

DBSF - Préformation

Algorithme

Bastien Gorissen & Thomas Stassin

PROLOGUE

UN LANGAGE DE PROGRAMMATION, VOUS DITES ?

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

PYTHON

“Où l’on abandonne enfin le papier”

POUR FAIRE DU PYTHON, ON A BESOIN DE QUOI ?

Principalement de 3 choses :

1. Python
2. Un moyen d'écrire du code (éditeur)
3. Un moyen de lancer son code

Pour avoir tout cela d'un coup, je vous propose d'utiliser [Mu](#).



hello.py

```
1 print("Hello from Mu!")  
2
```

Running: hello.py

```
Hello from Mu!  
>>>
```

Saved file: /home/ntoll/mu_code/hello.py

Python

PYTHON SANS MU?

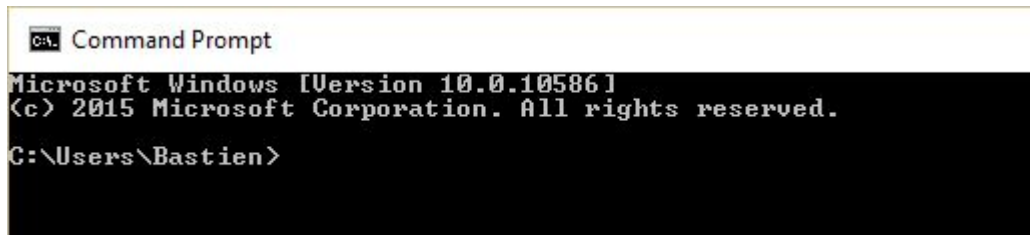
Les slides qui suivent expliquent comment installer et utiliser Python en ce passant de Mu. Nous ne le verrons pas pendant le cours, mais sentez-vous libre de le faire chez vous, et si vous avez des questions, nous sommes bien-sûr là pour y répondre.

ET DONC, COMMENT ON LANCE SON CODE ?

Nous allons utiliser “l’invite de commande” de Windows.

C’est un poil barbare, mais on s’y fait.

Pour l’ouvrir, tapez “cmd”, et là...

A screenshot of the Windows Command Prompt window. The title bar at the top says "Command Prompt". The main area is black with white text. It shows the Windows version "Microsoft Windows [Version 10.0.10586]", the copyright notice "(c) 2015 Microsoft Corporation. All rights reserved.", and the current directory "C:\Users\Bastien>".

```
Command Prompt
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Users\Bastien>
```


...

Oui, ok, il faut un peu s'accrocher, mais on a besoin de ça pour exécuter les scripts que nous allons écrire.

Nous allons avoir besoin de 4 commandes, principalement :

1. cd
2. dir
3. python
4. pip

CD COMME...

cd est l'abréviation de “*change directory*” (ou “changer de dossier”), et permet de changer de dossier.

Par exemple, si je suis dans le dossier C:\MonDossier, la commande suivante :

```
C:\MonDossier>cd superJeu
```

m'amènera dans le dossier

```
C:\MonDossier\superJeu>
```

CD, DEUXIÈME SERVICE

Deux cas particuliers...

Si je veux “remonter” d’un dossier :

cd ..

Et si je veux changer de disque (par exemple passer de C: à D:) :

d:

Ca va jusque là ?

DIR COMME DIRECTORY ?

Exact, la commande “dir” permet de lister le contenu d’un dossier, utile quand on veut vérifier que notre fichier est bien dans le dossier actuel.

Voyons un exemple...

```
C:\>d:
```

```
D:\>cd Gamedev
```

```
D:\Gamedev>dir
```

```
Volume in drive D is Yggdrasil  
Volume Serial Number is 1E3E-BF80
```

```
Directory of D:\Gamedev
```

```
12/12/2015  09:25    <DIR>          .  
12/12/2015  09:25    <DIR>          ..  
11/11/2015  16:11             4,897 $hatchingHLSL.shader  
11/11/2015  15:55          360,952 archimesh_v1_1_1.zip  
04/09/2015  21:09          791,244 bedroom.blend  
16/05/2015  07:05    <DIR>          blop  
03/06/2015  23:06    <DIR>          Builds  
15/05/2015  07:38          578,996 chevet.blend  
25/11/2015  19:52    <DIR>          Domiverse  
11/11/2015  15:42    <DIR>          gws-test-scene  
07/04/2015  12:03    <DIR>          gws-test-scene-test_scene_basic  
07/04/2015  18:34          28,719,226 gws-test-scene-test_scene_basic.zip  
25/11/2015  21:52    <DIR>          Interface3  
27/10/2015  19:27    <DIR>          intro2programming  
15/05/2015  07:22    <DIR>          ld27  
12/12/2015  10:22    <DIR>          ld34  
07/04/2015  19:39    <DIR>          Sandbox  
04/11/2015  22:42    <DIR>          voxels  
                5 File(s)          30,455,315 bytes  
                13 Dir(s)  919,278,841,856 bytes free
```

```
D:\Gamedev>
```

ET PYTHON DANS TOUT ÇA ?

Pour Python, nous allons avoir deux commandes.

pip install cocos2d

...qui installe le petit moteur de jeu dont nous aurons besoin (à ne faire qu'une fois).

python monscript.py

Qui lance le script monscript.py, si celui-ci se trouve dans le dossier actuel.

C'EST TOUT ?

Oui, c'est tout !

A vous de jouer !

Créez un script `helloworld.py` vide
et lancez-le avec la console!

UNE VARIABLE, C'EST QUOI ?

Parfois (souvent), il est nécessaire ou utile de garder une donnée en mémoire pour la réutiliser.

Ex.:

- Les points de vie du héros
- Le nom d'un personnage
- Le nombre d'ennemis à battre
- ...

Pour ce faire, on utilise des “*variables*”.

ET PYTHON DANS TOUT ÇA ?

Pour Python, nous allons avoir deux commandes.

pip install cocos2d

...qui installe le petit moteur de jeu dont nous aurons besoin (à ne faire qu'une fois).

python monscript.py

Qui lance le script monscript.py, si celui-ci se trouve dans le dossier actuel.

C'EST TOUT ?

Oui, c'est tout !

A vous de jouer !

Créez un script `helloworld.py` vide
et lancez-le avec la console!

LEVEL 1-2

“Spam 101.”

EST-CE QU'ON PEUT ENVOYER SES PROPRES MESSAGES ?

Il existe plusieurs façons de faire “sortir” un message depuis le script Python vers le monde extérieur.

Ce sont des “*instructions de sortie*”.

La plus courante est **print()**, qui permet d’écrire dans la console.

Ex. :

```
print(“Hello World!”)
```

ET DONC ON PEUT ÉCRIRE CE QU'ON VEUT ?

Oui, oui ! D'ailleurs...

A vous de jouer !

Utilisez **print()** pour écrire un message dans la console (ce que vous voulez).

LEVEL 1-3

“Prévision : temps variable.”

UNE VARIABLE, C'EST QUOI ?

Parfois (souvent), il est nécessaire ou utile de garder une donnée en mémoire pour la réutiliser.

Ex.:

- Le résultat d'un calcul
- Le nom d'un utilisateur
- Le nombre de fichiers ouverts
- ...

Pour ce faire, on utilise des “*variables*”.

C'EST JUSTE UNE VALEUR ?

Techniquement, c'est un peu plus compliqué que ça, mais en pratique, on peut dire ça.

C'est un nom qu'on donne à une valeur, qui peut évoluer dans le temps.

En Python, on crée une variable en utilisant "l'affectation", notée =.

Ex.: **a = 3** # on stocke la valeur **3** dans la variable **a**

ON PEUT METTRE CE QU'ON VEUT DEDANS ?

Plus ou moins. Voici plusieurs exemples :

```
address = "https://api.myservice.com/"
```

```
connection = HTTPConnection(address)
```

```
results = connection.fetch(num_results=1000)
```

Une variable peut référencer une autre variable, une donnée complexe, un simple nombre...

UN AUTRE EXEMPLE ?

a = 3

b = 4

c = a + b

a = c + 1

print(c)

print(a)

Résultat ? a -> 8 b -> 4 c -> 7

OK, VOYONS VOIR...

A vous de jouer !

Stockez votre âge dans une variable,
et le double de votre âge dans une
autre, et affichez les deux.

LEVEL 1-5

“Bel enchaînement.”

EST-CE QU'ON PEUT METTRE DU TEXTE DANS UNE VARIABLE ?

On peut, et c'est un cas assez courant.

En informatique, on appelle un bout de texte une “*chaîne de caractères*” ou “*character string*” (ou “*string*”) en anglais.

Ex.:

```
message = “Ceci est mon message.”
```

La variable **message** contient donc le texte “**Ceci est mon message.**” Plutôt logique, non ?

QUEL AVANTAGE ?

On peut utiliser des “*string*” pour manipuler du texte dynamiquement.

Par exemple, prenons le petit script suivant :

```
hero_name = “Brutor”  
message = hero_name + “, il reste des monstres à vaincre.”  
print(message)
```

Devinez quel sera le résultat...

On appelle ceci de la “*concaténation*” de chaîne.

PEUT-ON CONVERTIR QUELQUE CHOSE EN STRING ?

Il existe une fonction en Python qui permet la “*conversion*” de données en texte.

C’est **str()**.

Ex. :

```
a = 3
```

```
b = str(a)
```

a contient le nombre **3**, et **b** contient le texte “**3**”.

ET TIENS, SI ON COMBINAIT...

A vous de jouer !

Stockez votre nom dans une variable,
votre âge dans une autre et affichez
une phrase contenant ces deux
variables.

LEVEL 1-6

“Toc toc!” “Entrée!”

ENTRÉE PRINCIPALE

Parfois, vous ne voudrez pas seulement afficher, mais aussi récupérer des données.

Pour se faire, on utilise une instruction dite d'entrée.

La plus courante se nomme **input()**.

Elle s'utilise comme suit:

```
hero_name = input("Quel est ton nom? ")
```

ENTRÉE PRINCIPALE

```
hero_name = input("Quel est ton nom? ")
```

input va réaliser plusieurs choses:

- Afficher la chaîne de caractères "Quel est ton nom? "
- Attendre que l'utilisateur écrive quelque chose au clavier, et confirme en appuyant sur Enter.
- Renvoyer ce que l'utilisateur a écrit.

Dans la ligne ci-dessus, la valeur retournée est ensuite mise dans la variable **hero_name**.

ENTRÉE VIP

Attention: ce que **input()** vous renvoie est toujours de type **string**.

Si vous avez besoin d'autre chose, vous devrez procéder à une conversion.

On appelle cette conversion un “casting”.

LEVEL 1-7

“Le cas Sting”

CASTING

Le “casting” est le processus de conversion d’un type de données vers un autre type.

Il s’agit d’une généralisation de ce que vous avez vu plus haut avec la fonction **str()**.

Par exemple, pour convertir une chaîne de caractères en entier :

```
a = “11”
```

```
a = int(a)
```

CASTING

Il suffit donc d'utiliser la fonction portant le nom du type vers lequel vous voulez convertir vos données.

Les types de bases de Python sont les suivants:

- **int** pour les nombres entiers.
- **float** pour les nombres à virgule.
- **str** pour les chaînes de caractères.
- **bool** pour les variables booléennes (voir plus loin).

CASTING

Évidemment, le casting ne fonctionnera que si la conversion a du sens.

`int("waaaaah")` n'aurait pas vraiment de valeur facile à définir...

A vous de jouer !

Récupérez l'âge de l'utilisateur au clavier,
ensuite affichez le double de son âge.

LEVEL 1-9

“Comparaison n’est pas raison.”

COMMENT FAIRE INTERAGIR 2 VARIABLES ?

En python, il existe deux fonctions calculer le minimum et le maximum entre deux valeurs : **min()** et **max()**. Derrière ces 2 fonctions il y a une notion commune : la comparaison.

Si je fais : **min(a, 3)**

Je me pose la question : “Est-ce que a est plus petit que 3 ?”

En d’autres termes : **a < 3 ?**

Ceci est une comparaison.

COMMENT RÉCUPÉRER LE RÉSULTAT D'UNE COMPARAISON ?

Comme pour tout, on peut mettre le résultat d'une comparaison dans une variable.

Ex. :

```
a = 3  
comp = a < 10  
print(comp)
```

Et que se passe-t-il si on change la valeur de a par 42 ?

TRUE ET FALSE ? KÉZAKO ?

Souvent, en Python, la réponse se trouve dans la traduction des mots utilisés.

True <-> Vrai

False <-> Faux

Python a un type (comme `int`, `str`, ...) qui correspond à une valeur vraie ou fausse, c'est le type **bool**, ou booléen.

Nous allons voir comment nous en servir.

ON PEUT FAIRE QUOI COMME TYPE DE COMPARAISON ?

Excellente question ! Il existe un nombre varié de comparaisons.

- $<$, ou “plus petit que”
- $<=$, ou “plus petit ou égal à”
- $>$, ou “plus grand que”
- $>=$, ou “plus grand ou égal à”
- $==$, ou “égal à”
- $!=$, ou “différent de”

Encore un truc de matheux :p

AUTANT ESSAYER...

A vous de jouer !

Essayez les différents opérateurs de
comparaison dans l'invite de
commande de Python.

EST-CE QU'IL Y A DES PIÈGES ?

On peut dire ça...

`4 < 4` par rapport à `4 <= 4`

`“message” < “texte”`

`3 > “texte”`

`==` qui est différent de `=` (affectation)

Mais dans l'ensemble ça fait ce qui marqué sur l'étiquette...

LEVEL 1-10

“Swimming bool.”

QU'EST-CE QU'ON FAIT AVEC DES TRUE/FALSE ?

Comme nous l'avons vu, Python a donc un type qui correspond à vrai/faux.

C'est une notion qui vient du monde de la logique.

Comme pour les nombres ou le texte, il existe toute une série d'opérations qu'on peut effectuer spécifiquement sur des variables ou des valeurs de ce type.

Ca porte le nom barbare d'"Algèbre Booléenne".

LE NOM N'INSPIRE PAS CONFIANCE...

Il s'agit surtout de “codifier” une série de notions qu'on rencontre finalement tout le temps dans la vraie vie.

Par exemple :

“Je veux acheter une voiture **OU** acheter un abonnement de train.”

“J'ai envie de partir loin **ET** au soleil.”

“Je n'ai **PAS** de poisson rouge.”

ÇA SEMBLE DE FAIT LOGIQUE.

Comme tout bon mathématicien, George Boole a bien évidemment trouvé tout un tas de règles et de théorèmes spécifiques à ce qui est une véritable branche des mathématiques.

Mais...

Nous allons nous concentrer sur trois choses simples qu'on rencontre tout le temps en programmation :

Le **NOT**, le **AND** et le **OR**.

NOT ?

Le **NOT**, c'est la négation. Comme dans une phrase, quand on fait une négation, on inverse les valeurs de vrai ou faux.

Il y a donc deux cas :

not True # -> ceci est équivalent à False

not False # -> ceci est équivalent à True

Une idée de ce que donne **not (3 > 10)** ?

AND ?

Le **AND**, c'est la conjonction. Ou plus sobrement, le “et”. Ça donnera **True** seulement si 2 conditions sont toutes les deux vraies.

True and True # -> True

False and True # -> False

True and False # -> False

False and False # -> False

OR ?

Le **OR**, c'est la disjonction. Ou encore, le "où". Ca donnera **True** si au moins une des 2 conditions est vraie.

True or True # -> True

False or True # -> True

True or False # -> True

False or False # -> False

DES EXEMPLES ?

a = 3

b = 5

(a == b) or (a < b)

(b >= a) and (a != b)

(not (a > 0)) and (b == 5)

not(a == 3 or b == 5)

not(a == 3) and not(b == 5)

LEVEL 1-11

“If only...”

QUAND EST-CE QU'ON FAIT UN TRUC UTILE AVEC TOUT ÇA ?

Maintenant !

Vous vous souvenez qu'un de notre problème, c'était que le personnage ne pouvait pas faire de choix "intelligent" ?

Pour introduire la notion de choix, on utilise un instruction dite de "*contrôle de flux*".

Ca revient à dire : "**Si** une certaine condition est remplie, **alors** fait quelque chose."

ET EN PYTHON ?

Le “si”, en Python, ça se dit “**if**” (c’est étonnamment proche de l’anglais, le Python, au final...)

```
a = 3
```

```
if a < 10:
```

```
    print(str(a) + " est plus petit que 10 !")
```

C’est aussi simple que ça !

Attention à deux choses : les “:” et l’indentation du code à l’intérieur du if !

ET AVEC LES BOOLÉTRUCS ?

Un **if**, schématiquement, c'est :

if condition:

 # Un bout de code super top

condition peut être n'importe quelle valeur de type **bool**, ce qui veut dire :

a > 3 and b <= 10 -> OK !

not (name == "Roger") -> OK !

On peut mettre exactement la condition qu'on veut !

ON POURRAIT UTILISER ÇA DANS LE JEU ?

Oui, oui ! D'ailleurs...

A vous de jouer !

Une nouvelle méthode de la variable hero permet de vérifier si la case en face est libre. Utilisez-là pour contourner un bloc !

LEVEL 1-12

“if, version full option.”

UN "SI", ÇA DEMANDE SOUVENT UN "SINON", NON ?

Un **if** en Python peut contenir une clause "sinon", en anglais **else**

if condition:

 # Un bout de code super top

else:

 # Un autre bout de code si condition == False

Attention, c'est soit le "if", soit le "else" qui est exécuté, impossible d'avoir les 2 en même temps !

ÇA ME DONNE UNE IDÉE...

Ca tombe bien!

A vous de jouer !

Utilisez un **else** pour améliorer
votre évitement de bloc.

ET SI ON A PLUSIEURS CONDITIONS À TESTER ?

Une dernière extension du **if** existe si vous avez plusieurs conditions à tester :

```
a = 3
```

```
b = 5
```

```
if a < b:
```

```
    print(str(a) + " plus petit que " + str(b))
```

```
elif a == b:
```

```
    print(str(a) + " est égal à " + str(b))
```

```
else:
```

```
    print(str(a) + " est plus grand que " + str(b))
```


ET SI PLUSIEURS CONDITIONS SONT REMPLIES ?

```
a = 3
b = 5
if a < b:
    print(str(a) + " plus petit que " + str(b))
elif a < 10:
    print(str(a) + " est plus petit que 10")
else:
    print(str(a) + " est plus grand que " + str(b))
```

Python ne va exécuter que le bloc de code correspondant à la première condition remplie !

LEVEL 1-13

“[insert random joke here]”

NOMDRA

Beaucoup de jeux mettent à profit la notion de hasard pour offrir au joueur une expérience toujours renouvelée.

Bien entendu, derrière, ne se cache rien d'autre qu'une instruction spécifique destinée à générer des valeurs aléatoires.

En Python, ces fonctions sont reprises dans la librairie **random**, et nous allons surtout nous intéresser à la fonction **randint()**, qui sert à générer un entier aléatoire.

MARDON

Pour pouvoir utiliser **randint()**, nous allons devoir demander à Python de l'importer.

Pour ce faire, il faut placer en début de script une ligne telle que:

```
from random import randint
```

Ensuite, vous pouvez utiliser **randint()** comme toute autre fonction de Python:

```
d6 = randint(1, 6)
```

RONDMA

```
d6 = randint(1, 6)
```

La fonction **randint()** accepte 2 arguments:

- Un minimum (ici, 1)
- Un maximum (ici, 6)

...et génère un nombre entier compris entre le minimum et le maximum (compris).

Il existe bien d'autres fonctions dans la librairie **random**, mais nous aurons l'occasion de les voir plus tard !

LEVEL 1-14

“ボス・バトル”

BOSS BATTLE !

Cette fois, vous allez partir d'un script vierge, et créer un jeu de devinette simple.

- Le joueur devra deviner un nombre choisi aléatoirement entre 1 et 10.
- Le jeu lui dira après chaque essai si sa proposition était trop haute, trop basse, ou s'il a deviné juste.
- Il aura 3 chances pour découvrir le nombre, après quoi il aura perdu, et le programme lui annoncera la bonne réponse.

HOMework

DEVOIR :)

Créer un script qui:

- Affiche “Bonjour”
- Demande à l'utilisateur “Est-ce que tu vas bien?” et stocke la réponse dans une variable
 - Si la réponse est “oui” ou “Oui”, le programme affiche “Bonne journée”
 - Si la réponse est “non” ou “Non”, le programme une phrase réconfortante