Discord Bot - Aflevering

September 2022

I denne opgave skal i lave jeres egen Discord-bot fra bunden. Hvis i ikke kan finde på jeres egen bot skal i enten lave en linksaver, et spil eller en user-manager.

Hvis i vælger spillet kan i forsøge at genskabe jeres tekst-adventure som en discord-bot, eller implementere et "simpelt"spil som tic-tac-toe. User-manageren er svær fordi det kræver at man kigger meget i discord.pys dokumentation på egen hånd.

Krav til jeres bot

Jeres bot, skal overholde følgende krav:

- Der skal være en eller anden form for data, som botten holder styr på.

  Eksempel: I et spil skal der holdes styr på spillerens inventory eller boardets status.

- Brugere skal kunne indtaste flere forskellige kommandoer i Discord som f.eks. tilføjer data til botten, ændrer på bottens data, viser bottens data og/eller sletter data.

  Eksempel: I en linksaver kan en bruger indtaste en kommando for tilføje et link til en dictionary, en anden kommando for at slette et link, en tredje kommando for at vise links og en fjerde kommando for at redigere i et link.

Sørg for at der er mere end bare en enkelt dictionary med en enkelt kommando der opdaterer – i skal vise hvad i har lært!

Planlægning

Dette er et lidt større projekt, så for at gøre det overskueligt starter vi med at lave en plan for hvad botten skal kunne.

Man må gerne spørge om lov til at lave andre typer bots end dem der er foreslået.

Snak sammen i grupperne om hvordan botten skal bruges. Find ud af hvad bottens kommandoer skal være: hvad skal brugerne skrive på discord, og hvad

skal botten gøre som respons? Skriv ned hvad i finder på – i kan bruge det i jeres rapport.

Dette er også et godt tidspunkt at snakke om hvad det er for noget data botten skal holde styr på. Hvad har den brug for af information, og hvornår skal det opdateres? Skriv en liste af hvad for noget information botten skal holde styr på, og gem det så i kan bruge det i rapporten.

Det er ikke sikkert at i kommer til at følge denne her plan til punkt og prikke, men det skal gerne give jer et udgangspunkt, så i nemmere kan komme igang med at kode.

Tips til koden

1) Print-funktion

Når i skal kode jeres bot, så start med at lave en funktion der kan printe det data botten holder styr på, enten som en Discord-besked, eller bare til terminalen. Hvis i så har en funktion senere hen der ikke virker, så kan i altid kalde denne print-funktion, og se om jeres data er gemt rigtigt. For Emoji-tæller botten kunne man f.eks. have:

```
emoji_count = {}
def print_emoji_count():
  for emoji in emoji_count.keys():
    print(emoji + " : " + str(emoji_count[emoji]))
```

Hvis man så for eksempel har problemer med at få optælningen til at virke, kan man bruge denne funktion i `on_reaction_add`:

```
@client.event
async def on_reaction_add(reaction, user):
  # print tilstand inden vi har gjort noget
  print("REACTION ADD START:")
  print_emoji_count()

  # ============
  # Din kode her
  # ============

  # print tilstand efter aendringer, for at se om det
  # virkede
  print("REACTION ADD END:")
  print_emoji_count()
```

Det kode kan man så fjerne inden man afleverer – det findes kun for at gøre det nemmere at kode botten.

Nu kan i så tilføje én kommando ad gangen. Sørg for at teste koden hver gang i har lavet noget nyt. For kommandoer der opdaterer det data botten holder styr på, kan i bruge jeres print-funktion til at teste om de virker.

2) Struktur

Hvis man har en bot med mange kommandoer, så kan `on_message`-funktionen godt bliver ret lang og svær at finde rundt i. Derfor er det en god ide at dele sin kode op i flere funktioner, f.eks. en funktion for hver kommando. For eksempel kunne man lave en bot med en echo-funktion, en reverse-funktion, og en yell-funktion:

```
async def echo(text, channel):
  reply = "Du sendte: " + text
  await channel.send(reply)

async def reverse(text, channel):
  reply = "Omvendt: " + text[::-1]
  await channel.send(reply)

async def yell(text, channel):
  reply = text + "!"
  await channel.send(reply)

@client.event
async def on_message(message):
  contents = message.content
  channel = message.channel

  if contents.startswith("!echo "):
    await echo(contents[6:], channel)
  elif contents.startswith("!reverse "):
    await reverse(contents[9:], channel)
  elif contents.startswith("!yell "):
    await yell(contents[6:], channel)
```

3) Intents

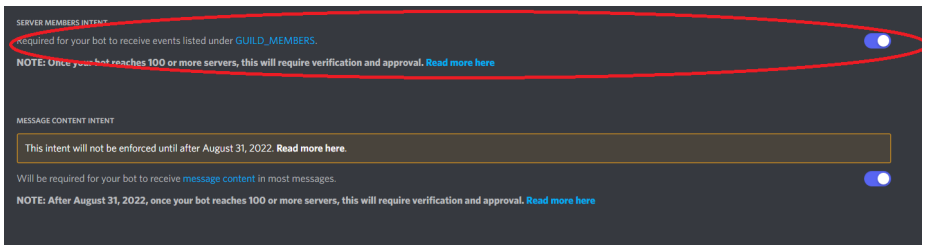
Dette tip er kun relevant for de af jer der arbejder med funktioner, udover dem som vi har brugt i øvelserne.

Hvis i skal tilføje funktioner som f.eks. on_reaction_remove(reaction, user) skal i sætte jeres `intents` anderledes. Dette skyldes at Discord kræver at bot-programmører tilkendegiver, hvilke former for data, de regner med at benytte i deres programmer. on_reaction_remove(reaction, user) kræver at members gemmes i cache og derfor skal man sætte intents.members til True.

Dette gøres således:

```
intents = discord.Intents.default()
#Message Content er med i den kode i fik i starten
intents.message_content = True
#Denne her skal tilfoejes
intents.members = True
client = discord.Client(intents=intents)
```

Members er det som Discord kalder et priviligeret intent og kræver derfor også at man giver tilladelse i sin Developer Portal (klik her).



I skal aflevere...

En zip-fil der indeholder:

- Kode til jeres discord-bot.

  Sørg for at det er nemt at bruge sit eget discord-token, f.eks. ved at erstatte en fil.

- En rapport på maks fem sider, som indeholder:
  - En problemformulering: Hvad er det for et problem jeres bot løser? Eller, hvordan gør den livet nemmere for dem der skal bruge den?
  - En beskrivelse af hvordan man bruger jeres bot. Dette skal være henvendt til en bruger af jeres bot – ved at læse dette afsnit skal jeg, uden at kigge på jeres kode, kunne invitere jeres bot til min discord-server og finde ud af at bruge den.

- Forklaring af koden. Sørg for at give et overblik over hvordan koden er struktureret, og fremhæv evt. særligt interessante dele af koden. I behøver ikke gennemgå alle linjer – det bliver for langt.

  Dette er et godt sted at skrive om hvad for noget data botten holder styr på, og hvornår det bliver opdateret.

- En konklusion: Fik i implementeret alt det i gerne ville? Fungerer botten som ønsket? Hvis ikke, hvad mangler så/hvad virker ikke? Hvad kunne forbedres? Hvad kunne i godt have tænkt jer at lave, men manglede viden om?