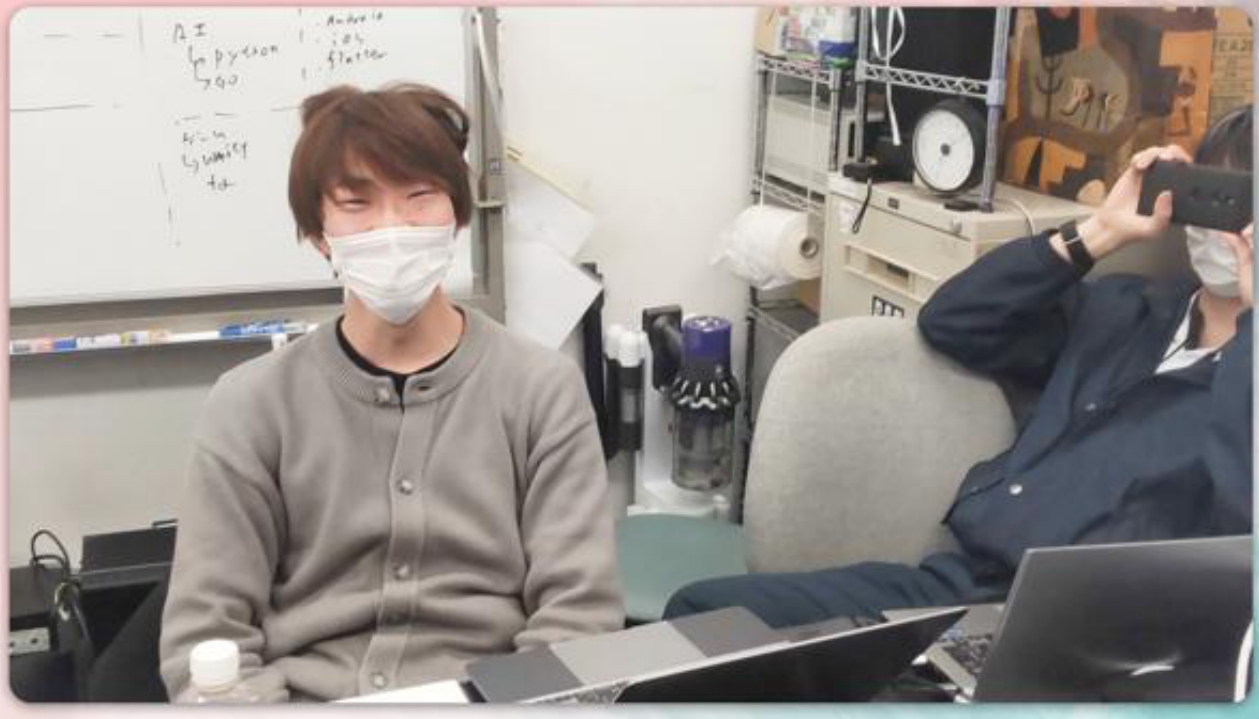
シス研の技術本
テスト作成
表紙



目次

1 これは chapter

1.1 これは section

我輩は猫である[*1]。

どこで生れたかとんと見当がつかぬ。何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している。吾輩はここで始めて人間というものを見た。しかもあとで聞くとそれは書生という人間中で一番獰悪な種族であったそうだ。この書生というのは時々我々を捕えて煮て食うという話である。

```
1  /* ここにはソースコードを書く */
2  #include<stdio.h>
3
4  int main(void)
5  {
```

[*1] こんな感じで脚注を書く

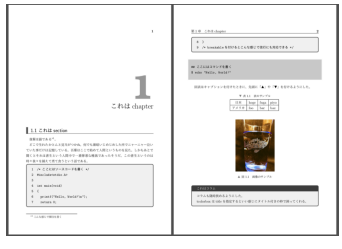
```
    printf("Hello, World!\n");
    return 0;
}
/* breakable を付けるとこんな感じで改行にも対応できる */
```

```
## ここにはコマンドを書く
$ echo "Hello, World!"
```

図表はキャプションを付けたときに、先頭に「▲」や「▼」を付けるようにした。

▼ 表 1.1　表のサンプル

日本	hoge	fuga	piyo
アメリカ	foo	bar	baz



▲ 図 1.1　画像のサンプル

これはコラム

コラムも随時挟めるようにした。
tcolorbox は title を指定するといい感じにタイトル付きの枠で囲ってくれる。

2

DiscordBot を作ってみよう!

2.1 DiscordBot を作ってみよう

2.1.1 はじめに

はじめまして、suda です。私は Discord で人とチャットをしている時に同じ会話が頻繁に続き、これ Bot で返事をするようにしたら返事をする手間が省けるし面白いのでは？と思い Bot を作ることにしました。発想がひどいって！？まあでも自分の発想したものを形にすることが面白いことだと思うので今回はそこには目を瞑りましょう…もちろん自分が送ったメッセージに対して Bot に返答させることもできるので、自分だけのオリジナル DiscordBot を作ってみましょう！

2.1.2 何を作るのか

Discord のサーバで特定のメッセージが来たら、特定のメッセージを返す Discord の Bot を作ります。サンプルプログラムを参照したい方は以下の URL からご覧下さ

い。[*1] 例えば自分が「仕事終わった」と言うと Bot が「お疲れ様」と返してくれます。

2.2 実行環境・使用技術

- Python 3.10.8

2.3 ローカル環境で Bot が動作するようにする

まずはローカル環境で Bot が動作するようにしてみます。

2.3.1 Bot の作成・管理をする

初めに、機能などはまだついていない Bot を Discord のポータルサイトから作成します。Discord の Bot の作り方 (メモ) という記事の「1.Discord 上の Bot の作成」を見ながら Bot を作成してみて下さい。[*2]

2.3.2 ファイルの作成

Bot を実行する Python ファイルを作ります。

```
mkdir message_discord_bot
cd message_discord_bot
touch main.py
```

[*1] 今回作る DiscordBot のサンプルプログラム https://github.com/sudamichiyo/Discord_Bot_sampleprogram

[*2] Discord の Bot の作り方 (メモ)https://note.com/exteoi/n/nf1c37cb26c41(参照 2023.3.29)

2.3.3 discord.py の準備

ここからは discord.py のドキュメントを見ながら環境構築をしていきます。[*3] Python で Discord の API を操作するために必要なライブラリをインストールします。先ほど作成したディレクトリにアクセスして、以下のコマンドで discord.py をインストールします。

```
python -m pip install -U discord.py
```

次に、先ほど作成した main.py を以下のソースコードに書き換えます。

```
1 import discord
2
3 class MyClient(discord.Client):
4     async def on_ready(self):
5         print(f'Logged on as {self.user}!')
6
7     async def on_message(self, message):
8         print(f'Message from {message.author}
9                                         : {message.content}')
10
11 intents = discord.Intents.default()
12 intents.message_content = True
13
14 client = MyClient(intents=intents)
15 client.run('my token goes here')
```

ここで、以下のボットに関する 2 つの設定を Discord のポータルサイトから設定してください。

[*3] discord.py ドキュメント https://discordpy.readthedocs.io/ja/latest/intro.html#basic-concepts(参照 2023.3.29)

- ポータルサイトの「Bot」からトークンを取得する
- ポータルサイトの「Bot」の「MESSAGE CONTENT INTENT」を有効にする

'my token goes here' は取得した Bot のアクセストークンを書きます。以上の設定が終わったところで python3 main.py を実行すると、Bot のサーバが立ち上がります。Bot サーバ起動後に Bot のいるサーバでメッセージを投げると、コマンドライン上に「書いた人」と「メッセージ」がそのまま出力されます。

2.3.4 環境変数の設定

ソースコードに直接トークンを書いてしまうと、Github でソースコードをホスティングするときにトークンキーが他の人にバレてしまいます。これを防ぐために.env ファイルを作成して、その中に Discord のアクセストークンを書きます（下記参照）。

```
1 DISCORD_TOKEN='My token goes here'
```

Python の中で.env ファイルに書かれている変数を取得するために dotenv というライブラリを使用します。以下のようにインストールします。

```
pip install python-dotenv
```

インストール後に main.py に下記のコードを付け加えて下さい。
main.py の import discord と class MyClient の間に以下のコードを追加します。

```
1 import os
2 from dotenv import load_dotenv
3 load_dotenv()
```

そして、最後の行を以下のように書き換えて下さい。

```
1 client.run(os.environ['DISCORD_TOKEN'])
```

書き換えたあとに python3 main.py を実行すると，先ほどと同じようにメッセージの受け取りをしてくれるサーバーサイドアプリケーションが立ち上がります。

2.3.5 Bot が特定のワードに反応して、特定のメッセージを返答する機能をつける

プログラムを起動して正常にサーバーサイドアプリケーションがメッセージを受け取れるようになったら、Bot が特定のワードに反応して、特定のメッセージを返答する機能をつけていきます。Bot に機能をつけるには上記のソースコードの 8 行目と 10 行目の間に以下のコードを付け足していきます。

```
1 # メッセージを書いた人が Bot なら処理終了
2 if message.author.bot:
3     return
4 channel = message.channel
5 if message.content == '仕事終わった':
6     await channel.send('お疲れ様')
```

付け足したコードの解説をしていきます。
2,3 行目でメッセージを書いた人が Bot なら処理を終了させています。
4 行目でメッセージが投稿されたチャンネル取得しています。
5 行目の message.content はメッセージの内容で、今回の場合「仕事終わった」というメッセージをチャンネルに投稿すると、メッセージが投稿されたチャンネルに Bot が「お疲れ様」と返答します。

2.4 まとめ

今回は Discord の Bot の作り方を説明しました。上記の「仕事終わった」や「お疲れ様」に当たる部分を変えたりして自分好みに改良してみて下さい。

3 これは chapter

3.1 これは section

我輩は猫である[*1]。

どこで生れたかとんと見当がつかぬ。何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している。吾輩はここで始めて人間というものを見た。しかもあとで聞くとそれは書生という人間中で一番獰悪な種族であったそうだ。この書生というのは時々我々を捕えて煮て食うという話である。

```
1  /* ここにはソースコードを書く */
2  #include<stdio.h>
3
4  int main(void)
5  {
```

[*1] こんな感じで脚注を書く

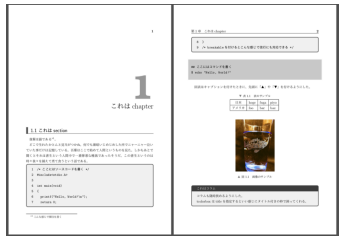
```
6    printf("Hello, World!\n");
7    return 0;
8  }
9  /* breakable を付けるとこんな感じで改行にも対応できる */
```

```
## ここにはコマンドを書く
$ echo "Hello, World!"
```

図表はキャプションを付けたときに、先頭に「▲」や「▼」を付けるようにした。

▼ 表 3.1　表のサンプル

日本	hoge	fuga	piyo
アメリカ	foo	bar	baz



▲ 図 3.1　画像のサンプル

これはコラム

コラムも随時挟めるようにした。
tcolorbox は title を指定するといい感じにタイトル付きの枠で囲ってくれる。

4

これは chapter

4.1 これは section

我輩は猫である[*1]。

どこで生れたかとんと見当がつかぬ。何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している。吾輩はここで始めて人間というものを見た。しかもあとで聞くとそれは書生という人間中で一番獰悪な種族であったそうだ。この書生というのは時々我々を捕えて煮て食うという話である。

```
1  /* ここにはソースコードを書く */
2  #include<stdio.h>
3
4  int main(void)
5  {
```

[*1] こんな感じで脚注を書く

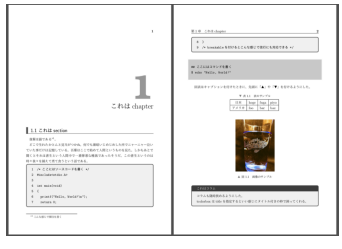
```
    printf("Hello, World!\n");
    return 0;
}
/* breakable を付けるとこんな感じで改行にも対応できる */
```

```
## ここにはコマンドを書く
$ echo "Hello, World!"
```

図表はキャプションを付けたときに、先頭に「▲」や「▼」を付けるようにした。

▼ 表 4.1　表のサンプル

日本	hoge	fuga	piyo
アメリカ	foo	bar	baz



▲ 図 4.1　画像のサンプル

これはコラム

コラムも随時挟めるようにした。
tcolorbox は title を指定するといい感じにタイトル付きの枠で囲ってくれる。

5 これは chapter

5.1 これは section

我輩は猫である[*1]。

どこで生れたかとんと見当がつかぬ。何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している。吾輩はここで始めて人間というものを見た。しかもあとで聞くとそれは書生という人間中で一番獰悪な種族であったそうだ。この書生というのは時々我々を捕えて煮て食うという話である。

```
1  /* ここにはソースコードを書く */
2  #include<stdio.h>
3
4  int main(void)
5  {
```

[*1] こんな感じで脚注を書く

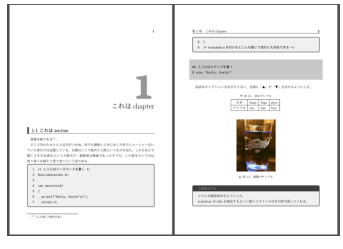
```
    printf("Hello, World!\n");
    return 0;
}
/* breakable を付けるとこんな感じで改行にも対応できる */
```

```
## ここにはコマンドを書く
$ echo "Hello, World!"
```

図表はキャプションを付けたときに、先頭に「▲」や「▼」を付けるようにした。

▼ 表 5.1　表のサンプル

日本	hoge	fuga	piyo
アメリカ	foo	bar	baz



▲ 図 5.1　画像のサンプル

これはコラム

コラムも随時挟めるようにした。
tcolorbox は title を指定するといい感じにタイトル付きの枠で囲ってくれる。

6 戦闘機に乗ろう

6.1 DCS World

6.1.1 はじめに

はじめまして、松土です。いきなりですが、僕はミリオタ[*1]で、特に戦闘機が好きです。僕は戦闘機に乗りたい！！！。しかし、戦闘機というのは皆さんもご存知の通り、軍用の航空機のため主に軍隊が保有しており、日本では航空自衛隊のみが保有しています。そんな戦闘機に乗ろうと思ったら、航空自衛隊に入隊し、高い倍率の選抜を受けた後に何年もの厳しい訓練を乗り越えないといけません。そこで、一度は耳にしたことであろう、フライトシミュレーターというのがこの世には存在しており、これは一般人が仮想のコクピットに乗り込み、操縦をシミュレートすることができる素晴らしいものです。本物のパイロットが実機を操縦する前の訓練をするにあたって使用する事もあります。僕はこれを利用して戦闘機に乗りたい欲を解消することにしました。

[*1] ミリタリーオタクの略　所謂軍事が好きな人

6.1.2　DCS World とは

いきなり出てきた DCS World[*2]とは何か？、これは戦闘機に特化したフライトシミュレーターで、フランス空軍にも採用されている大変優れたものです。現実に存在する戦闘機がいくつかモジュールとしてあり、一般人が家庭でパソコンの中にインストールするだけで、実機を仮想状で操縦できるようになります。プラットフォームは Microsoft Windows で、2008 年からリリースされており、開発元は本社をロシアのモスクワに置いている Eagle Dynamics 社です。

＞＞＞アプリは無料です＜＜＜　※モジュールは有料です

6.2 動作環境

6.2.1　最小構成

- OS: 64-bit Windows 10 , DirectX11（version 2.7 以降は Windows 7 非対応かつ Windows 8 が明記落ち、version 2.5.x までは Windows 7/8 も可）
- CPU: Intel Core i3 2.8 GHz / AMD FX
- RAM: 8 GB (重いミッションをプレイする場合: 16 GB)
- HDD 空き容量: 60 GB (※注：2022 年夏時点の version2.7.16 では 120GB に増加している)
- ビデオカード: NVIDIA GeForce GTX 760 / AMD R9 280X 以上
- ユーザー認証用インターネット接続

6.2.2　推奨構成

- OS: 64-bit Windows 10 , DirectX11（version 2.7 以降は Windows 8 が明記落ち、version 2.5.x までは Windows 8 も推奨内）
- CPU: Core i5 3GHz 以上 / AMD FX 又は Ryzen
- RAM: 16 GB (重いミッションをプレイする場合: 32 GB)
- HDD 空き容量: 130 GB (SSD 推奨　全モジュール導入には 460GB)
- ビデオカード: NVIDIA GeForce GTX 1070 / AMD Radeon RX VEGA 56

[*2] Degital combat simulator world の略

(8GB VRAM) 以上
- ジョイスティック
- ユーザー認証用インターネット接続

6.2.3 VR を使用する場合

推奨構成に上書きして

- CPU: Core i5 3GHz 以上 / AMD FX 又は Ryzen
- RAM: 16 GB (重いミッションをプレイする場合: 32 GB)
- HDD 空き容量: 130 GB (SSD 推奨　全モジュール導入には 460GB)
- ビデオカード: NVIDIA GeForce GTX 1080 / AMD Radeon RX VEGA 64 (8GB VRAM) 以上

6.3 導入

6.3.1 公式版と Steam 版

公式 DCS Wolrd のサイトよりダウンロードする場合とゲームプラットフォームで有名な Steam でダウンロードする場合があります。特に両者違いはありませんが、Steam 版は公式版よりも更新が遅いことがあります。

6.3.2 安定版と Open Beta 版

新しいモジュールはまず Open Beta 版のみに対して提供され、何度かの Open Beta アップデートを経て十分にバグを無くしてから安定版への提供となります。

- Open Beta の利点

マルチプレイサーバーの多くは Open Beta を使用している (2018/8/30 現在) ためマルチプレイをしたいのなら Open Beta 推奨新しいモジュールをより早く遊ぶことができるレーダーや FLIR など、前バージョンではモジュールへの実装がオミットされていた一部機能や兵装が追加実装されたのを早く体験できる

- Open Beta の欠点

Open Beta は当然バグが多く、特定の武器を使うとゲームがクラッシュする現象が起きることもあるゲームがクラッシュまではしなくても、以前のバージョンでは正常だったはずの特定のミサイルの誘導能力がおかしくなっている、レーダーや照準などの装置の操作や挙動がおかしくなっている、といったバグが新しく増えていることもある

6.3.3　モジュールの購入

アプリ自体は無料のため、インストール後起動し、プレイすることが可能ですが、初期で乗ることのできる機体は、第二次世界大戦で使用されたプロペラ機の練習機版 TF-51 とソ連[*3]が開発した攻撃機 Su-25 だけです。どちらも、戦闘機ではない上にカッコ悪い[*4]です。

注意として、機体内部のボタンやスイッチを一つ一つまで操作できる機体（クリッカブル機）と、キーボードを使い、キーで細かい機体の操作を行う機体（FC3 機）があり、クリッカブル機は基本的に高価だが、実機と同じく全てのボタンが操作でき、リアルな操作を楽しむことができる。

6.4 まとめ

今回は、ミリオタ向けフライトシミュレーターとして、DCS World を紹介させていただきました。導入した後の楽しみ方は、実機と同じエンジンスタートを行ってみたり、マルチプレイで友人と飛んでみたり、または戦ってみたり、と様々であり、あなたの思うがままにやりたいことができるのがこの DCS World の良いところです。もしも、これを読んでいるあなたが DCS World を初めたら、僕と一緒に飛びましょう。

[*3] ソビエト連邦の略

[*4] あくまで個人の感想です

7 これは chapter

7.1 これは section

我輩は猫である[*1]。

どこで生れたかとんと見当がつかぬ。何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している。吾輩はここで始めて人間というものを見た。しかもあとで聞くとそれは書生という人間中で一番獰悪な種族であったそうだ。この書生というのは時々我々を捕えて煮て食うという話である。

```
1  /* ここにはソースコードを書く */
2  #include<stdio.h>
3
4  int main(void)
5  {
```

[*1] こんな感じで脚注を書く

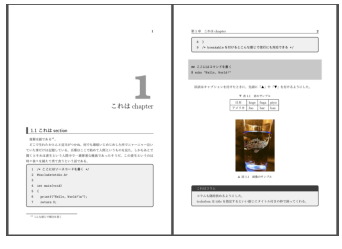
```
    printf("Hello, World!\n");
    return 0;
}
/* breakable を付けるとこんな感じで改行にも対応できる */
```

```
## ここにはコマンドを書く
$ echo "Hello, World!"
```

図表はキャプションを付けたときに、先頭に「▲」や「▼」を付けるようにした。

▼ 表 7.1　表のサンプル

日本	hoge	fuga	piyo
アメリカ	foo	bar	baz



▲ 図 7.1　画像のサンプル

これはコラム

コラムも随時挟めるようにした。
tcolorbox は title を指定するといい感じにタイトル付きの枠で囲ってくれる。

8 これは chapter

8.1 これは section

我輩は猫である[*1]。

どこで生れたかとんと見当がつかぬ。何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している。吾輩はここで始めて人間というものを見た。しかもあとで聞くとそれは書生という人間中で一番獰悪な種族であったそうだ。この書生というのは時々我々を捕えて煮て食うという話である。

```
1  /* ここにはソースコードを書く */
2  #include<stdio.h>
3
4  int main(void)
5  {
```

[*1] こんな感じで脚注を書く

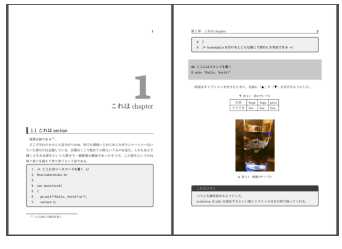
```
6    printf("Hello, World!\n");
7    return 0;
8  }
9  /* breakable を付けるとこんな感じで改行にも対応できる */
```

```
## ここにはコマンドを書く
$ echo "Hello, World!"
```

図表はキャプションを付けたときに、先頭に「▲」や「▼」を付けるようにした。

▼ 表 8.1　表のサンプル

日本	hoge	fuga	piyo
アメリカ	foo	bar	baz



▲ 図 8.1　画像のサンプル

これはコラム

コラムも随時挟めるようにした。
tcolorbox は title を指定するといい感じにタイトル付きの枠で囲ってくれる。