Algorithmique - paradigmes

Informatique et Science du Numérique

Les paradigmes de programmation

Approche conceptuelle de l'écriture d'un programme.

Programmation impérative (Fortran, COBOL, BASIC, Pascal, C, PHP, ...)
Programmation procédurale (Fortran, COBOL, BASIC, Pascal, C, PHP, ...)
Programmation structurée (Pascal, C, PHP, ...)
Programmation fonctionnelle (Lisp, OCaml, ...)
Programmation évènementielle (javascript, ...)
Programmation logique (Prolog, ...)
Programmation orientée objet (C++, PHP, Python, …)
...

Algorithmique - paradigmes

Informatique et Science du Numérique

Exemple : la balle

Une balle est lâchée d'une hauteur H = 10 m et rebondit sur le sol. À chaque rebond, le sol absorbe 10 % de l'énergétique de la balle. La balle s'arrête de rebondir lorsque la hauteur du rebond devient inférieure à 1 mm.

Déterminer :
1. le nombre N de rebonds effectués
2. la distance d, en mètres, parcourue par la balle

Algorithmique - paradigmes

Informatique et Science du Numérique

```
Algorithme Rebond
constantes
        (hauteur : réel) := 10000.0     {en mm}
        (limite : réel) := 1.0          {balle à l'arrêt}
        (abs : réel) := 0.1
variables
        (rebond : entier) := 0
        (distance : réel) := hauteur
        (h : réel) := hauteur
début
        tantque (h > limite) ET (abs < 1) faire
        début
                h := h * (1 - abs)
                distance := distance + (2 * h)
                rebond := rebond + 1
        fin

        afficher(rebond, distance / 1000)
fin
```

Algorithmique - paradigmes

Informatique et Science du Numérique

La modèle de la boite noire (fonction)

hauteur
limite
absorption

rebond
distance

généricité
Abstraction IHM

Une boite noire doit rester noire...
et étanche : sortie = f(entrée)

Algorithmique - paradigmes

Informatique et Science du Numérique

Implémentation

Programmation procédurale
En langage C, tout est fonction

Programmation orientée objet
En Python, tout est objet

Algorithmique - paradigmes

Informatique et Science du Numérique

```
// fonction prototype
static unsigned int calcul_rebonds(const float, const float, const float, unsigned int *, float *);

int main()
{
        unsigned int rebond;
        float distance;

        calcul_rebonds(1000, 1, 0.1, &rebond, &distance) ;

        printf("%d -> %.2f m", rebond, distance / 1000);

        return 0;
}

static void calcul_rebonds(const float hauteur, const float limite, const float abs, unsigned int *rebond, float *distance)
{
        *rebond   = 0;
        *distance = hauteur;

        float h = hauteur;

        while ( h > limite && abs < 1 )
        {
                h         = h * (1 - abs);
                *distance = *distance + (2 * h);
                *rebond   = *rebond + 1;
        }
}
```

on utilise des pointeurs

on passe les @ des variables

on met à jour les valeurs passées par @

Algorithmique - paradigmes

Informatique et Science du Numérique

```
# coding: utf-8

def calcul_rebonds(hauteur, limite, abs, rebond, distance):
                h = hauteur

                while h > limite and abs < 1:
                                h           = h * (1 - abs)
                                distance = distance + (2 * h)
                                rebond   = rebond + 1


rebond   = 0
distance = 1000

calcul_rebonds(1000, 1, 0.1, rebond, distance)

print(rebond, distance / 1000)
```

on variables sont passées par valeur

Le résultat attendu n'est pas correct

Solutions :

1. passer un objet mutable en paramètre (liste)
2. utiliser le paradigme objet

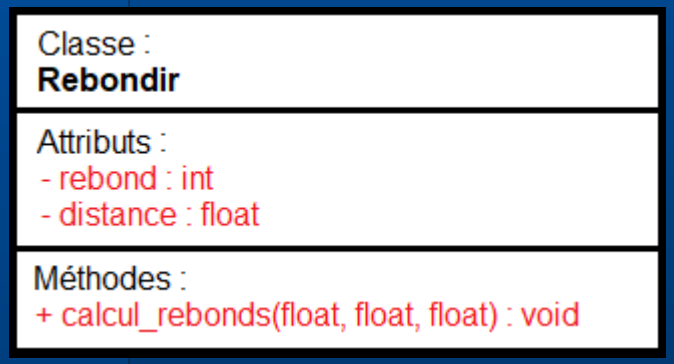
Algorithmique - paradigmes

Informatique et Science du Numérique

La POO

1. Création d'une classe

2. Instanciation de la classe pour la manipuler en tant qu'objet



Nom de la classe

Variables internes (encapsulées)

Fonctions appelées par l'instance

Conservation du modèle de la boite noire
Les E/S s'effectuent par échanges de messages

Algorithmique - paradigmes

Informatique et Science du Numérique

```
# coding: utf-8

class rebondir:
    def __init__(self):
        self.__rebond = 0
        self.__distance = 0

    def get_rebond(self):
        return self.__rebond

    def get_distance(self):
        return self.__distance

    def calcul_rebonds(self, hauteur, limite, abs):
        # les erreurs ne sont pas traitées...
        self.__rebond  = 0
        self.__distance = hauteur

        while hauteur > limite and abs < 1:
            hauteur *= (1 - abs)
            self.__distance += 2 * hauteur
            self.__rebond   += 1

r = rebondir()                          # instanciation de la classe
r.calcul_rebonds(1000, 1, 0.1)          # appel à la méthode de calcul
print(r.get_rebond(), r.get_distance())  # appels aux accesseurs
```

Attributs (propriétés)

Accesseurs des attributs

Méthode (fonction)