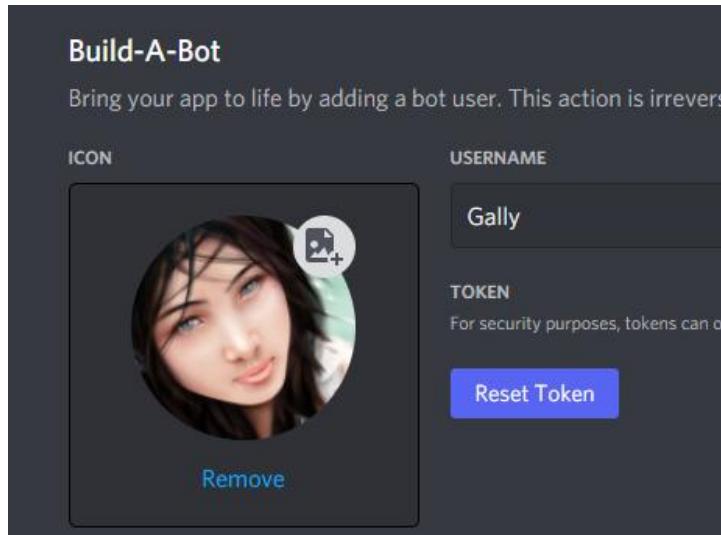


Bot Discord



Un ami du net a reprit un projet abandonné d'un anglophone. Ce projet recensait tous les monstres d'un jeu mobile sur lequel nous jouons.

Le problème était qu'il ne s'y connaissait pas vraiment en programmation et essayait tant bien que mal de faire en sorte que ça fonctionne et de le mettre à jour.

Je me suis alors proposé de regarder le code, voir si je comprenais quelque chose et, si oui, si je pouvais l'aider dans cette tâche.

C'est comme ça que j'ai commencé mon apprentissage en Python.

Présentation du programme :

Ce programme vient du projet nommé Bot Athena. Il a été fait afin de garder Athena déployé pendant que j'effectuais mes tests.

Ces 2 bots sont hébergés par Heroku, nous avons optés pour la version gratuite et n'avions donc que 23 jours de connexion environ.

Nous y hébergeons aussi les variables sensibles grâce à "env".
Pour le reste du code, GitHub est un outil parfait.

Le code en lui même est fait en python, simple à comprendre et à utiliser, il nous a permis d'aller plus loin dans notre démarche.

Le code source et leur action

```
1 import discord
2 from discord.ext import commands, tasks
3 import os
4 import json
5
6 import asyncio
7
8 TOKEN = os.getenv("DiscordBotToken")
9
10 bot = commands.Bot(command_prefix='', description="Ceci est un Bot Discord pour le jeu Monster Super League")
```

Tout commence par les imports, le token et la déclaration de la commande commands.Bot.

Le token a été mis en variable d'environnement.

Lors de l'initialisation de la commande, j'ai décidé de ne pas mettre de préfix pour l'appel du bot, ce qui n'est pas recommandé pour plusieurs raisons :

- Le bot doit "écouter" tout ce qui se dit et analyser chaque début de phrase.
- Le bot peut intervenir à n'importe quel moment alors que l'utilisateur ne faisait que discuter avec un autre utilisateur.

Il ne faut donc pas choisir n'importe quelle commande d'appel car les utilisateurs pourraient s'en trouvé perturbés.



```

378     @bot.command(help="ping_pong")
379     async def ping(ctx: commands.Context):
380         print("ping")
381         if ctx.channel.name != CHANNEL_WORK:
382             return
383         await ctx.send('pong')

```

Pour savoir si le bot est bien déployé, il suffit de regarder s'il est "EN LIGNE". Ça ne signifie pas qu'il est disponible pour autant. J'ai donc fait un ping pour recevoir pong s'il est disponible.

Au début du projet, le code ressemblait plutôt à ça :

```

377 #####
378 ##### Anubis #####
379 #####
380
381 if any([message.content.startswith(item) for item in ['Anu', 'FeuAnu']]):
382     embed=discord.Embed(title="",
383                         url="https://cdn.discordapp.com/attachments/552550283885019166/552551381249032192/AnubisR_large.jpeg",
384                         color=0xffffffff)
385     embed.set_author(name="#16 Anu (Feu)")
386     embed.set_thumbnail(
387                         url="https://cdn.discordapp.com/attachments/552550283885019166/552551381249032192/AnubisR_large.jpeg")
388     embed.add_field(name="★★★", value="**Type**: Tank\n**Lead**: CR +15~20%"
389                     "\n**Passif**: Étourdissement 70% 1 tour\n(No skillbooks)\n**Actif**: Agression (PV)"
390                     "\n(No skillbooks)\n**PV**: 38684\n**Attaque**: 2112\n**Défense**: 2439\n**Récupération**:1772",
391                     inline=False)
392
393     await message.channel.send(embed=embed)

```

Beaucoup de texte, beaucoup de redondance et très peu de lisibilité dans le code.

Exemple de nouvelle commande :

```
98     @bot.command(help="Calculer la meilleure config pour l'epv d'un Healeur")
99     async def GemHeal(ctx: commands.Context, pv: int, defense: int, recuper: int):
100         SmplPv = int(pv*1.68)
101         DblPv = int(pv*2.36)
102         SmplDef = int(defense*1.68)
103         DblDef = int(defense*2.36)
104         SmplRec = int(recuper*1.68)
105         DblRec = int(recuper*2.36)
106         TrplRec = int(recuper*3.04)
107         Epv1 = round(SmplPv / (1 - (SmplDef / (SmplDef + 1500))))
108         Epv2 = round(SmplPv / (1 - (defense / (defense + 1500))))
109         Epv3 = round(pv / (1 - (SmplDef / (SmplDef + 1500))))
110         Epv4 = round(DblPv / (1 - (SmplDef / (SmplDef + 1500))))
111         Epv5 = round(SmplPv / (1 - (DblDef / (DblDef + 1500))))
112         Epv6 = round(pv / (1 - (defense / (defense + 1500))))
113
114         embed=discord.Embed(title="Choix de Gemmes pour un healer",
115                             description="Quelles gemmes choisir ? voici la réponse")
116         embed.set_thumbnail(url="https://wiki.dungeondefenders2.com/images/6e/Heal.png")
117         embed.add_field(name="1 Pv, 1 Def, 1 Recup", value="**Epv : "+str(Epv1)+"**\n\n"
118                         "\nPy : "+str(SmplPv)+"\nDef : "+str(SmplDef)+"\nRecup : " +str(SmplRec), inline=True)
119         embed.add_field(name="1 Pv, 2 Recup", value="**Epv : "+str(Epv2)+"**\n\n"
120                         "\nPy : "+str(SmplPv)+"\nDef : "+str(defense)+"\nRecup : " +str(DblRec), inline=True)
121         embed.add_field(name="1 Def, 2 Recup", value="**Epv : "+str(Epv3)+"**\n\n"
122                         "\nPy : "+str(pv)+"\nDef : "+str(SmplDef)+"\nRecup : " +str(DblRec), inline=True)
123         embed.add_field(name="2 Pv, 1 Def", value="**Epv : "+str(Epv4)+"**\n\n"
124                         "\nPy : "+str(DblPv)+"\nDef : "+str(SmplDef)+"\nRecup : " +str(recuper), inline=True)
125         embed.add_field(name="1 Pv, 2 Def", value="**Epv : "+str(Epv5)+"**\n\n"
126                         "\nPy : "+str(SmplPv)+"\nDef : "+str(DblDef)+"\nRecup : " +str(recuper), inline=True)
127         embed.add_field(name="3 Rec", value="**Epv : "+str(Epv6)+"**\n\n"
128                         "\nPy : "+str(pv)+"\nDef : "+str(defense)+"\nRecup : " +str(TrplRec), inline=True)
129         embed.add_field(name="Attention",
130                         value="\n\nles sub, attirails, amélio, etc. ne sont pas pris en compte-- ;",
131                         inline=False)
132         await ctx.channel.send(embed=embed)
```

La commande GemHeal suivie de 3 nombres permet de calculer le meilleur équipement à avoir afin de survivre le plus longtemps.

Cette fonction attend 3 int afin de faire tous ses calculs et rendre un affichage.

 horotopia 13/04/2022
GemHeal 26859 3153 2479

 Gally BOT 13/04/2022

Choix de Gemmes pour un healer

Quelles gemmes choisir ? voici la réponse

1 Pv, 1 Def, 1 Recup Epv : 204467 Pv : 45123 Def : 5297 Recup : 4164	1 Pv, 2 Recup Epv : 139972 Pv : 45123 Def : 3153 Recup : 5850	1 Def, 2 Recup Epv : 121707 Pv : 26859 Def : 5297 Recup : 5850
2 Pv, 1 Def Epv : 287228 Pv : 63387 Def : 5297 Recup : 2479	1 Pv, 2 Def Epv : 268963 Pv : 45123 Def : 7441 Recup : 2479	3 Rec Epv : 83317 Pv : 26859 Def : 3153 Recup : 7536

Attention
les sub, attirails, amélio, etc. ne sont pas pris en compte ;)



```
134     @GemHeal.error
135     async def GemHeal_Error(ctx, error):
136         if isinstance(error, commands.MissingRequiredArgument):
137             await ctx.channel.send("Erreur, Ecrivez GemHeal suivi des PV de base, "
138                                     "de la Def de base et de la récup de base")
```

En cas d'erreur dans l'écriture de la commande, un message sera affiché.

 horotopia Aujourd'hui à 17:06
GemHeal 1

 Gally BOT Aujourd'hui à 17:06
Erreur, Ecrivez GemHeal suivi des PV de base, de la Def de base et de la récup de base

Autre exemple de commande :

```
182     @bot.command(help= "connaitre les PV et la nature du titan selon son niveau : Titan 203")
183     async def Titan(ctx: commands.Context, lvl: int, iterations=10):
184
185         nature = [ " :dark:", " :eau:", " :bois:", " :light:", " :feu:"]
186         pvTitan = [ "3", "3.25", "3.5", "3.75", "4", "5.5", "7", "8.5", "10", "12.5",
187                     "15", "17.5", "20", "27.5", "35", "42.5", "50", "62.5", "75",
188                     "87.5", "100", "125", "150", "175", "200", "225", "250", "275",
189                     "300", "325", "350", "375", "400", "425", "450", "475", "500",
190                     "525", "550", "575", "600"]
191         messageSortie =""
192         #creation lines
193         for i in range(iterations):
194             iterationLvl = lvl+i
195             indexNature = (iterationLvl%5)-1
196             indexPvTitan = int((iterationLvl-1)/5)
197             messageSortie += ("Titan " + str(iterationLvl) +
198                             str(nature[indexNature]) +
199                             " (" + str(pvTitan[indexPvTitan]) + "m) : ")
200             if i<iterations:
201                 messageSortie += "\n"
202             await ctx.channel.send(messageSortie)
203
204     @Titan.error
205     async def Titan_Error(ctx, error):
206         if isinstance(error, commands.MissingRequiredArgument):
207             await ctx.channel.send("Erreur, Ecrivez Titan "
208                                     "suivi du chiffre qui vous intéresse : Titan 21")
```

Cette commande permet d'afficher les prochains Boss, leur niveau, nature et Point de Vie (PV).

On peut décider du nombre d'itérations ou laisser 10 par défaut

L'une de mes premières fonctions les plus élaborées à ce moment là.

Ce que ce projet m'a apporté :

Ce projet n'a rien de professionnel. C'est juste les balbutiements de quelqu'un qui s'essaye à la programmation et qui demande à en savoir plus. Beaucoup d'essais, de nuits de recherches afin de trouver la moindre information qui permette d'avancer sur de nouvelles découvertes. Quelqu'un derrière son ordinateur fou de joie parce que son code lui répond ce qu'il faut. Obligé de chercher des moyens de travailler dessus de plusieurs postes à différents endroits, trouver un hébergeur gratuit parce que l'ordinateur de mon ami s'est éteint pour la dernière fois.

Ce morceau de code n'est pas fini et ne sera peut-être jamais le cas. Mais il a été le facteur déclencheur pour entrer dans le monde de la programmation.

J'ai pu y voir les bases que je n'avais pas compris en essayant le C et aller jusqu'à la compréhension des fonctions. J'ai essayé les classes sans vraiment y parvenir.

Ce qui pourrait être amélioré :

- De nouvelles implémentations
 - l'ajout d'une BDD
 - Finir la refactorisation
 - orienter le code en POO