CONTRIBUTEURS

Introduction

Étapes de développement

1. L'analyse
   - Quel est le *problème* ?
   - Que veut l'*utilisateur* ?
   - Quel est son *budget* ?
   - Quelles sont les **conséquences** d'une *erreur* ?
2. La conception
   - Comment résoudre le problème ?
   - Quelles sont les *structures de données* appropriées ?
   - Quels sont les *algorithmes* nécessaires ?
   - Quelles sont les *interfaces* requises ?
3. La programmation
   - Implantation de la solution développée aux étapes précédente, en utilisant un ou plusieurs langages de programmation.
4. Les tests d'intégration
   - L'intégration des différents modules en un tout cohérent ;
   - Les procédures de tests qui permettent d'établir la validité et la fiabilité du logiciel.

L'approche hiérarchique pour traiter des données

- les **programmes** sont composés de **modules** ;
- les **modules** contiennent des **énoncés** ;
- les **énoncés** contiennent des **expressions** ;
- les **expressions** créent et manipulent les **données**.

Syntaxe de base

Affectation énoncé (e.g. `pi = 3.1415`) ayant habituellement 3 éléments :

1. un nom de variable (e.g. `pi`) appelé **identifieur** ;
2. l'opérateur (e.g. =) ;
3. une **valeur** affectée à la variable (e.g. 3.1415) appelée **expression**.

Commentaires Des commentaires sont des lignes du code, qui commencent par #, qui ne sont pas exécutées.

Opérateurs arithmétiques

- `+` addition ;
- `-` soustraction ;
- `*` multiplication ;
- `/` division régulière ;
- `//` division entière ;
- `%` reste de la division entière ;
- `**` exponentiation.

Fonction `int` retourne la partie entière d'un nombre. 3 types de nombres : entiers, flottants et complexes. Lorsque l'on effectue une opération arithmétique entre certains nombres, Python conserve le type le plus général. de même pour `float`
Module de maths a plus d'opérateur (e.g. sin, cos, sqrt) et est importé avec `import math`. Puis, on utilise ses fonctions avec, p. ex., `math.sqrt()`.

Fonctions de base

`print`

Permet d'afficher à la console la valeur d'une ou de plusieurs expressions.

- permet aussi avec toute sorte d'options de spécifier la façon dont cette ou ces valeurs seront affichées.

`input`

Permet de lire ce que vous entrez au clavier et retourne le résultat sous la forme d'une chaîne de caractères.

Fonctions

Introduction

Qualités d'une fonction

1 Cohérence

Une fonction est ***cohérente*** si elle accomplit une seule tâche. On doit pouvoir résumer en peu de mots ce qu'accomplit la fonction.

2 Indépendance

Une fonction est ***indépendante*** si sa sortie dépend uniquement de ses entrées (arguments) et d'aucune autre variable. Il ne faut pas définir des fonctions qui dépendent de *variables globales*.

3 Concision

La *concision* consiste à limiter la longueur des fonctions. Plus la fonction est courte, plus elle sera facile à comprendre pour un humain.

Définition

De façon générale, on définit une fonction en Python avec `def` :

```
def name(arg1, arg2, ..., argn):
        #       bloc d'énoncés indentés
        return expression       #       optionnel
```

Booléens

Les opérateurs suivants permettent de comparer les valeurs respectives de deux objets :

$<$ inférieur ;

$>$ supérieur ;

$<=$ inférieur ou égal ;

$>=$ supérieur ou égal ;

$==$ égal ;

$!=$ pas égal.

Les opérateurs suivants permettent de combiner une ou plusieurs expressions booléennes :

and conjonction ($\cap$)

or disjonction ($\cup$)

not négation ($A^{\complement}$)