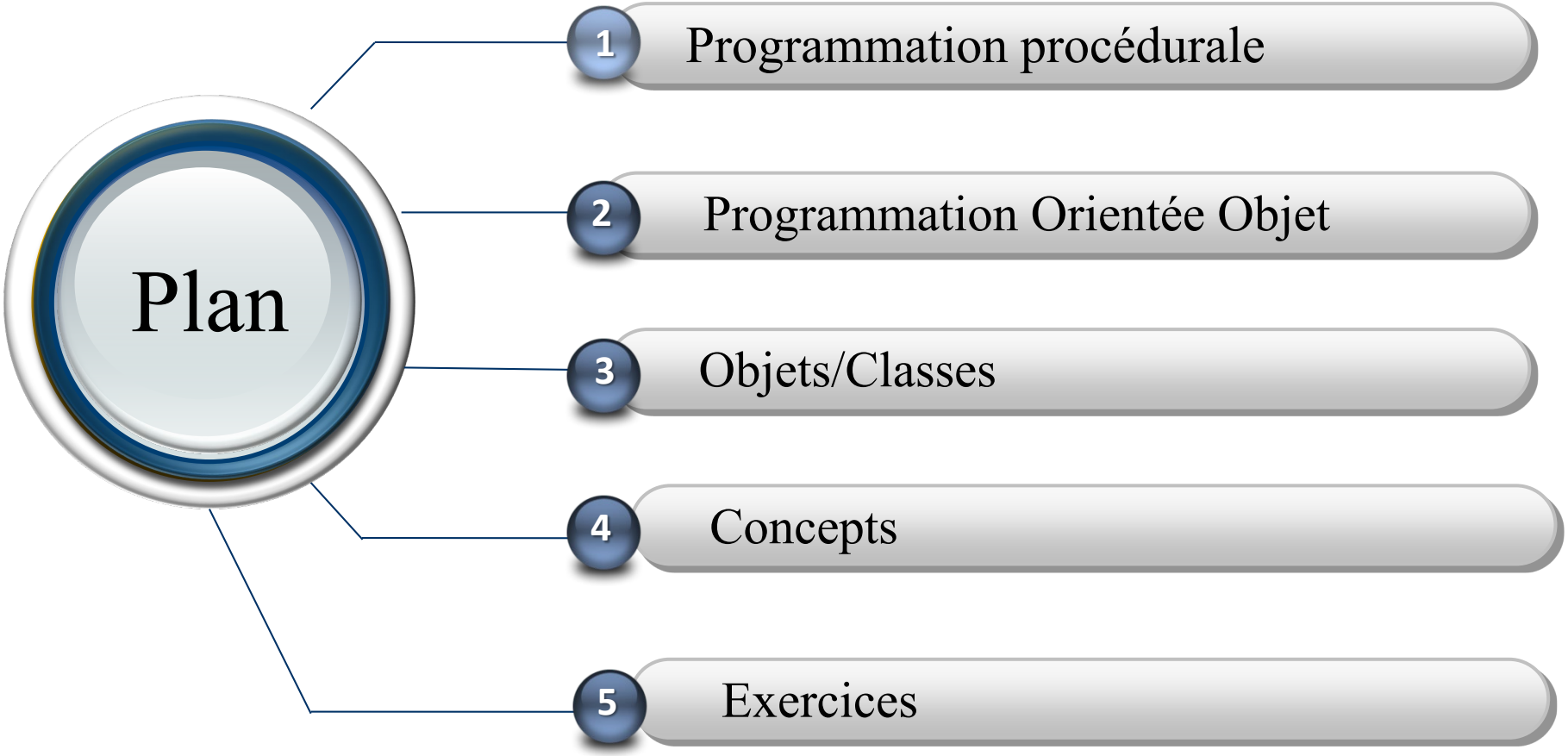
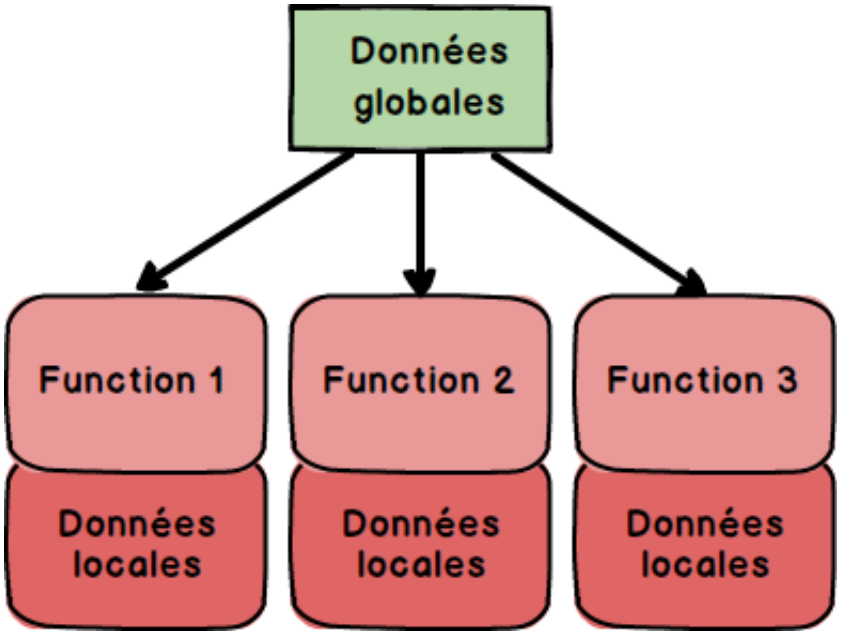
Informatique 3

Dr. Ing. NOUBISSI Justin-Hervé

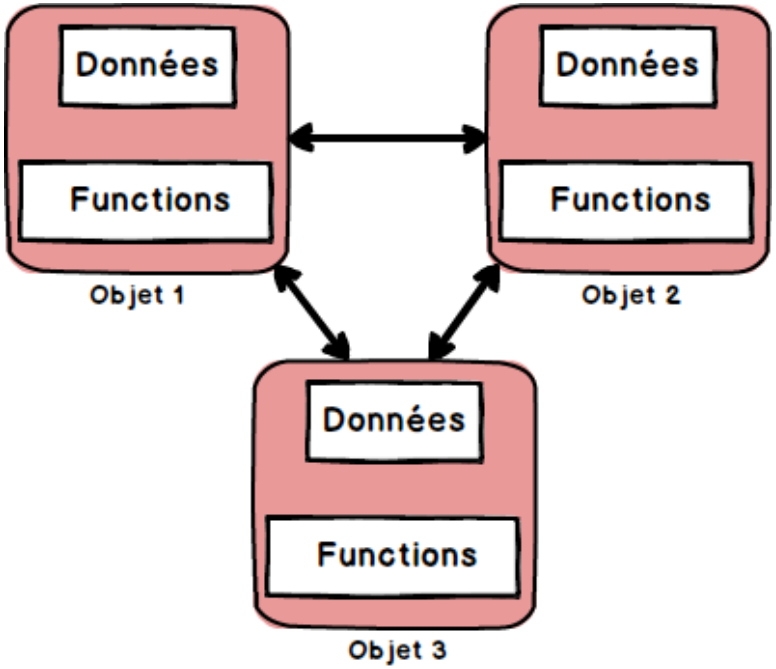

Plan
1
Programmation procédurale
2
Programmation Orientée Objet
3
Objets/Classes
4
Concepts
5
Exercices

Programmation procédurale



- Programme divisé en petites parties (**fonctions** ou **procédures**)
- Procédure à exécuter étape par étape
- Problème décomposé en petites parties
- Une ou plusieurs fonctions utilisées pour résoudre chaque partie
- Données facilement accessibles et modifiables

Programmation Orientée Objet (P.O.O)



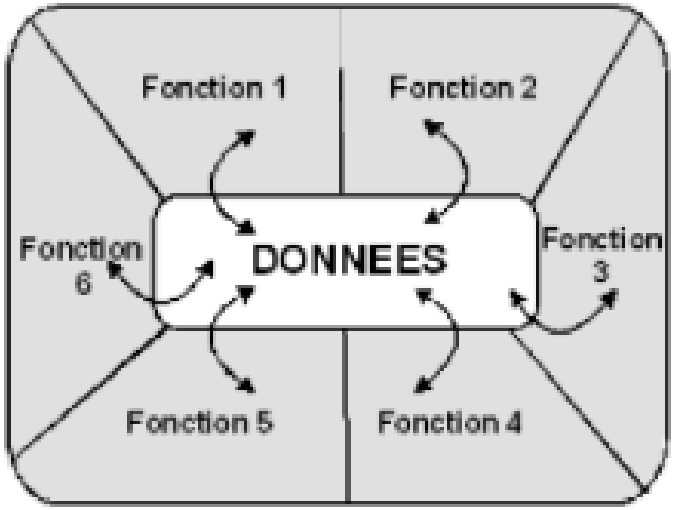
- Programme divisé en parties appelées **Objets**.
- Se concentre sur l'objet et les données plutôt que sur les actions
- Corrige les défauts du **programmation procédurale**
- Améliore la sécurité des données
- Améliore l'initialisation et le nettoyage automatiques des objets

Tableau comparatif

	Programmation Procédurale	Programmation Orientée Objet
Programmes	Le programme principal est divisé en petites parties selon les fonctions.	Le programme principal est divisé en petit objet en fonction du problème.
Les données	Chaque fonction contient des données différentes.	Les données et les fonctions de chaque objet individuel agissent comme une seule unité.
Permission	Pour ajouter de nouvelles données au programme, l'utilisateur doit s'assurer que la fonction le permet.	Le passage de message garantit l'autorisation d'accéder au membre d'un objet à partir d'un autre objet.
Exemples	Pascal, Fortran	PHP5, C ++, Java. **Python**
Accès	Aucun spécificateur d'accès n'est utilisé.	Les spécificateurs d'accès public, private, et protected sont utilisés.
La communication	Les fonctions communiquent avec d'autres fonctions en gardant les règles habituelles.	Un objet communique entre eux via des messages.
Contrôle des données	La plupart des fonctions utilisent des données globales.	Chaque objet contrôle ses propres données.
Importance	Les fonctions ou les algorithmes ont plus d'importance que les données dans le programme.	Les données prennent plus d'importance que les fonctions du programme.
Masquage des données	Il n'y a pas de moyen idéal pour masquer les données.	Le masquage des données est possible, ce qui empêche l'accès illégal de la fonction depuis l'extérieur.

P.O.O : Objets et Classes

Objet



- Exemples : Ma voiture, Ton ordinateur

Classe (d'objets)

- Exemples : Les voitures, Les ordinateurs

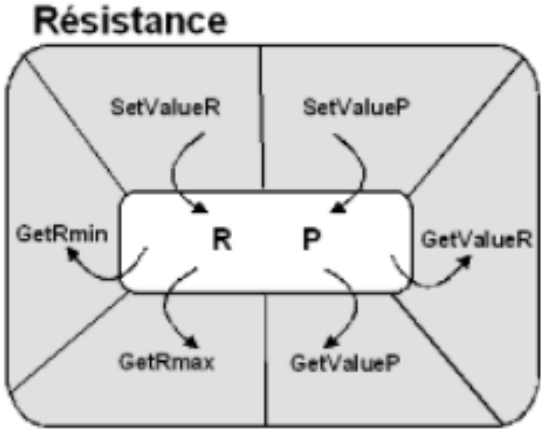
P.O.O : Objets et Classes

Classe (d'objets)

- Type de données défini par l'utilisateur
- Se comporte comme un type de langage de langage de programmation intégré
- Les objets peuvent être vus comme des variables de classes
- C'est une collection d'objets de type similaire

P.O.O : Concepts

Exemples d'objet



Planete
•*Type* •*DistanceAuSoleil* •*Diametre*

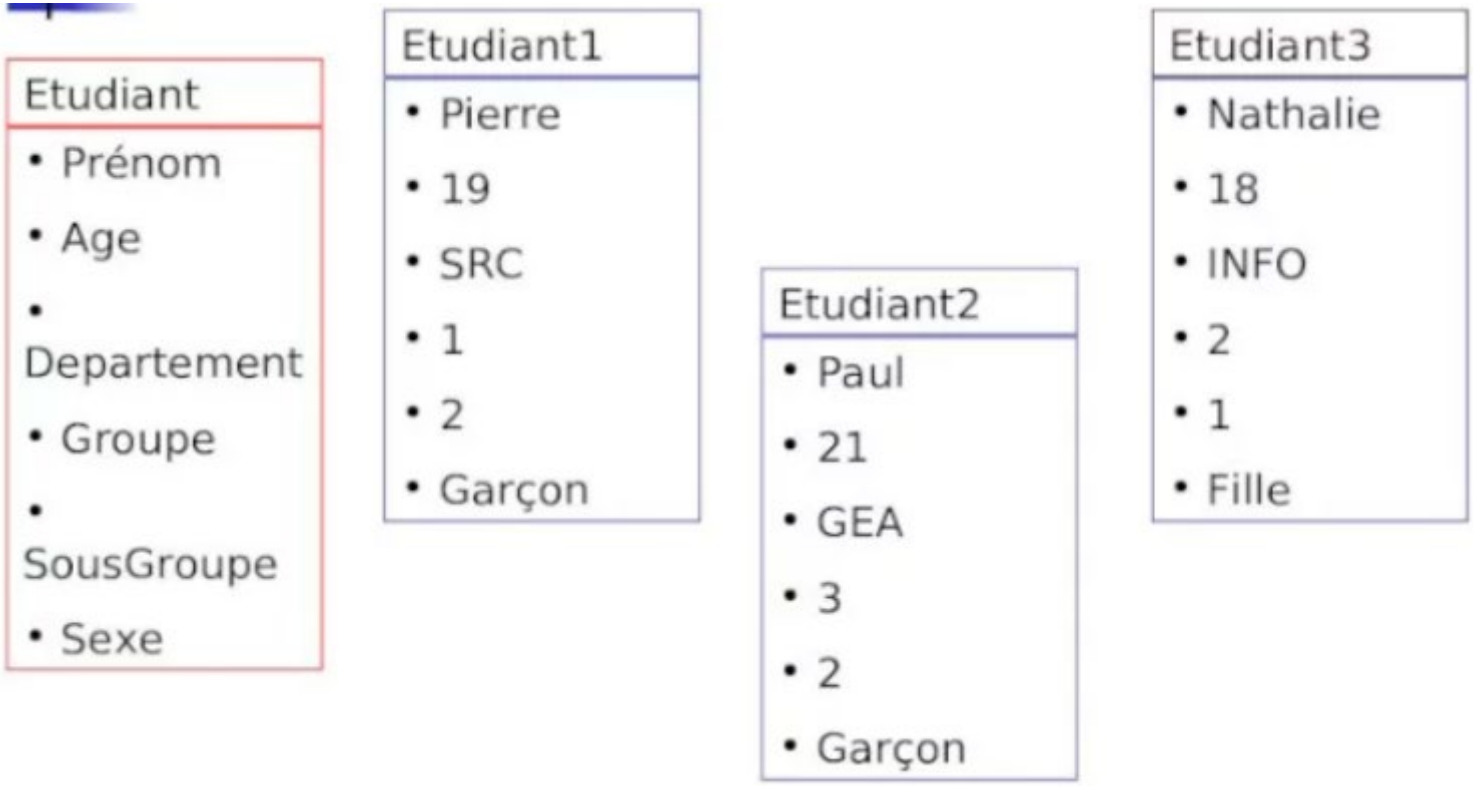
Terre
• Rocheuse • 150000000 • 12756,28

Saturne
• Gazeuse • 1426725400 • 120536

Jupiter
• Gazeuse • 779 millions • 142984

Mars
• Rocheuse • 227936640 • 6794

Exemples d'objet



Encapsulation des données

- L'accès aux données des objets est réglementé
  - Données privées → accès uniquement par les fonctions membres
  - Données publiques → accès direct par l'instance de l'objet
- Conséquences
  - Un objet n'est vu que par ses spécifications
  - Une modification interne est sans effet pour le fonctionnement général du programme
  - Meilleure réutilisation de l'objet
- Exemple de la résistance
  - Les données R et P ne peuvent être utilisées que par les fonctions membres

Polymorphisme

- Un nom (de fonction, d'opérateur) peut être associé à plusieurs mais différentes utilisations
- Exemple
  - Si **a** est un nombre complexe, sqrt(**a**) appellera si elle existe la fonction adaptée au type de **a**.
  - Dans un langage non OO, il aurait fallu connaître le nom de deux fonctions distinctes (selon que **a** soit complexe ou réel)
  - C'est le système qui choisit selon le type de l'argument ou de l'opérande

P.O.O : Concepts

Polymorphisme

```
import math


class Forme:
    def aire(self):
        raise NotImplementedError()


class Carre(Forme):
    def __init__(self, cote):
        self.cote = cote

    def aire(self):
        return self.cote * self.cote


class Cercle(Forme):
    def __init__(self, rayon):
        self.rayon = rayon

    def aire(self):
        return math.pi * self.rayon * self.rayon
```

P.O.O : Concepts

Héritage

- Permet de définir les bases d'un nouvel objet à partir d'un objet existant
- Le nouvel objet hérite des propriétés de l'ancêtre et peut recevoir de nouvelles fonctionnalités

Avantages

- Meilleures réutilisations des réalisations antérieures parfaitement au point

P.O.O : Concepts

Instance d'une classe

- C'est un objet initialisé à partir de la description figée d'une classe

Fonctions membres publiques

- Les constructeurs
- Les destructeurs
- Les accesseurs
- Les modificateurs

Les Constructeurs

- Permettent d'initialiser l'objets lors de sa création
  - copie des arguments vers les données membres
  - initialisation de variables dynamiques à la création de l'objet
- Sont appelés de manière automatique à la création de l'objet
- Peuvent être surchargés → On peut en définir plusieurs de même nom mais acceptant des arguments différents
- Possèdent le même nom que la classe

Le Destructeur

- Il est unique
- Est appelé automatiquement lors de la destruction de l'objet
- Sert généralement à éliminer les variables dynamiques de l'objet (créées en général par le constructeur)
- Il a pour nom le nom de la classe précédé du symbole ~

P.O.O : Concepts

Les Accesseurs

- Catégorie de fonctions membres qui permettent l'accès et l'utilisation des informations privées contenues dans l'objet

Les Modificateurs

- Permettent de modifier l'état des données internes (publiques ou privées) de l'objet

Définition d'une Classe

```
class nom_classe
{
    private:
      Déclaration des données privées et des fonctions membres
    privées
    public:
      Déclaration des données publiques et des fonctions
    membres publiques
};
```

Exemple de définition d'une classe

C++

```
class Point{
private:
  int x, y;

public:
  Point();

  void set(int x0, int y0);
  int getx();
  int gety();

  void move(int dx, int dy);

  void print();
};
```

Exemple de définition d'une classe

Python

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

    def deplace(self, dx, dy):
        self.x = self.x + dx
        self.y = self.y + dy
```

Merci pour votre attention