Tutoriel robot (bot) pour *Discord* en *Python*

Version 3.5

DIVERSITY by EPITECH

Sommaire

Consignes

- Pour ce projet-là, il vous sera demandé de créer un repository avec le nom : *cc_bot_discord_python*.
- N'oubliez pas de push régulièrement !
- En cas de question, pensez à demander de l'aide à votre voisin de droite. Puis de gauche. Ou inversement. Puis demandez enfin à un Cobra si vous êtes toujours bloqué(e).
- Pensez à faire valider chaque partie que vous réaliserez à un Cobra lorsque vous l'aurez terminée.
- N'hésitez pas à faire des bonus et à ajouter des fonctionnalités lorsque votre projet sera terminé et validé.
- Vous avez tout à fait le droit d'utiliser internet pour trouver des réponses ou pour vous renseigner.

Présentation

Cet hiver est le plus froid depuis des années. Les habitants de Pokéville, étant obligés de restés chez eux, s'ennuient socialement. Le maire connait un outil qui pourrait leur permettre de communiquer entre eux et de s'occuper par ces temps : Discord. Le maire sait aussi qu'il est possible de créer des robots en Python mais n'y connaissant rien à la programmation il requiert vos compétences de développeur.

Il souhaite mettre en place beaucoup de fonctionnalités en très peu de temps. Cependant vous savez que c'est impossible mais que vous ferez de votre mieux pour avoir de premiers bons résultats rapidement.

Etant donné qu'il fait froid, vous vous dites que choisir un Pokémon de type feu permettrait aux gens de se réchauffer ! Mais les gens aiment aussi ce qui est mignon, votre choix s'oriente donc vers *Salamèche*.

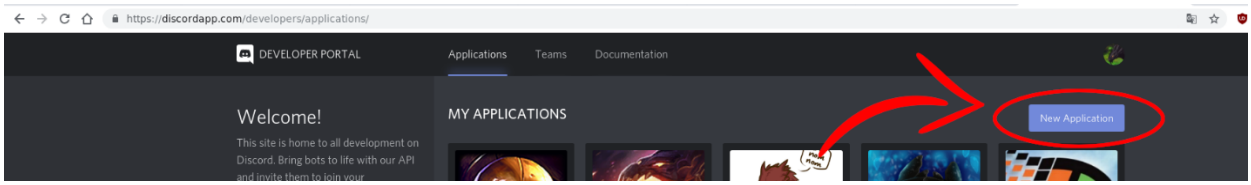
Votre outil principal sera la documentation pour les bots en Python qui se trouve sur ce site.

Pour le bon déroulé de ce document, il vous faudra installer Python ainsi que les dépendances nécessaires précisées dans le document d'installation de Python et ses outils. Il vous faudra aussi installer FFMPEG, voir document dédié à ce propos.

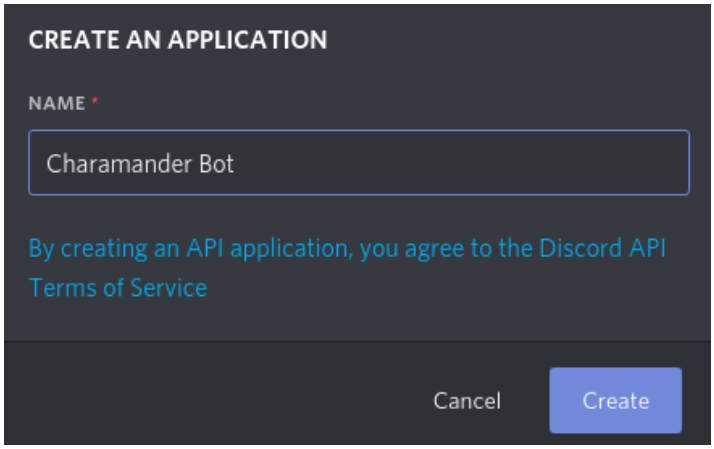
I - Création du compte bot

Pour créer votre bot, connectez-vous ici à votre compte Discord (ou faites-en un si vous n'en avez pas).

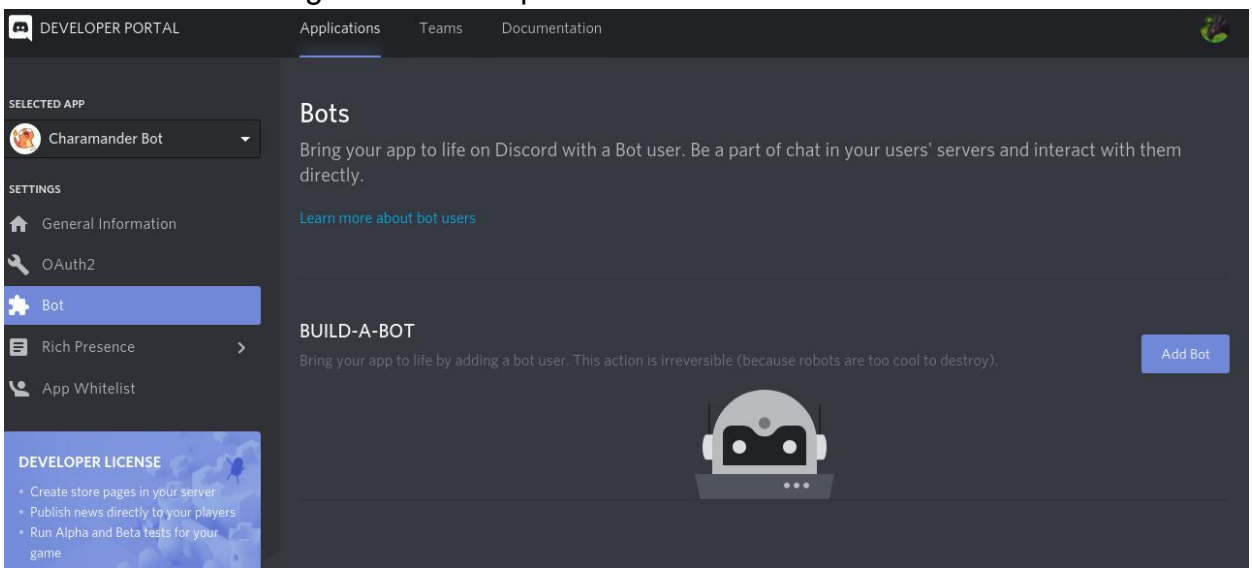
Ensuite, rendez-vous sur le portail des développeurs. Et cliquez sur "New Application".



Nommez votre bot et cliquez sur "Create".



Allez dans l'onglet "Bot" et cliquez sur "Add Bot".

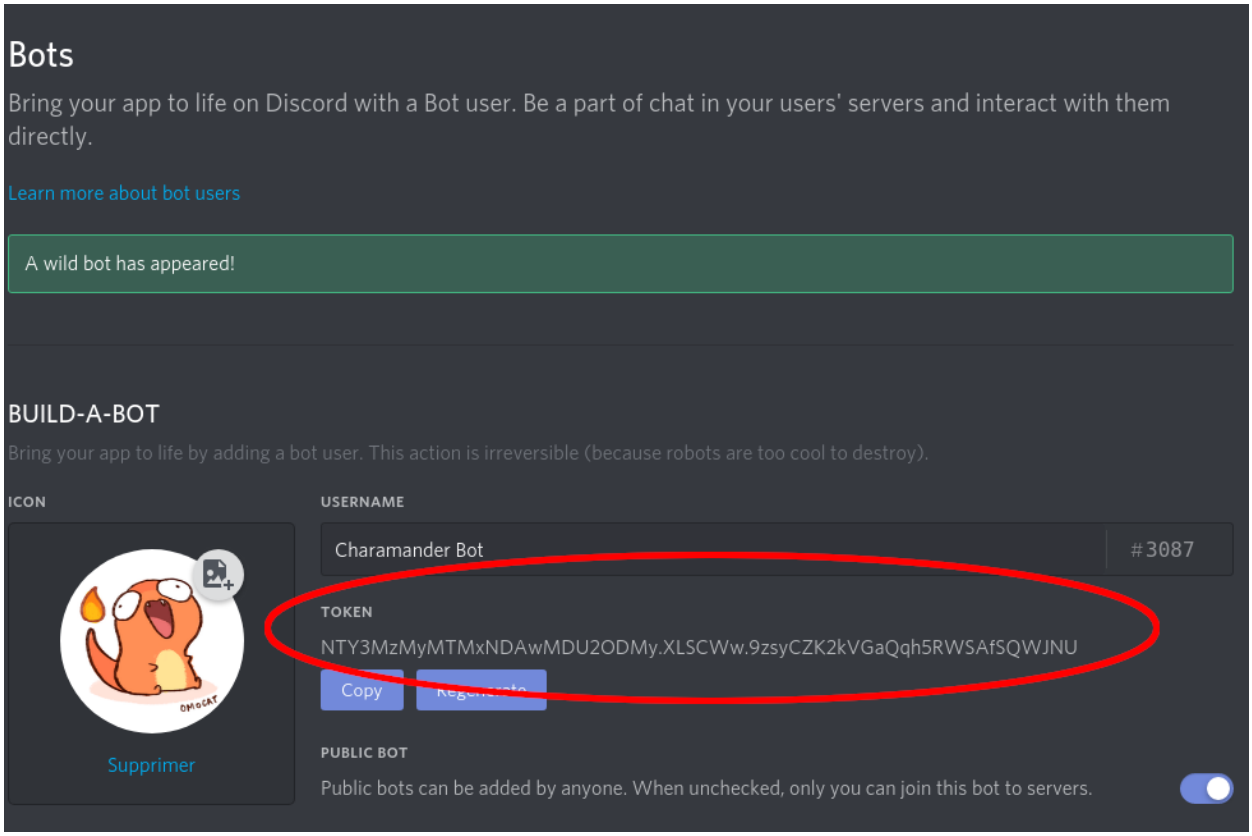


/!\ Ne divulguez pas votre token

Votre **token** correspond à votre “clé d’identification”, c’est à dire que c’est cette chaîne de caractère va vous servir à connecter votre programme au compte bot.

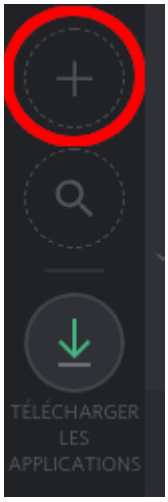
Une personne mal intentionnée pourrait faire n'importe quoi sur les serveurs sur lesquels se trouve votre bot !

Il ne faut donc absolument **JAMAIS LE PARTAGER**.

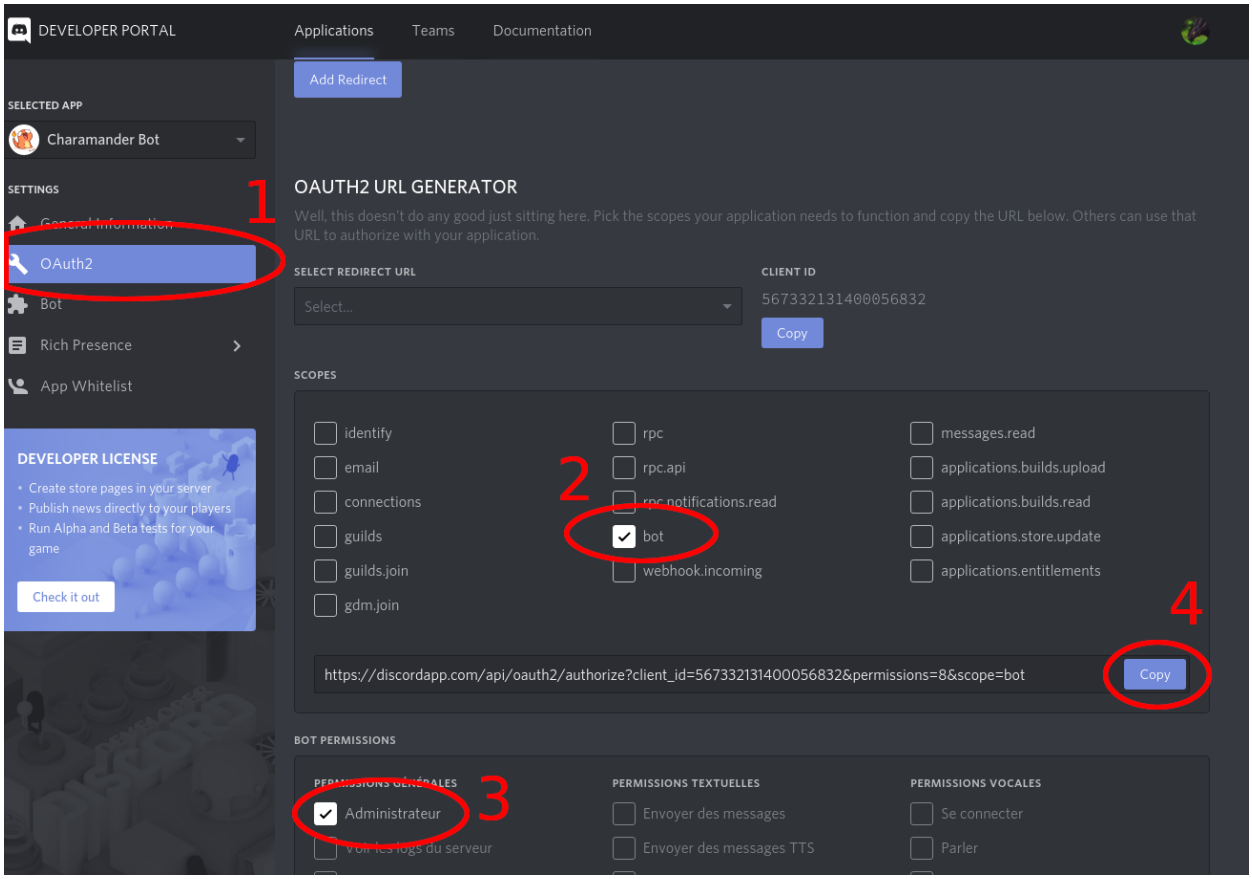


Cette chaîne sera à mettre à la place de “VOTRE TOKEN” dans le code plus tard.

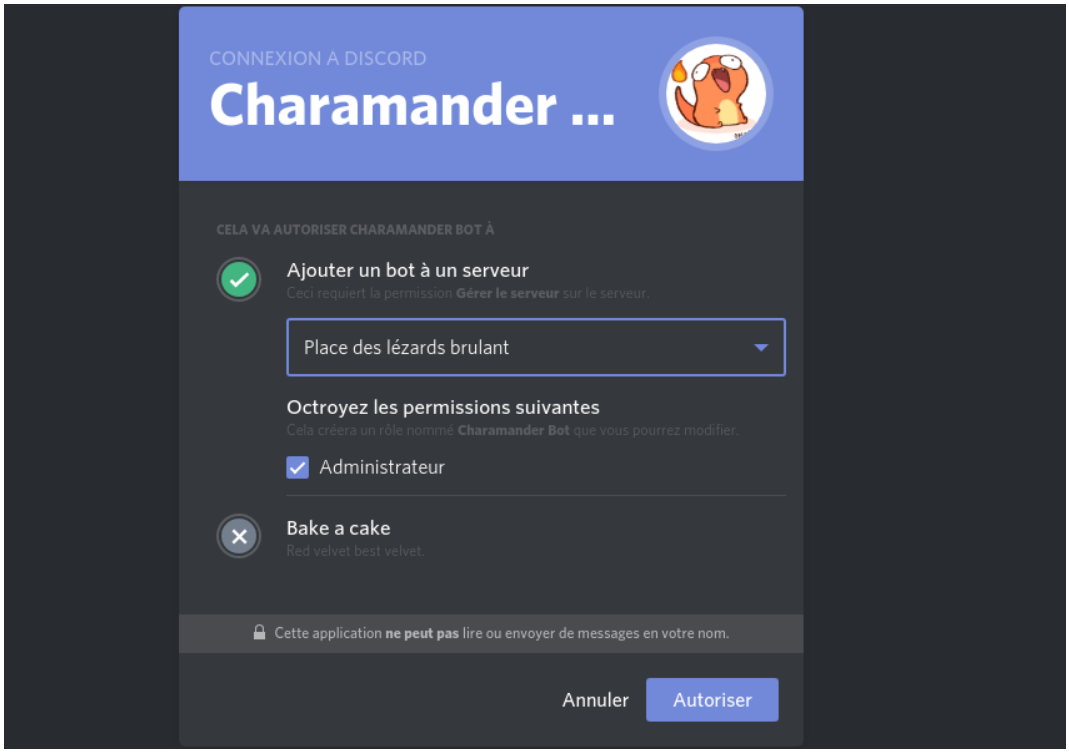
Ensuite, créez un nouveau serveur pour tester le bot.



Retournez sur la page précédente, et créer l'URL Oauth en suivant ces étapes :



Collez l'URL dans votre navigateur, vous allez arriver sur cette page, sélectionnez le serveur que vous venez de créer et cliquez sur "**Autoriser**".



À présent, le bot est enfin sur notre serveur, il apparaît déconnecté et c'est normal (le programme n'est pas lancé).

Pour connecter le bot, il faut écrire un programme qui va se connecter avec le token qu'on a généré plus haut.

Et c'est ce qu'on va faire !

Félicitations, vous avez atteint le niveau 1 !

Vous avez réussi à créer un bot et à le mettre sur un serveur.

II - Hello World !

Avant de coder quoi que ce soit, créez un fichier **.py** dans votre dossier de travail. Ce fichier vous servira justement à pouvoir coder.

A - La fonction on_message

Votre première tâche sur ce bot est de réussir à le faire fonctionner un tout petit peu pour voir si vous arriver au moins à récupérer les messages qui passent sur le serveur de Pokéville.

Pour cela, vous avez trouvé le petit bout de code suivant qui permettrait d'après le Pokénet de bien débuter pour créer le bot :

```
import discord
import asyncio

client = discord.Client()

@client.event
async def on_ready():
    print("Capitaine Salamèche à l'écoute !")

@client.event
async def on_message(message):
    print(message.content)


client.run("VOTRE_TOKEN")
```

Vous êtes perdu tel un petit lézard dans les bois ? Pas de panique !

La seule chose qu'il faut comprendre, c'est que la fonction `on_message` est exécutée à chaque fois que votre bot voit un message. Dans cet exemple, il se contentera d'afficher le contenu du message reçu dans la console.

B - En avant Salamèche

Pour lancer votre programme, il suffit d'appuyer sur F5 si vous êtes sur l'éditeur Python sous Windows, une fenêtre pop-up va apparaître pour vous demander de sauvegarder votre fichier, cliquez sur "oui". Sinon, écrivez la commande suivante dans un terminal : `python votreFichier.py` ou `python3 votreFichier.py` selon votre système d'exploitation.

Si une erreur comme ci-dessous apparaît, il n'y a rien de grave, ne vous inquiétez pas ! C'est probablement que vous n'avez pas indenté votre code. Le Python n'utilisant pas les accolades pour délimités ses "blocks" mais les indentations (touche de tabulation, n'utilisez pas la barre espace).

```
  File "bootstrap.py", line 9
    print(message.content)
        ^
IndentationError: expected an indented block
```

Si tout s'est bien passé, vous devriez voir dans la console les messages que vous envoyez sur votre serveur.

C'est un bon début, mais vous pouvez faire mieux encore !

C - Ses premiers mots

Vous allez écrire votre première commande !

Vous avez pu voir que vous étiez capable de récupérer le contenu d'un message, mais vous pouvez aussi l'utiliser dans votre programme.
Par exemple, vous pouvez comparer si le message correspond à une commande, pour ça vous allez dire au bot :

Si le contenu du message est égal à "!hello" : alors on envoie dans le canal de provenance du message "Bonjour humain !".

En code, ça donne ça :

```
@client.event
async def on_message(message):
    if message.content == "!hello":
        await message.channel.send("Bonjour humain !")
```

Le « **await** » est notion complexe mais pour la résumer simplement, elle permet à votre programme d'attendre que la commande suivante soit revenue pour faire autre chose. En attendant, sachez qu'il faut l'utiliser uniquement quand votre programme

envoie des choses sur Discord.

Vous avez maintenant un bon petit bot capable de traiter des messages et de répondre en conséquence. Vous pouvez utiliser plusieurs conditions si vous voulez plusieurs comportements, par exemple :

```
@client.event
async def on_message(message):
    if message.content == "!hello":
        await message.channel.send("Bonjour humain !")
    if message.content == "!carapuce":
        await message.channel.send("Beurkkkk")
```


Félicitations, vous avez atteint le niveau 3 !

Vous avez fait toutes les installations pour le bot.

III - Un objet mystérieux

Durant vos recherches sur les différentes fonctionnalités et comment les faire fonctionner, vous avez beaucoup entendu parler (ou plutôt lu) à propos des objets.

A - Un objet ? Keskecé ?

Un objet, pour résumer, c'est quelque chose qui contient des valeurs : ça peut être une chaîne de caractères (aussi appelé "*string*" ou "*str*") comme notre `message.content`, une structure plus complexe comme une liste ou un dictionnaire ou bien un autre objet.

Par exemple, notre objet `message`, qui nous est donné par la fonction `on_message`, contient plusieurs attributs intéressants :

`content` -> Donne le contenu du message.
`channel` -> Donne l'objet *channel* (les informations du salon (textuel ou vocal)).
`guild` -> Donne l'objet *guild* (les informations du serveur discord).
`author` -> Donne l'objet *member* (les informations de la personne ayant écrit le message)

Si vous souhaitez connaître l'objet message sur le bout des griffes, vous pouvez vous renseigner sur cette partie de la documentation officielle.

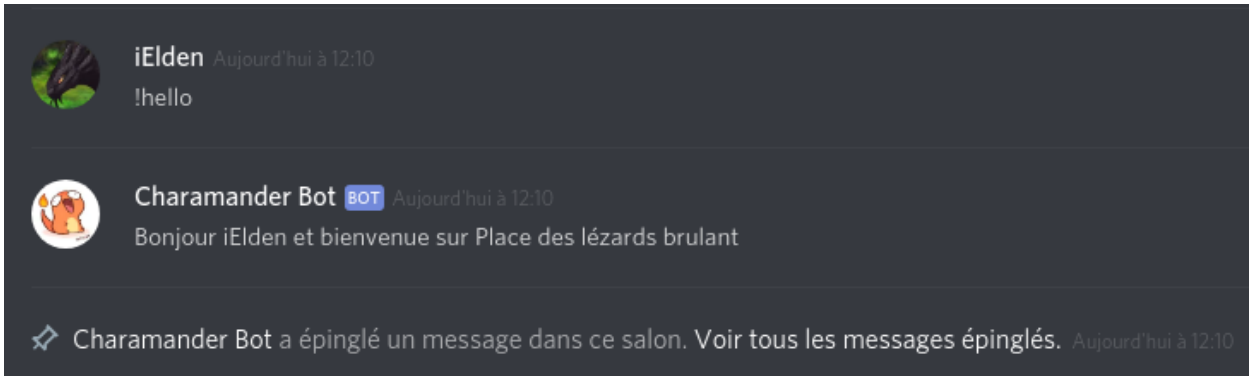
B - Exploitons la puissance de l'objet

Vous avez tout ce qu'il vous faut dans votre boîte à outils à présent ! Donc vous pouvez ajouter quelques fonctionnalités au bot ! Le Maire de Pokéville aimerait que ces concitoyens puissent retrouver facilement des informations, dans la même idée qu'ils pourraient trouver des affiches dans la rue.

Vous allez pouvoir, par exemple, améliorer votre `!hello` pour qu'il salue la personne en disant son nom.
Pour faire cela, on va utiliser l'attribut `name` de l'objet `message.author`.

L'objet message est très complet, vous pouvez vraiment faire énormément de choses avec. Par exemple, rien ne vous empêche de saluer votre interlocuteur en lui souhaitant la bienvenue sur votre serveur, puis ensuite d'épingler le message.

Résultat :



Vous avez ici un bon exemple sur l'utilisation général des objets :

Les objets ont des attributs (comme `.channel`, `.guild` ou `.author`), ce qui nous permet d'avoir d'autres objets, généralement pour récupérer des informations.

Les objets ont des fonctions (comme `.send()` ou `.pin()`) ce qui nous permet d'effectuer des actions avec un bot (envoyer un message, le supprimer, gérer un membre, un rôle ect...).

Vous pouvez, par ailleurs, voir que .send() renvoie un nouvel objet `Message`, qui est ensuite utilisable comme celui que l'on reçoit au début.

Félicitations, vous avez atteint le niveau 4 !

Vous savez maintenant utiliser et interpréter, en partie, les messages.

IV - Embed

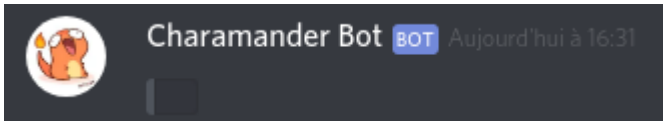
A - Un peu de mise en forme

Vous souvenez-vous de la dernière requête du maire ? Il parlait d'affiches, quoi de mieux qu'un "**embed**" pour que les informations, qu'il souhaite transmettre, soient plus jolies que dans un simple message ?

Pour faire un embed minimum, ce n'est pas très difficile :

```
em = discord.Embed()
await message.channel.send(embed=em)
```

Résultat :



Bon, ce n'est pas giga ouf à ce moment-là, mais ça va permettre d'introduire cette nouvelle notion.

B - Les paramètres optionnels

En python, les fonctions peuvent prendre des paramètres optionnels, on appelle ça des ***K**ey**w**ord **Arg**uments* (abrégés en ***kwargs***).
Par exemple, la fonction `.send()` que vous pouvez voir ci-dessus prend un paramètre optionnel appelé `embed` qui nous permet de rajouter un embed en plus du message.

Ce qu'il y a de bien avec les embeds, c'est qu'ils prennent PLEIN de kwargs.

Essayez par exemple avec ce code :

```
    em = discord.Embed(title="titre", description="desc",
colour=0xFF0000,
                       timestamp=message.created_at)
    em.add_field(name="Un field", value="Youpi", inline=True)
    em.add_field(name="Un autre field", value="o/", inline=True)
    em.add_field(name="Un field pas sur la même ligne", value="grâce
au 'inline'",
                 inline=False)
    em.set_author(name="Un gars super",
icon_url=message.author.avatar_url)
    em.set_footer(text="Sur un super serveur",
icon_url=message.guild.icon_url)
    em.set_image(url="https://cdn.discordapp.com/attachments/"
                     +
"567630033246617621/581496185424969749/1733852_0.jpg")
    await message.channel.send(embed=em)
```

Résultat :

Ça ressemble quand même plus à quelque chose comme ça, non ?

On ne va pas trop s’attarder sur les embeds, mais si vous voulez plus d’information dessus et connaître toutes leurs possibilités (et il y en a plein), renseignez-vous sur internet.

Félicitations, vous avez atteint le niveau 5 !

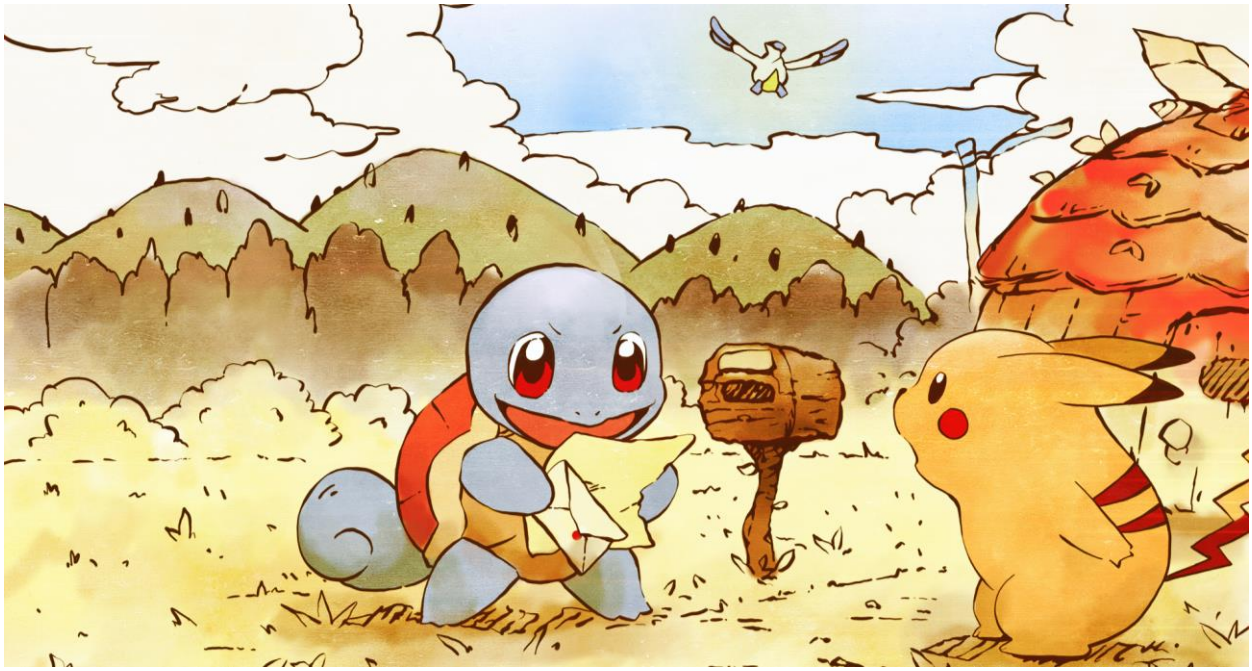
Vous savez maintenant utiliser des embeds.

V - Charmander the Robocop

A la mairie de Pokéville, une nouvelle loi a été votée interdisant à la population de prononcer certains mots jugés contrariants.

Le service courrier Bekipan a distribué la liste à tous les habitants de Pokéville mais la mairie n'a pas assez de poké$ pour engager assez d'administrateurs sur son serveur Discord pour faire respecter la nouvelle loi.

C'est pour ça que l'on fait appel à vous !



A - Les listes

Une liste en Python est une structure qui permet de contenir plusieurs variables. Ici, on va s'en servir pour contenir les mots interdits par la nouvelle loi.

Déclarez, en haut de notre code, une liste de mots interdits comme par exemple :

```
BAN_WORDS = ["pokeball", "carapuce", "dresseurs", "javascript", "giratina"]
```

En Python, les variables sont écrites en majuscules pour informer les autres programmeurs que ce sont des constantes. C'est-à-dire que le programme doit consulter les variables mais pas les modifier.

B - Les boucles for

Une boucle for en Python, va exécuter le code compris dans le bloc autant de fois qu'il y a d'éléments dans la liste. A chaque tour de boucle, la variable va prendre la valeur d'un élément de la liste.

Par exemple, le code ci-dessous :

```
for word in BAN_WORDS:
    print("La variable word vaut : " + word)
```

Va produire ce résultat :

```
La variable word vaut : pokeball
La variable word vaut : carapuce
La variable word vaut : dresseurs
La variable word vaut : javascript
La variable word vaut : giratina
```

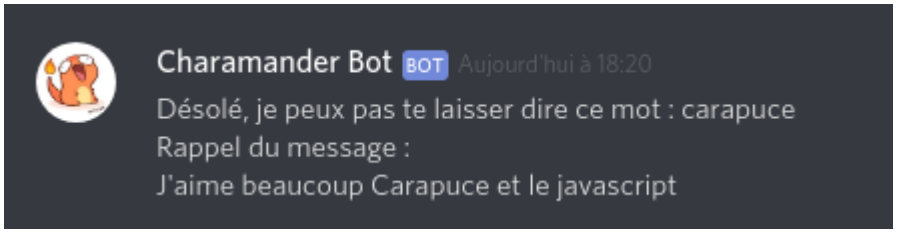
On va donc vérifier un par un si chaque mot est présent dans le contenu du message. Et pour ça, il vous faut utiliser le mot-clé `in`.

```
if word in message.content:
    print("ALERTE : mot interdit utilisé")
```

En combinant cela et ce que vous avez vu dans le Chapitre III, vous pouvez donc effectuer la procédure suivante :

* Supprimer le message de la personne incriminée.
* Envoyer un avertissement contenant le mot interdit utilisé et le message d'origine.

On souhaite ici que le résultat soit équivalent au suivant :



C - Triomphe de l’ordre

Avez-vous regardé dans la documentation ce que faisaient les fonctions suivantes ?

* `delete()`
* `send("Quelque chose")`
* .lower()

Il y a fort à parier que cela vous servirait !

Le saviez-vous ? `break` est un mot-clé qui “casse” la boucle lorsque vous l’utilisez. Ce qui peut vous éviter que le bot veuille supprimer deux fois le message si plusieurs mots bannis sont utilisés.

Dans votre algorithme, avez-vous pensé à regarder qui était l’auteur du message ? Ce serait dommage que *Salamèche* verbalise ses verbalisations…

Par ailleurs, voici le message du maire pour *Salamèche* :

Toute mes félicitations agent Salamèche ! Grâce à vous et à votre maîtrise sans égal des listes, la paix et l’ordre ont été rétablis sur le serveur de Pokéville, loin de tous les mots barbares que l’on peut rencontrer.

Félicitations, vous avez atteint le niveau 5 !

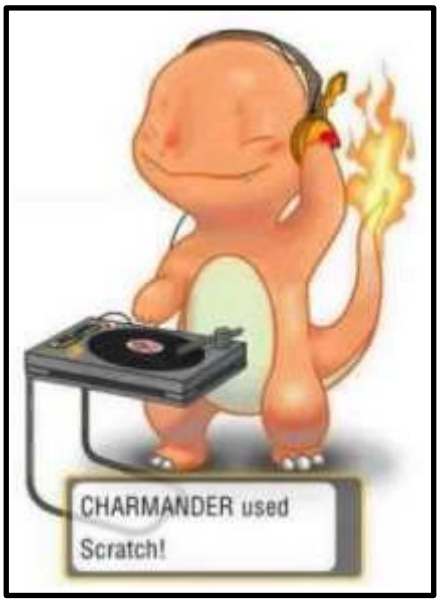
Vous avez réussi à sécuriser le serveur de Pokéville pour que tout se passe au mieux pour ses habitants.

VI - DJ Salamèche

Queulorior est venu vous voir, en panique, pour vous parler de son problème.

En effet, *Rondoudou* ne veut plus animer la soirée discothèque à la maison car il est frustré que les gens s'endorment pendant ses prestations.

Il compte sur vous pour animer la soirée d'aujourd'hui !



A – Mise en place du matériel

Queulorior ayant déjà commencé à préparer la soirée a trouvé quelque chose qui pourrait vous servir. Allez savoir par quel hasard ça porte de nom de *SalamecheMusic.py*… Cependant il semblerait que ce soit un module dont vous pourriez avoir besoin.

Donc dans un premier temps, déplacez le module nommé *SalamecheMusic.py* dans votre dossier de travail. Puis, dans votre fichier .py, importez le module en ajoutant cette ligne en haut du fichier :

```
import SalamecheMusic
```

Attention on fait import *SalamecheMusic* et non *SalamecheMusic.py* car Python ajoute tout seul le *.py*.

B - La méthode split

Une dernière chose avant de terminer, la méthode *split*. Elle va vraiment vous être utile. Elle permet de séparer une chaîne de caractère en une liste de chaîne de caractères, en d'autres mots : changer une phrase en liste de mot.

Quelques exemples :

```
>>> message.content
!play http://youtube.com/randomvideo
>>> message.content.split(' ')
['!play', 'http://youtube.com/randomvideo']
>>> message.content.split(' ')[0]
!play
>>> message.content.split(' ')[1]
http://youtube.com/randomvideo
```

Maintenant, vous avez tous les outils en mains pour pouvoir assurer la soirée ! Essayez donc de faire une commande "*!play {lien youtube}*" pour que le bot joue la musique envoyée dans le canal vocal.

Petit guide de départ pour le module *CobraMusic* :

`music_client = await CobraMusic.get_client(message, client)`

Renvoie un objet qui représente le client vocal du bot, le crée et le connecte au vocal s'il n'existe pas déjà.

`await music_client.play(url_youtube)`

Joue la musique dans le canal où vous vous trouvez.

Regardez les autres commandes auxquelles vous avez accès dans le module. Elles pourraient vous être utiles. 😉

Félicitations, vous avez atteint le niveau 6 !

Vous avez sauvé de nombreuses soirées endiablées !

VII - Pour aller plus loin

Le maire de Pokéville est très heureux du travail que vous avez fourni ! Le serveur devrait donc pouvoir tourner avec de bonnes fonctionnalités maintenant ! Bien que ce soit déjà la fin de cette histoire, ce n'est que le début de votre épopée !

Les possibilités de création sont infinies et si vous ne savez pas quoi faire actuellement, voici quelques petits défis (relativement difficiles) :

* Reprendre l'exercice précédent et rajoutez des fonctionnalités, comme une *queue* pour les musiques par exemple.
* Refaire l'exercice précédent sans utiliser le module *SalamecheMusic*.
* Créer une nouvelle fonctionnalité comme un questionnaire interactif.
* Faire une image personnalisée avec la photo de profil de chaque nouvel arrivant sur un serveur (manipulation d'images).

Si vous avez fini et que vous vous ennuyez, essayez de trouver d'autre défis par vous-même !

Un des grands avantages de Python, c'est qu'il existe des tonnes de modules pour pouvoir faire facilement tout ce dont vous avez envie, on a vu *youtube-dl* pour *Youtube*, mais il en existe plein d'autre, comme *PIL* pour la manipulation d'image, *datetime* pour encore plus gérer les heures et dates ou bien l'*API* de votre jeu vidéo favori pour récupérer des informations !

En espérant vous avoir donner envie de développer encore plus !