Creating a Discord Bot

Bots are a popular feature on Discord for enhancing the server experience. Users can program bots to perform various tasks like playing music in voice calls, displaying user statistics, answering questions, streamlining moderation, and more.

In this guide, you will learn how to (1) create a Discord bot user, (2) configure your bot settings, (3) invite the bot to your server, and (4) program a simple greeting bot using the Python programming language. This guide is for Discord users with at least minimal experience in Python and any IDE or text editor already installed.

Creating a bot user

Like users, bots also need Discord accounts to communicate and perform tasks on the platform. The *Discord Developer Portal* is where you create and manage settings and profile data for your custom Discord applications. Inside an application, you can create a *bot user* that serves as a visible Discord account that the bot can use.

To create an application with a bot user:

1. Go to your Discord Developer Portal. Your home page will include a list of your applications (if any) and tabs that link to teams and developer documentation.
2. In the Applications tab, select **New Application**. A **Create an Application** popup will appear.
3. Type the name of your bot and select the checkbox to agree to Discord's Developer Terms of Service and Developer Policy.
4. Select **Create**. Your application is now created, and you will be redirected to your application's settings.
5. On the sidebar, select the **Bot** tab.
6. Select **Add Bot**. A popup will appear to warn you that creating a bot user is irreversible. Select **Yes, do it!** Your bot user has been created.
    a. If you have 2FA enabled on Discord, another popup may appear asking for a 6-digit authentication code. Enter the code and select **Submit**.

Note to Instructor: This guide is for Discord users looking to create a simple Discord bot. The audience should already be comfortable with Discord and have an account, and should already have at least basic experience with Python and any IDE or text editor that is already installed. This guide does not require previous knowledge of bot and API concepts like scopes, tokens, and authentication.

Configuring bot settings

After creating the bot user, the Bot tab now becomes your bot's settings page. Here you can customize display info and configure permissions and intents. *Intents* refer to groups of related events that serve a specific function. Most bots, including your greeting bot, need intents that can read and send messages and retrieve server member data. To enable these functions:

1. Turn on the **Server Members Intent** toggle switch, enabling access to certain events involving server member data.
2. Turn on the **Message Content Intent** toggle switch. Now the bot can receive most message content.
3. Under **Bot Permissions**, select the **Send Messages** and **Read Messages/View Messages** checkboxes.

Inviting the bot to your server

Your bot is ready to join a server for testing and release. To invite the bot to your server:

1. In the Developer Portal, select the **OAuth2** tab.
2. Under Scopes, select **bot**. This scope allows the bot to access Discord servers. See the *OAuth2 Scopes* section in the OAuth2 documentation to learn more about the different scopes.
3. Under Bot Permissions, select the **Read Messages/View Channels** and **Send Messages** checkboxes.
4. Copy the generated URL and paste it into the search bar.
5. In the **Add to Server** dropdown, select the server you want to invite the bot to.

6. Select **Authorize**, then select the **I am a human** checkbox. Your bot now appears as an offline member of your server (see Figure 1).

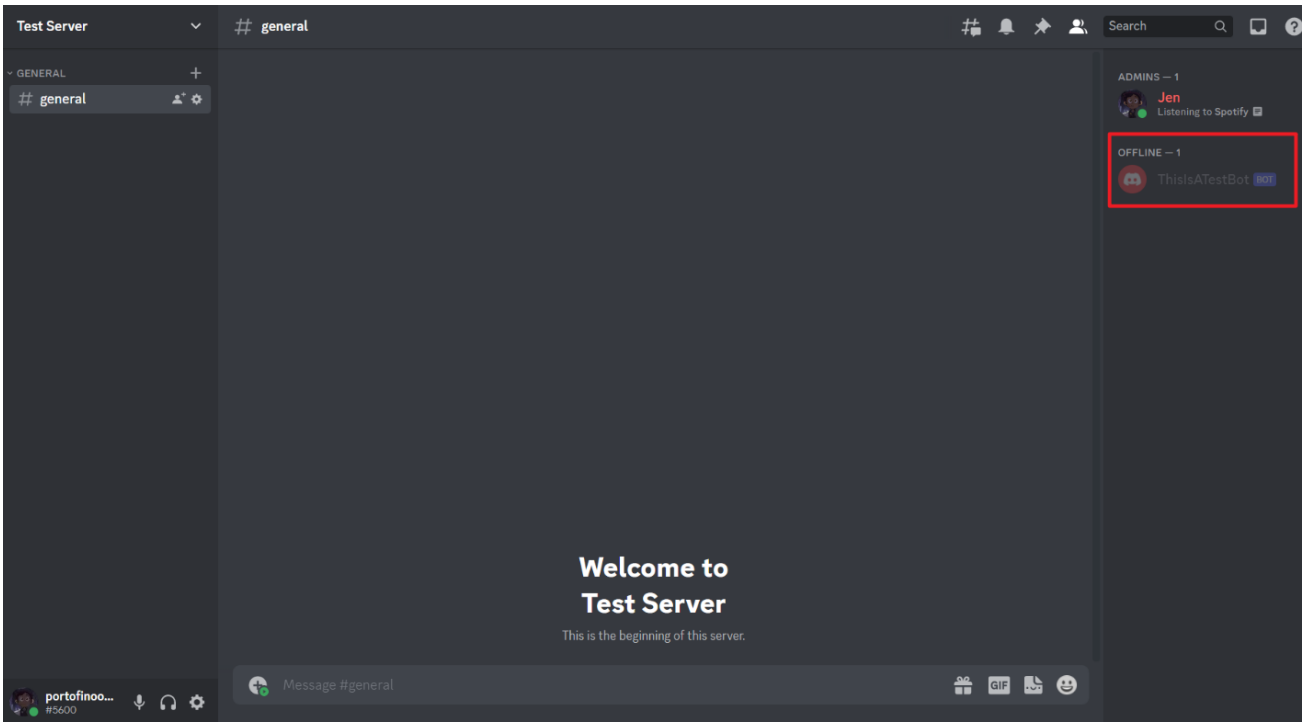

Figure 1: In the example here, the bot user called "ThisIsATestBot" is now an offline member of "Test Server."

Programming a greeting bot

Your bot will stay offline and idle until a program takes control and defines the bot's functions and commands. Regardless of complexity or purpose, most Python bot programs are structured similarly and require the following items:

- **discord.py**: Python library specifically for creating Discord bots. To install discord.py, type `pip install -u discord.py` into the Command Prompt.
- **Intents settings:** Though you previously configured intents for the bot user in the Developer Portal, the program also needs to know which intents they can access.
- **Log-in event:** Event trigger that prints to the console that the program has successfully logged into the bot.
- **Commands**: Pre-defined commands the bot performs whenever a user types it. Commands are formatted with an assigned prefix like **!** or **?** to alert the bot of an incoming command, followed by the command name and any arguments.
- **A token**: A string that serves as the bot's password. To view your bot's token, select **View Token** in the Bot tab of the Developer Portal.

  Caution: Do not share this token. Like passwords, other people can use this token to log in and inject malicious code into your bot.

Your first bot will be a simple greeting bot with the command **!hello** that greets the user every time they execute the command. To write and run the greeting bot:

1. Create a new project in your chosen IDE or text editor.
2. In your project, create a python file. Write the following code in this file, replacing **'your token'** with your Discord bot's token.

```
# Import discord.py and command functions
import discord
from discord.ext import commands

# Set default permissions/intents
intents = discord.Intents.default()
intents.members = True
intents.message_content = True

# Assign '!' as the command prefix.
bot = commands.Bot(command_prefix='!', intents=intents)

# Trigger that prints to the console every time the bot runs.
@bot.event
async def on_ready():
    print(f'Logged in as {bot.user}')

# Command named "hello" that greets the user who executed the command.
@bot.command()
async def hello(ctx):
    await ctx.send("Hello " + ctx.message.author.mention)

bot.run('your token')# Takes control of the bot using its token.
```

3. Select **Run**. Your bot will now appear online.
4. In the server that your bot is in, type **!hello** into the chat. The bot should respond "Hello" followed by a mention of your username as shown in Figure 2.

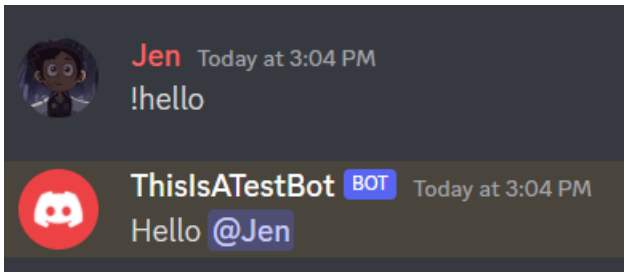

Figure 2: The !hello command in action.

To program bots with more advanced features like playing music, see the official discord.py documentation.