribut `nom`.

La correction

Il faut écrire `self.nom = nom` dans le constructeur.

Ex. 4 Un chronomètre ●●● Type bac

Écrire une classe `Duree` qui stocke des heures, minutes et secondes, avec une méthode `en_secondes()` et une méthode `ajouter(s)` qui ajoute `s` secondes en gardant minutes et secondes entre 0 et 59.

Indice

Passer par le total en secondes, puis reconvertir avec // et %.

© Astrolab Cours · astrolabcours.fr/nsi/terminale/programmation-objet/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits

L'idée

On convertit tout en secondes, on ajoute, puis on redécoupe : heures = total // 3600, minutes = (total % 3600) // 60, secondes = total % 60.

La classe

```
class Duree:
    def __init__(self, h, m, s):
        self.h, self.m, self.s = h, m, s

    def en_secondes(self):
        return 3600 * self.h + 60 * self.m + self.s

    def ajouter(self, s):
        t = self.en_secondes() + s
        self.h = t // 3600
        self.m = (t % 3600) // 60
        self.s = t % 60
```

© Astrolab Cours · astrolabcours.fr/nsi/terminale/programmation-objet/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits