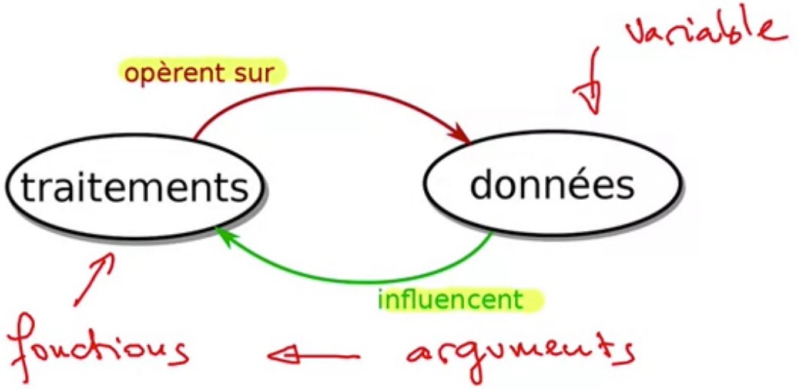
La programmation orientée objet

Procédurale vs POO

Programmation procédurale:

- Fondée sur la définition de fonctions et procédures
- Paradigme de programmation utilisée jusqu'à présent
  - Variables, boucles, instructions conditionnelles, fonctions...
- Limitations: rigidité, difficulté de modification pour grands projets.



Procédurale vs POO

Exemple : écrire une fonction pour calculer le périmètre d'un carré.

```
def perimeter(side):
    return side * 4

# calculer le périmètre d'un carré de côté 3
print(perimeter(3))
```

Procédurale vs POO

Maintenant, on veut pouvoir traiter des carrés et des cercles. Que faire ?

```
def perimeter_square(side):
    return side * 4

def perimeter_circle(radius):
    return radius * 2 * 3.1415

figure = input("Cercle ou carré ? ")
dimension = input("Insérer rayon ou côté ")
if figure == "cercle":
    print(perimeter_circle(dimension))
else:
    print(perimeter_square(dimension))
```

Procédurale vs POO

Et si on augmente les formes géométriques à traiter ?

```
if figure == "cercle":
    print(perimeter_circle(r))
elif figure == "carré":
    print(perimeter_square(a))
elif figure == "triangle équilatéral":
    print(perimeter_triangle(a))
elif figure == "rectangle":
    print(perimeter_rectangle(a,b))
elif figure == "penthagone":
    ...
```

POO

- Robustesse:
  - Changement
  - Erreurs de manipulation des données
- Modularité
- Lisibilité
- Un des objectifs principaux de la notion d'**objet**: **organiser** des programmes complexes

POO

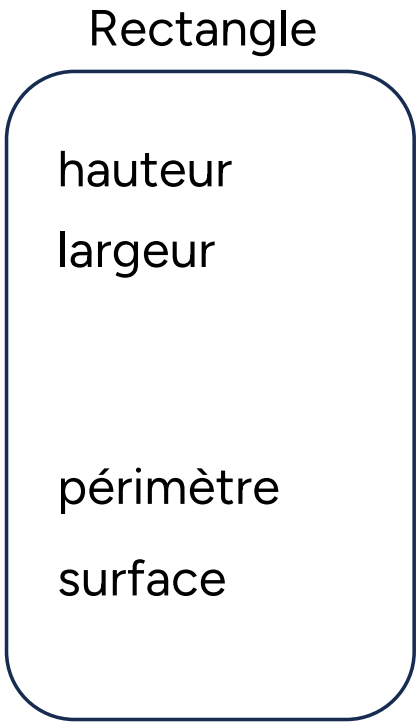
Notions clé :

- Encapsulation
- Abstraction
- Héritage
- Polymorphisme

Encapsulation

Idée: regrouper dans le même objet les donnes et les traitements qui lui sont ***spécifiques***.

- **Attributs** : les données incluses dans un objet
- **Méthodes** : les fonctions (= traitement) définies dans un objet
- Les objets sont définis par leurs attributs et leurs méthodes



Abstraction

- Pour être intéressant, un objet doit permettre un certain degré d'**abstraction**
- Le processus d'abstraction consiste à identifier pour un ensemble d'éléments:
  - des caractéristiques communes à tous les éléments
  - des mécanismes communs à tous les éléments
- Description générique de l'ensemble considéré

Classes

En programmation orientée objet :

- Le résultat des processus d'encapsulation et abstraction s'appelle une **classe**
  - Classe = catégorie d'objets
- Une classe définit un **type**
- Une réalisation particulière d'une classe s'appelle une **instance**
  - Instance = **objet**
- Un objet est une **variable**

```
class Rectangle:
    # on verra tout  bientôt
    que mettre ici...
    pass

rect1 = Rectangle()
rect2 = Rectangle()

print(type(rect1))

>>
<class '__main__.Rectangle'>
```

Classes, attributs et méthodes

En Python, une classe se déclare par le mot clé `class`.

- Exemple: `class Rectangle:`

Ceci définit un nouveau type.

La déclaration d'une instance d'une classe se fait de façon similaire à la déclaration d'une variable:

```
nomInstance = nomClasse()
```

Exemple:

```
rect1 = Rectangle()
```

Déclare une instance `rect1` de la classe `Rectangle`.

Sources

- https://www.coursera.org/learn/programmation-orientee-objet-java/
- J. Sam, J.-C. Chappelier et V. Lepetit - EPFL