

The background is a pixelated Super Mario Bros. style scene. It features a green grassy field with a blue sky above and a grey brick floor at the bottom. Scattered throughout the grass are several small, pixelated enemies: Goombas (brown and red), Koopas (blue and green), and a Piranha Plant (red and black). A single tree with a green canopy and a brown trunk is located on the right side. The title text is centered in the grass area.

INITIATION À LA PROGRAMMATION EN PYTHON

Bastien Gorissen & Thomas Stassin

PLATS DU JOUR

AU MENU:

- Correction des exercices de révisions
- Les fonctions
- Encore les fonctions
- Toujours les fonctions
- Et plus de fonctions

Vous l'aurez compris, aujourd'hui on voit les fonctions sous toutes les coutures.

LEVEL 5-1

“Fonction” (Warning Blague Star Wars incoming)

FONCTION 101

Une fonction est un outil qui permet d'abstraire une partie du code afin d'augmenter la clarté et la lisibilité de celui-ci.

Nous avons déjà rencontré des fonctions dans les cours précédents: **len**, **print**, **input**, etc.

Nous allons voir comment créer nos propres fonctions

LA RECETTE D'UNE FONCTION

Le mot magique pour définir une fonction est **def**

Ce mot doit être suivi du nom de la fonction, suivi lui-même de parenthèse et de “:”.

```
def mega_function_of_death():
```

```
    # code qui arrache du steak de poney
```

Notez bien que les parenthèses ne resteront pas vides, on verra plus tard comment les remplir.

APPELER UNE FONCTION

Appeler une fonction est bête comme choux.

Il suffit de faire comme suit:

mega_function_of_death()

Encore une fois, c'est similaire à **print**, **len** et compagnie, puisque ce sont aussi des fonctions.



Les noms de vos fonctions doivent **TOUJOURS** être **EXPLICITES**.

Ceci afin qu'on ne doive pas lire l'intégralité du code de la fonction pour connaître son utilité.

Maintenant que ceci est dit...

BINGO, LE RETOUR...

Dans le jeu du bingo , ajoutez une fonction **“bingo”** qui affiche "Bingo!" dans la console.

Dans le code principal du jeu, remplacez la partie qui s'occupe d'écrire "Bingo!" par l'appel de votre fonction

LEVEL 5-2

“Le retour”

```
def the_return_of_the_jedi():  
    the_jedi = Jedi  
    return the_jedi
```

LE RETOUR...

Le retour d'une fonction est la valeur que renvoie la fonction au code appelant.

```
def dice_simulation():  
    dice = randint(1, 6)  
    return dice
```

```
result = dice_simulation()  
print("Résultat du dé: " + result)
```

Ici la fonction renvoie la valeur contenue dans la variable **dice**.

LE RETOUR...

C'est le mot clé **return** qui renvoie la valeur.

Noter bien que le fait de retourner une valeur arrête immédiatement le traitement de la fonction, pour *retourner* à l'appelant.

```
def weird_function():  
    result = randint(1,6)  
    return result  
  
    print("Texte qui ne sera jamais afficher")
```

Dans ce script la dernière ligne ne sera jamais affichée car elle est située juste après le return.

LE BINGO CONTRE-ATTAQUE

Faites en sorte de bouger dans une fonction la partie du code qui demande à l'utilisateur de choisir un chiffre et faites en sorte que cette fonction renvoie le nombre choisi par l'utilisateur.

LEVEL 5-3

“Entre parenthèses”

PARAMÈTRE...

Toute fonction peut avoir un ou plusieurs paramètres.

```
from random import randint
```

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

Dans la fonction **randint** 1 et 6 sont les paramètres de la fonction.

...ARGUMENT

Pour pouvoir utiliser ces paramètres une fonction a besoin d'arguments

```
def double(number):  
    return number * 2
```

Ici **number** est l'argument de la fonction **double**

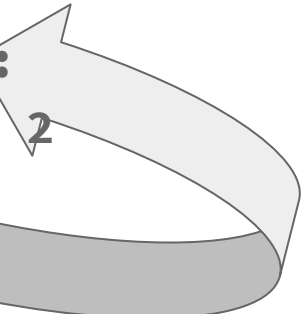

```
def list_multiplication(array, number):  
    clone = []  
  
    for n in array:  
        clone.append(n * number)  
  
    return clone
```

Et ici **array** et **number** sont les arguments de la fonction **list_multiplication**

ARGUMENT... PARAMÈTRES ...?

Lors de l'appel d'une fonction, le compilateur passe la valeur des paramètres dans les arguments respectifs:

```
def double(number):  
    return number * 2
```



```
value = double(5)
```

Et bien sûr, il faut récupérer la valeur renvoyée dans une variable ou autre, sinon elle est perdue.

LA MENACE BINGO

Dans le bingo, faites en sorte de sortir le code qui vérifie le nombre rentré par le joueur dans une fonction.

Le code affichera si le nombre est plus petit ou plus grand, et renverra **True** si le joueur à deviné juste et **False** dans les autres cas.

La fonction prendra en paramètres le nombre donné par le joueur ainsi que le nombre à deviner.

LEVEL 5-4

“Valeur par défaut...”

COURAGE...

Il y a moyen de donner une valeur par défaut à un argument.

Cette valeur lui sera donnée si le code appelant ne le fait pas.

```
def dice_simulation(faces=6):  
    dice = random.randint(1, faces)  
    return dice
```

Dans ce code l'argument **faces** vaudra 6 si l'appelant ne le définit pas.

DERNIÈRE CHOSE NOUVELLE POUR AUJOURD'HUI...

```
result = dice_simulation(10)
```

Ici **faces** vaut 10

```
result = dice_simulation()
```

et ici **faces** vaut 6

Il est donc possible d'omettre des paramètres s'ils ont une valeur par défaut.

ON PEUT BIEN-SÛR MÉLANGER

Il est évidemment possible de mélanger argument normal et argument avec valeur par défaut.

Mais, les arguments avec valeurs par défaut doivent toujours être en dernier dans la liste des arguments.

```
def dices_simulation(dice_number, faces=6):  
    result = 0  
  
    for n in range(dice_number):  
        dice = random.randint(1, faces)  
        result = result + dice  
    return result
```

LA REVANCHE DU BINGO

Dans le code du bingo, modifiez la détermination des bornes du jeu à l'aide d'une fonction avec une valeur par défaut.

La première fois, le nombre est choisi de manière normale, mais à chaque fois que le joueur recommence de jouer, la borne maximum est modifiée de manière à valoir **$\text{born_max} + (7 - \text{tentatives})$** .

La toute première fois où elle est appelée, la fonction n'aura pas de paramètre, et les autres fois elle aura le nombre de tentatives comme paramètre.