

SUPINFO Academic Dept.

**Object Oriented Programming in Python**

Séances LABS

Activités pratiques

Version 1.0

Last update: 28/10/2014

Use: Students/Staff

Author: Laurent GODEFROY

SOMMAIRE

1 Préambule 3

1.1 Objectifs de ces séances Labs 3

1.2 Remarques sur ces séances et exercices 3

2 Familiarisation avec le Python 3

2.1 Suite de Fibonacci 3

2.2 Suite de Syracuse 4

2.3 Recherche d’un élément dans une liste 4

2.4 Calcul du minimum d’une liste 4

2.5 Recherche d’un élément dans une liste (bis) 4

2.6 Distance de Hamming 5

2.7 Suite de Conway 5

3 Généralités sur la bataille navale 5

3.1 Conception des classes 5

3.2 Où l’on commence à coder 8

4 où l’on va jouer au jeu « classique » 8

4.1 Autour de la classe Bateau 8

4.2 La classe gérant la grille d’un joueur 8

4.3 Fin du codage 8

5 Où l’on apporte quelques nouvelles fonctionnalités 8

5.1 Surcharge de l’opérateur < 9

5.2 Déplacement des bateaux 9

5.3 Le cas particulier du sous-marin 9

5.4 Laissez libre cours à votre imagination 9

# Préambule

## Objectifs de ces séances Labs

Les activités pratiques de la matière 2OOP se déroulent en **4 séances**. Chaque séance correspond à une partie de ce présent document.

A l’issue de ces séances, un étudiant devra être capable de manipuler tous les concepts fondamentaux de la programmation orientée objet, et ce en utilisant le langage Python :

* Déclarer une classe et instancier des objets.
* Respecter le principe d’encapsulation.
* Implémenter une relation d’héritage.
* Implémenter une relation de composition.
* Savoir surcharger les opérateurs usuels.
* Utiliser le polymorphisme.

## Remarques sur ces séances et exercices

Ces exercices se prolongent d’une séance à l’autre, il est donc nécessaire de conserver ses codes, et éventuellement de les finir chez soi avant la séance suivante.

# Familiarisation avec le Python

See the 1ADS Labs document.

# Battleship introduction

## Classes design

We will throughout these sessions develop a battleship game. We will follow the following rules: <http://en.wikipedia.org/wiki/Battleship_(game)>

<https://www.novelgames.com/zh-HK/battleship/>

The game will be played by 2 human player. Each time a player will played, we will display to him 2 board of 10x10 boxes (2D table).

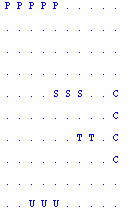
* The first containing information about his own ships and enemy fire (Plonk in water or hit).
* The second containing information on his shots (Plonk in the water, hit or sunk).

These two board will not own the same type of information.

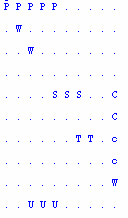
The fleet will consist for each player :

* an aircraft carrier (5 boxes) (P in French, A in English)
* a Battleship (4 boxes) (C in French, B in English)
* a Destroyer (3 boxes) (U in French, D in English)
* a sub-marine (3 boxes) (S in French and English)
* a patrol boat (2 boxes) (T in French, P in English)

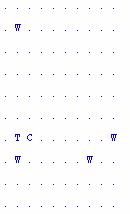
Here is one of the first board display example at the beginning of the game (one letter for each type of boat):



Then after a few rounds (lowercase means that the boat in question has been hit and the 'W' means a shot into the water of the opponent):



Here is an example of the opponent board ("W" for "water", "T" for "hit and" C "for" sunk "):



We will make sure you can easily change the number of boxes of each of these boats.

Each boat can be positioned horizontally or vertically. Two boats will occupy adjacent squares.

**Submit (but not yet implement) a set of classes in order to play this game. It will also consider their attributes and methods.**

## Let’s CODE!

Code skeletons classes and all methods not affected by an inheritance relationship.

We constantly respect the principle of encapsulation (private attributes, public methods)

# Let’s play the classical game

## About the “boat” class

Implement inheritance relationships between the "boat" class and its subclasses.

## Class managing the player’s board

Code this class taking care to ask the player successively the position of each of its boat in its constructor.

Of course we will ensure that two boats can not overlap.

## Finish Coding the game

Complete the code and test that the game is properly running.