Coding Club

Bot Discord

Réalisation d'un bot discord avec *discord.py*

Fait par : Theo

1. Introduction

L'objectif du jour est de réaliser un Bot pour votre serveur **Discord**. Pour le réaliser nous utiliserons **IDLE**. Ainsi que la librairie **discord.py**

Si vous n'avez pas encore de compte discord, rendez-vous ici

C'est quoi Python ?

Python est un **langage de programmation**. Tout comme les êtres humains utilisent plusieurs langages pour communiquer, en informatique il existe plusieurs langages pour communiquer avec sa machine, et le Python en fait partie.

L'avantage d'utiliser Python ? C'est que c'est un langage rapide à prendre en main, en effet, il suffit d'écrire ce que tu veux faire en anglais et en principe ça devrait fonctionner.

Il comporte néanmoins certaines particularités comparé aux autres langages, comme le fait qu'il est très à cheval sur l'**indentation** (le nombre d'espace – tabulation) et qu'il ne nécessite pas de **compilation**. C'est un langage **interprété**. On n'est pas obligé de convertir le fichier qui contient ton code en un fichier exécutable comme les « .exe ».

C'est quoi discord.py ?

Discord.py est une **bibliothèque**. En informatique une bibliothèque est un ensemble de fonctions, qui, dans notre cas nous permettrons de créer une fenêtre, d'afficher une image à telle endroit, etc…

Si nous n'avions pas cette bibliothèque, il nous faudrait recoder chacune des fonctions permettant d'envoyer/analyser les messages ou encore se connecter au serveur discord, ce qui serait long, fatiguant et compliqué, grâce à cette bibliothèque, ont va pouvoir se simplifier la tâche et utiliser des fonctions déjà prêtes.

Discord est un service de discussion instantanée en ligne, il permet de chatter dans différents salons eux-mêmes dans des serveurs, tant à l'écrit qu'à l'oral.

1.1 Environnement de travail

Fais un dossier **Bot** sur le bureau de ton Ordinateur, c'est dans ce fichier que tu vas mettre tout ce dont tu auras besoin pour ton bot.

Commence par créer un fichier Python qui te permettra d'écrire ton code (un fichier Python est un fichier dont l'extension est « .py », si tu vois toujours l'icône d'un fichier texte (.txt) ou que tu n'arrives pas à créer le fichier, **demande à un Cobra**, n'aies pas peur, ils sont gentils ! Evidemment tu peux nommer ton fichier python comme tu le souhaites tant qu'il finit bien par « .py » (évite « discord », pour limiter les conflits).

Si à un moment dans la journée ton bot a besoin d'utiliser des images, choisis-les sur Internet puis enregistre les, et n'oublies pas de les mettre dans ton dossier **Bot**.

Pour le moment, ton dossier « Bot » ne comporte que ton fichier Python ! Mais prend garde à comment tu comptes remplir ce dossier, pour éviter d'avoir toutes tes futurs images / musiques / etc… en vrac dans ton dossier, commences dès maintenant à créer un dossier « Image », « Music » où tu mettras toute tes images, musiques, …

Pour modifier votre fichier Python, on te recommande d'utiliser « **IDLE** », il te permettra d'écrire dans ton fichier tout en lançant ton programme Python.

Pour l'ouvrir, fais un **clic droit** sur ton fichier Python -> **Edit with IDLE** puis tu peux éditer ton fichier ! Pour le lancer tu n'auras qu'a appuyer sur **Run → Run Module** (en haut d'IDLE) ou **F5**.

Il existe plusieurs autres « éditeur de texte », si tu es plus famili » avec Vim, Notepad++, Sublime Text, Visual Code voire même le Bloc Note de Windows, tu peux les utiliser.

2. Programmation

2.1 Créer le bot

Commence par te connecter sur le navigateur de ton choix en cliquant ici.

Une fois fait, rends-toi sur la partie « développeurs » de discord en suivant ce lien : https://discordapp.com/developers.

Crée ensuite une nouvelle application en cliquant sur le bouton en haut à droite, entre le nom que tu souhaites donner à ton bot, puis cliques sur « créer ».

Tu as maintenant accès à tous les paramètres du bot. Changes sa photo, ajoutes une description, ça te permettra de plus facilement te repérer si tu crées beaucoup de bots. Tu peux chercher un peu dans les menus toutes les options disponibles. (Attention à ce que tu fais certains paramètres sont importants).

Nous allons à présent nous rendre dans le menu « bot » et cliquer sur « ajouter un bot ». Encore une fois, de nouvelles options apparaissent. Cette fois-ci tu peux définir le nom et l'image de ton bot lorsqu'il arrivera sur le serveur.

On va maintenant appliquer des permissions. Coche la case « Administrateur ».

Invitons maintenant le bot sur ton serveur. Dans le menu à gauche cliques sur « Informations générales » puis rends toi sur ce lien :

https://discordapp.com/oauth2/authorize?client_id=<CLIEN ID>&scope=bot&permissions=8

En remplaçant « CLIENT ID » par le code qui apparaît sur la page.

Et voilà ! Ton bot est presque prêt à l'emploi. Ne reste plus qu'à le coder

Attention ! Donner la permission administrateur à un bot peut s'avérer dangereux sur un serveur. Nous le faisons ici car ce n'est qu'un serveur de test. Ne donnes à ton bot que les permissions nécessaires à son fonctionnement si tu comptes l'ajouter sur ton propre serveur plus tard.

2.2 Connecter son bot

On arrive maintenant à la partie la plus drôle ! On va enfin toucher au code, en commençant par connecter le bot au serveur

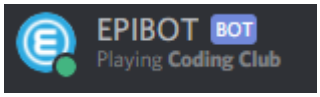
C'est le moment de te rappeler que tu peux à tout moment faire appel à un Cobra ! Ils sont là pour t'aider et répondre à questions si tu as du mal à réaliser une tâche.

```
# Import permet d'importer une bibliothèque pour utiliser ses fonctions, ici discord.
import discord

# Le token est un code unique que tu trouveras dans Bot → Token
TOKEN = 'le token de ton bot'
# le symbole qui se touvera devant les commandes
prefix = ' !'

class MyClient(discord.Client):
    # Lorsque le bot se connectera il vous le dira
    async def on_ready(self):
        print('Logged on as', self.user)

# Cette partie sert uniquement à lancer le bot
client = MyClient()
client.run(TOKEN)
```

Maintenant ton bot devrait se connecter sans difficultés au serveur. Si ce n'est pas le cas, appelle un Cobra

2.3 L’activité

Avant de commencer à mettre des options fun, on va ajouter une activité au bot. Son activité correspond à ça :



Pour se faire on va rajouter 2 lignes dans le code au moment où il se connecte.

```
class MyClient(discord.Client):
    # Lorsque le bot se connectera il vous le dira
    async def on_ready(self):
        activity = discord.Game(name="Coding Club")
        await client.change_presence(status=discord.Status.online, activity=activity)
        print('Logged on as', self.user)
```

Pour changer le jeu auquel joue ton bot, modifie simplement la valeur de « name »

2.4 La première fonctionnalité

Ce bot est bien calme tu ne trouves pas ?

Il est temps qu’il réagisse aux messages ! Nous allons lui faire observer ce qu’il se passe dans les différents salons. Nous allons vérifier qu’il détecte les messages. À la suite de la fonction « on_ready » ajoute ceci :

```
# Lorsqu’il détecte un message
async def on_message(self, message):
    # On vérifie que ce ne soit pas son message
    if message.author == self.user:
        return

    # Si le message est « ping »
    if message.content == 'ping':
        # On répond « pong »
        await message.channel.send('pong')
        return
```

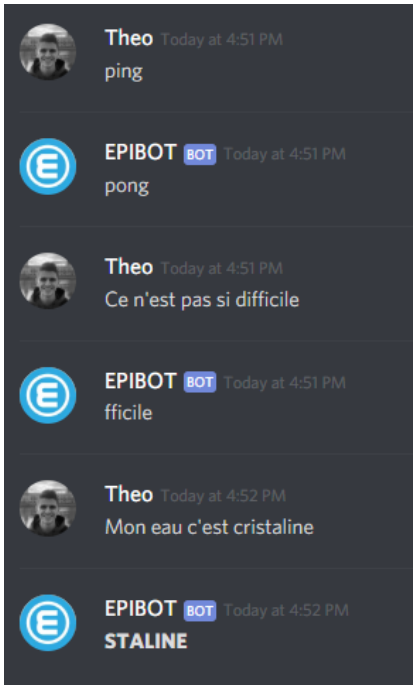
2.5 Fonctionnalités inutiles

Nous voici à la partie la plus fun de ce Coding Club *(j'ai menti)* !

Nous allons ajouter quelques fonctionnalités totalement inutiles mais plutôt fun.
Commençons par 2 réactions, répéter la suite d'un mot qui contient le préfixe « di » ou « cri »

Rajoute à la suite du « ping » le code qui suit :

```
# On sépare le message en 2 parties en fonction du préfixe « di »
message_split = message.content.split("di")
    # On vérifie que le préfixe soit présent
    if len(message_split) == 2:
        # On ajoute une sécurité pour limiter les erreurs
        if len(message_split[1]) > 0:
            # On envoie finalement le message
            await message.channel.send(message_split[1])
        return

# On applique le même code que précédemment et on l'adapte à la situation
message_split = message.content.split("cri")
    if len(message_split) == 2:
        if len(message_split[1]) > 0:
            await message.channel.send("**" + message_split[1].upper() + "**")
    return
```



La fonction split permet de séparer le message en 2 à chaque fois qu'on y trouve l'identifiant *(ici « di » ou « cri »).* On enregistre le tout une « liste ». Tu y retrouves les valeurs en appelant cette liste grâce aux crochets *(ex : [0], [1]).*

Ton bot avance bien, il peut exécuter des tâches simples sans problèmes.

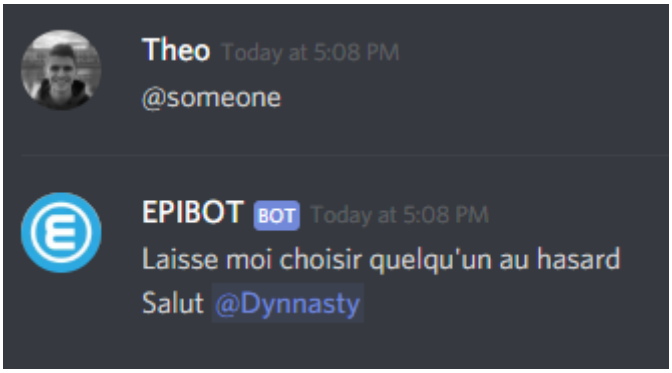
Si quelque chose ne fonctionne pas à cette étape, vérifié l'indentation de ton code ou appelle un Cobra !

On va maintenant ajouter une fonctionnalité assez sympa !

On va tenter de mentionner un utilisateur au hasard présent dans le serveur d'une part parce que c'est un concept en python assez sympa et ensuite parce que c'est plutôt marrant à utiliser de façon abusive !

Toujours à la suite on va ajouter ces quelques lignes :

```
# Rajoute cette ligne juste après le « import discord »
import random

# Si le message contient « @someone »
if message.content == '@someone':
    # On prend un des utilisateurs présents sur le serveur
    select_user = random.randint(0, len(message.guild.members))
    # On récupère l'id de cet utilisateur
    random_user = message.guild.members[select_user].id
    await message.channel.send("Laisse moi choisir quelqu'un au hasard !");
    # On va mentionner l'utilisateur
    await message.channel.send("Salut <@" + str(random_user) + ">")
    return
```



Le bot commence à savoir faire pas mal de choses, ajoutons encore un dernier élément totalement inutile mais encore une fois marrant ! Nous allons détecter un nouvel évènement : la suppression d'un message. Revenons une indentation avant les derniers « if » et ajoutons ceci

```
# Si un message est supprimé
async def on_message_delete(self, message):
    # Si c'est un message du bot on s'arrête là
    if message.author == self.user:
        return
    await message.channel.send("Alors **" + str(message.author) + "** on
n'assume pas de dire `" + str(message.content) + "`")
```

2.6 Les commandes

Si tu es observateur tu as pu remarquer au début que nous avions créé une variable « prefix » il permet d'identifier une commande, si le message commence par ce(s) caractère(s) le bot sera alors capable d'identifier quelle commande l'utilisateur souhaite faire.

Nous avons mis '!' comme préfixe, tu peux bien entendu mettre ce que tu veux pour rendre ton code un peu plus personnel !

Retournons dans la fonction « on_message » et ajoutons encore quelques lignes

```
# Si une commande est détectée
if message.content[0] == prefix:
    # On commence par séparer les différents éléments du message
    args = message.content.split(' ');
    # Si le premier élément est le préfixe suivi de help
    if args[0] == prefix + "help":
        # On envoie les commandes en privé à l'utilisateur
        await message.author.send("Voici les commandes disponibles :")
        # Et on l'en informe dans le salon
        await message.channel.send("<@" + str(message.author.id) + ">, je
viens de t'envoyer la liste des commandes en mp !")

    # Si l'utilisateur a demandé de répéter quelque chose
    if args[0] == prefix + "repeat":
        # Renvoyer le 2e élément du message
        await message.channel.send(args[1])
```

Félicitations !

Tu es venu à bout du sujet ! Tu vois c'était plutôt simple !

Maintenant profite du temps qu'il te reste pour continuer ton bot, tu as vu plein de choses, à partir de maintenant le seul frein à l'amélioration c'est ton imagination ! Tu peux appeler un cobra pour te donner des idées de bonus ou ajouter tes propres commandes/fonctionnalités.