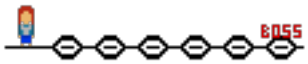
Initiation à la programmation

Bastien Gorissen & Thomas Stassin

Année 2016

Chapitre 1

Level 1



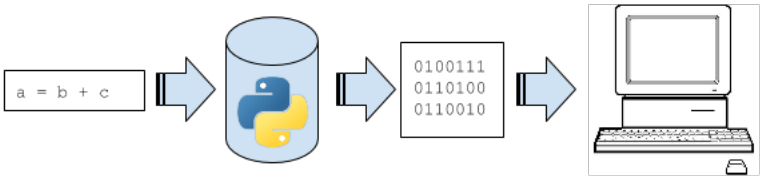
1.1 Qu'est-ce qu'un langage de programmation ?

> Un langage de programmation est une convention pour donner des ordres à un ordinateur. Ce n'est pas censé être obscur, bizarre et plein de pièges subtils. Ca, ce sont les caractéristiques de la magie.
>
> *Dave Small*

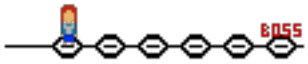
Un langage de programmation est un moyen d'intéragir avec l'ordinateur afin de lui donner des instructions, le langage qui est *intelligible*, sera *interprété* afin d'être compris par la machine.

1.1.1 Commentaires

Si vous ouvrez le script `run_game.py`, vous pourrez voir que certaines lignes commencent par `#`. Celles-ci seront ignorées par l'ordinateur. Ce sont des *commentaires* destinés à clarifier le code.



1.1.2 Stage 1

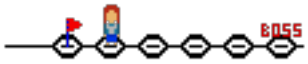


Exécuter le script se trouvant dans le dossier `lv1/run_game.py`

1.2 Commander à l'ordinateur

Pour "commander" l'ordinateur, on lui donne des *instructions*, le plus souvent, ces instructions seront regroupées dans un *script*.

1.2.1 Stage 2



Dans l'exercice précédent, nous avons exécuté le script `run_game.py`, ce script est rempli d'instructions, dont celles-ci :

```
dungeon_size = (5, 5)
game = world.Game(dungeon_size)
game.run()
```

Que ce passerait-il si on modifiait l'instruction qui définit la grandeur du donjon dans le script ? Changez les données de dimension du donjon dans le script et observez ce qu'il se passe lors de l'exécution du script.

1.2.2 À retenir

Python est un *langage sensible à la casse*[1], ce qui veut dire qu'il fait la différence entre les majuscules et les minuscules. Autrement dit `A` sera différent de `a` et `world` différent de `World`.

Donc si je remplace `world` par `World` dans le script précédent, il générera une erreur.

```
dungeon_size = (5, 5)
game = World.Game(dungeon_size)
game.run()
```

Vous devriez obtenir un message ressemblant à "`NameError: name 'World' is not defined`"

1.3 Les instructions de sortie

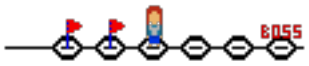
Il existe plusieurs sortes d'instructions, l'une d'elles sont les instructions de sortie. Une instruction de sortie envoie vers une "sortie" ce qu'on lui donne. Une des sorties les plus couramment utilisée est la console et avec Python, l'instruction de sortie vers la console est `print`.

Et donc voici le classique, mais indémodable "Hello World" en Python :

```
print("Hello World!")
```

1. *Case sensitive* en anglais.

1.3.1 Stage 3



Lorsque le donjon est créé, signalez-le par un message dans la *console*.

1.4 Les variables

Souvent, il sera utile de stocker certaines valeurs tout au long de l'exécution de votre code. Par exemple pour stocker la valeur d'un calcul, ou même afin de réutiliser ses valeurs plusieurs fois. Pour stocker des valeurs, on utilise des *variables*.

La *variable* est un moyen de stocker une valeur quelconque (un nombre, du texte, voire même des *objets* plus complexes) dans la mémoire du programme.

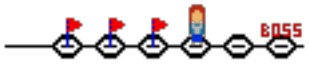
Dans le cas de notre jeu, le donjon, ainsi que le héro sont contenus chacun dans une *variable*.

```
dungeon_size = (5, 5)
game = world.Game(dungeon_size)
hero = Game.hero
```

En Python, l'*affectation* d'une valeur se fait avec l'opérateur =.

Dans `a=3`, on affecte `3` à la variable `a`[2]

1.4.1 Stage 4



Faites en sorte de stocker les dimensions du donjon dans des variables (par exemple `length` et `width`[3]).

2. En Python il suffit d'affecter une valeur à une variable pour qu'elle commence à exister, ce n'est pas vrai pour la plupart des langages (comme le C# par exemple).

3. Il est courant d'utiliser des noms de variables en anglais. La programmation est un monde très anglophone.

1.5 Opérations sur une chaîne de caractères

Il vous sera parfois utile de savoir manipuler des chaînes de caractères, l'exemple classique est dans une ligne de dialogue, où l'on voudrait dire au héros le nombre d'ennemis qu'il lui reste à tuer avant de finir la quête. Or, lorsque que vous codez, vous ne connaissez pas le nombre d'ennemis. Vous avez sûrement stocké cette information dans une variable et donc vous allez devoir intégrer cette variable à votre ligne de dialogue.

1.5.1 Conversion

Il y un moyen facile de convertir une variable en chaine de caractères[4], il suffit d'utiliser la fonction `str`.

Si je voulais convertir la variable `nbr_enemies` qui contient le chiffre `3` je procéderais comme suit :

```
nbr_enemies = 3
nbr_enemies = str(nbr_enemies)
```

A la fin de l'exécution de ce petit script, `nbr_enemies` ne vaut plus `3` mais `"3"`.

1.5.2 Concaténation

Le fait de coller deux chaînes de caractères l'une derrière l'autre porte un nom : *la concaténation*. En Python, l'opérateur pour *concaténer* deux variables ensemble est l'opérateur `+`.

Par exemple, si je voulais concaténer la variable contenant le nom du héro avec un message cela donnerait :

```
message = ", il te reste des ennemis    tuer."
message = hero_name + message
```

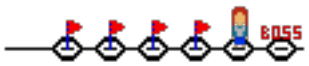
Donc si le héro se nomme Brutor, la chaîne de caractères en fin de script sera égale à `"Brutor, il te reste des ennemis à tuer."`

4. En Python les chaines de caractères ont le *type* `str` pour le mot anglais *string*, qui veut dire chaîne

On peut même aller plus loin en mêlant la *conversion* et la *concaténation* en indiquant aussi dans le message le nombre d'ennemis qu'il reste.

```
    message = ", il te reste " + str(nbr_enemies) + " ennemis     tuer."
    message = hero_name + message
```

1.5.3 Stage 5



Lorsque le donjon est créé, indiquer dans un message à la console la longueur et la largeur de celui-ci.

1.6 Les fonctions et les méthodes, première approche

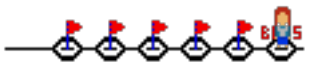
Pour terminer ce premier level, on va mettre des mots sur deux choses que l'on a vues précédemment. Les *fonctions globales* et les *méthodes*. Nous avons déjà vu plusieurs fonctions dans les parties précédentes :

- `print`
- `str`

Ces fonctions sont dîtes "globales". Ce type de fonctions n'est pas lié à un type d'*objet*.

Il existe aussi une série de fonctions liées à des *objets*, par exemple la fonction *run* de la variable *game*. On appelle ces fonctions des *méthodes* et elles sont liées à un type d'objets en particulier.

1.6.1 Stage 6



La variable du héros, sobrement appelée `hero` est du type `Hero`, c'est un type de variable que nous avons construit pour le jeu [5]. Nous avons donné trois *méthodes* à cette variable :

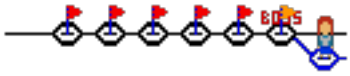
`turn_left :` ce qui fait tourner le héros à gauche.

`turn_right :` ce qui fait tourner le héros à droite.

`move :` ce qui fait avancer le héros d'une "case".

Essayez de faire bouger le héros grâce à ces trois méthodes.

1.7 Bonus Stage



Pour compléter le stage bonus il faut que vous arriviez à amener le héros sur la sortie du donjon. La sortie est représentée par la case bleue en haut à gauche de la pièce. Le donjon doit au moins avoir une longueur et une largeur de 7.

5. Nous verrons plus tard qu'il est possible de définir ses propres types et de leur donner les caractéristiques que l'on désire.