Set Up a telegram bot: -

Creating a Telegram bot involves several steps. Below, I'll outline a detailed guide to help you get started:

Step 1: Create a New Bot on Telegram
Create a Telegram Account: If you don't have a Telegram account yet, download the app and create one.

Find the **BotFather:** Search for "BotFather" in the Telegram app and start a chat with it. BotFather is the official bot that helps you create and manage your Telegram bots.

Create a New Bot:

Send /newbot to the BotFather.
Follow the prompts to **name your bot** and create a username for it. The username must end in "bot" (e.g., MyFirstBot or MyFirst_Bot).
Once the bot is created, BotFather will give you a token. This token is important as it allows you to interact with the **Telegram API**.

Step 2: Set Up Your Development Environment
Choose a Programming Language: Telegram bots can be developed using various programming languages, but Python is a popular choice due to its simplicity and the availability of libraries.

Install Python: Ensure you have Python installed on your machine. You can download it from python.org.

Install the python-telegram-bot Library: This library provides a wrapper for the Telegram Bot API.

bash

pip install python-telegram-bot

Step 3: Write Your Bot Code

Create a New Python File: Create a new file, e.g., my_bot.py.

Import Required Libraries:

python

```
from telegram import Update
from telegram.ext import Updater, CommandHandler, MessageHandler, Filters,
CallbackContext
```

Define Your Bot Functions:

python

```
def start(update: Update, context: CallbackContext) -> None:
    update.message.reply_text('Hello! I am your bot. How can I help you?')

def help_command(update: Update, context: CallbackContext) -> None:
    update.message.reply_text('Help!')

def echo(update: Update, context: CallbackContext) -> None:
    update.message.reply_text(update.message.text)
```

Set Up the Updater and Dispatcher:

```
python
def main():
    # Replace 'YOUR_TOKEN' with your actual token
    updater = Updater("YOUR_TOKEN")

    dispatcher = updater.dispatcher

    dispatcher.add_handler(CommandHandler("start", start))
    dispatcher.add_handler(CommandHandler("help", help_command))
    dispatcher.add_handler(MessageHandler(Filters.text & ~Filters.command, echo))

    updater.start_polling()

    updater.idle()

if __name__ == '__main__':
    main()
```

Step 4: Run Your Bot

Run your bot by executing the Python script:

bash

python my_bot.py

Step 5: Interact with Your Bot
Go to Telegram and search for your bot using the username you created. Start a chat and test the commands /start and /help. Your bot should respond according to the functions you defined.

Code :-

```
import telebot
from datetime import datetime
import logging
Import os

# Replace 'YOUR_TOKEN' with your actual bot token
TOKEN = "YOUR_TOKEN"
bot = telebot.TeleBot(TOKEN)

# Set up logging
logging.basicConfig(level=logging.INFO)

################################################################################
# Handler for the /start command

@bot.message_handler(commands=['start'])
def send_welcome(message):
    # Path to your local GIF file
    gif_path = '/Pkp.gif'

    try:
        # Open the GIF file
        with open(gif_path, 'rb') as gif_file:
```

```
        # Send the welcome message with media
        bot.send_chat_action(message.chat.id, 'upload_photo')
```

##

Or

At Website hosting , **After Importing os**

Handler for the /start command

```
@bot.message_handler(commands=['start'])

def send_welcome(message):

    # Adjust the path to Pkp.gif based on its location in PythonAnywhere
    gif_filename = 'Pkp.gif'
    gif_path = os.path.join('/home/darkcheveyo123/mysite', gif_filename)

    try:
        # Open the GIF file
        with open(gif_path, 'rb') as gif_file:
            # Send the welcome message with media
            bot.send_chat_action(message.chat.id, 'upload_photo')
```

Action to indicate bot is uploading photo

```
            bot.send_document(message.chat.id, gif_file, caption="Welcome to
dark_cheveyoBot! Use /help to see available commands.")

            logging.info(f"Sent welcome message to {message.chat.id}")

    except Exception as e:
        logging.error(f"Error sending welcome message: {e}")
```

Action to indicate bot is uploading photo

```
            bot.send_document(message.chat.id, gif_file, caption="Welcome to
dark_cheveyoBot! Use /help to see available commands.")
            logging.info(f"Sent welcome message to {message.chat.id}")
    except Exception as e:
        logging.error(f"Error sending welcome message: {e}")
```

Handler for the /help command

```
@bot.message_handler(commands=['help'])
def send_help(message):
    help_text = (
        "/start - Start dark_cheveyoBot\n"
        "/help - Show help\n"
        "/info - Get information\n"
        "/status - Get status update\n"
        "/time - Get current time\n"
        "/datascience - Get data science resources"
    )
    bot.reply_to(message, help_text)
    logging.info(f"Handled /help command from {message.chat.id}")

# Handler for the /info command

@bot.message_handler(commands=['info'])
def send_info(message):
    bot.reply_to(message, "This is a simple Telegram bot created for demonstration
purposes.")
    logging.info(f"Handled /info command from {message.chat.id}")

# Handler for the /status command

@bot.message_handler(commands=['status'])
def send_status(message):
    bot.reply_to(message, "All systems are operational!")
    logging.info(f"Handled /status command from {message.chat.id}")

# Handler for the /time command

@bot.message_handler(commands=['time'])
def send_time(message):
    now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    bot.reply_to(message, f"The current time is: {now}")
    logging.info(f"Handled /time command from {message.chat.id}")


# Handler for the /datascience command

@bot.message_handler(commands=['datascience'])
def send_data_science_resources(message):
    resources = (
        "Here are some useful data science resources:\n"
        "1. [Data Science Portfolio-KaggleDataset](https://t.me/DataPortfolio)\n"
```

```
        "2. [Programming for Data Science_365](https://t.me/data_science_365)\n"
        "3. [Machine Learning books and papers](https://t.me/Machine_learn)\n"
        "4. [Next-Gen Data Scientists](https://t.me/nextgendatascientists)\n"
        "5. [Data Science AI Project](https://t.me/pythonspecialist)\n"
    )
    bot.reply_to(message, resources, parse_mode='Markdown')
    logging.info(f"Handled /datascience command from {message.chat.id}")

# Handler for unrecognized commands

@bot.message_handler(func=lambda message: True)
def handle_invalid_command(message):
    bot.reply_to(message, "Invalid command. Use /help to see the list of available commands.")
    logging.info(f"Handled invalid command from {message.chat.id}")

# Start the bot

try:
    logging.info("Bot is polling...")
    bot.polling()
except Exception as e:
    logging.error(f"Error in bot polling: {e}")
```

Free hosting providers where you can host your Telegram bot and keep it operational 24/7:

1. **Heroku**:
   - Heroku offers a free tier with limitations, but it can be suitable for hosting small-scale applications like Telegram bots.
   - Provides a straightforward deployment process for Python applications.
   - URL: Heroku
2. **PythonAnywhere**:
   - PythonAnywhere provides free and paid plans for hosting Python applications.
   - It offers a web-based Python development environment with consoles and file browsers.
   - URL: PythonAnywhere
3. **Glitch**:
   - Glitch allows you to host Node.js applications (including bots built with Telegraf or other Node.js libraries).
   - It has a free tier with community and collaborative features.

    - URL: Glitch
4. **Google Cloud Platform (GCP)**:
    - Google Cloud Platform offers a free tier with $300 in credits for new users, valid for one year.
    - It provides Compute Engine instances where you can host your bot.
    - URL: Google Cloud Platform
5. **AWS Free Tier**:
    - Amazon Web Services (AWS) offers a free tier for new users, which includes limited usage of various services including EC2 (Elastic Compute Cloud) instances.
    - URL: AWS Free Tier
6. **DigitalOcean**:
    - DigitalOcean provides a $100 credit for new users, valid for 60 days, which you can use to host a VPS (Virtual Private Server) instance to run your bot.
    - URL: DigitalOcean

Considerations:

- **Resource Limits**: Free tiers often come with resource limits (like CPU, memory, and uptime restrictions). Ensure your bot's requirements fit within these limits.
- **Terms and Conditions**: Review the terms and conditions of each provider to understand any restrictions or usage policies.
- **Maintenance**: Regularly check on your bot and the hosting platform to ensure it remains operational.

Choose a provider that best fits your technical skills and the requirements of your Telegram bot application. Each platform has its strengths and may be better suited depending on factors like language support, ease of use, and scalability options beyond the free tier.

PythonAnywhere :-

Using PythonAnywhere to host your Python Telegram bot is straightforward. Here's a step-by-step guide to get started:

Step 1: Sign Up and Log In

1. **Sign Up**: Go to PythonAnywhere and sign up for a free account.
2. **Log In**: Once signed up, log in to your PythonAnywhere account.

Step 2: Set Up Your Python Virtual Environment (Optional but Recommended)

1. **Dashboard**: After logging in, you'll land on the dashboard. Click on the "Dashboard" menu item to go to your dashboard if you're not already there.
2. **Open a Bash Console**: Click on the "Consoles" tab and then click "Bash". This will open a terminal-like console where you can run commands.

Create a Virtual Environment: If you want to use a virtual environment (recommended for managing dependencies), create one by running:
bash

```
mkvirtualenv mybotenv --python=/usr/bin/python3.8
```

3. Replace `mybotenv` with your preferred virtual environment name, and adjust the Python version (`/usr/bin/python3.8`) if needed.

Activate the Virtual Environment: Activate your virtual environment:
bash

```
workon mybotenv
```

4. Replace `mybotenv` with your virtual environment name.

Step 3: Upload Your Bot Files

1. **Files**: Click on the "Files" tab. Here, you can upload your bot's Python script and any other necessary files (like images, configuration files, etc.).
2. **Upload**: Click on the "Upload a file" button to upload your Python script (`telebot_bot.py` or whatever you named it).

Step 4: Install Required Packages

1. **Run Bash Console**: Go back to the Bash console if you closed it or opened a new one.

Install Dependencies: Install any dependencies your bot requires using `pip`. For example:
bash

```
pip install pyTelegramBotAPI
```

Step 5: Run Your Bot

1. **Run Your Script**: In the Bash console, navigate to the directory where your bot script is located (use `cd` command to change directory).

Run the Bot: Run your bot script:
bash

```
python telebot_bot.py
```

2. Replace `telebot_bot.py` with your actual bot script filename.

Step 6: Keep Your Bot Running

1. **Keep Console Open**: To keep your bot running continuously, keep the console open. If you close the console, your script will stop running.
2. **Scheduled Tasks (Optional)**: PythonAnywhere also allows you to schedule tasks to run at specific times. You can use this feature to run your bot script periodically if needed.

Step 7: Monitor and Manage

1. **Logs**: PythonAnywhere provides logs that you can view to monitor the activity of your bot and debug any issues.
2. **Dashboard**: Use the PythonAnywhere dashboard to manage your files, consoles, scheduled tasks, and other aspects of your account.

Step 8: Troubleshooting

1. **Debugging**: If your bot encounters issues, check the logs for error messages. PythonAnywhere's support documentation and community forums are also helpful resources for troubleshooting.

Additional Tips

- **Environment Variables**: If your bot requires API keys or other sensitive information, use environment variables to store them securely. PythonAnywhere allows you to set environment variables in the "Web" tab of the dashboard.
- **Upgrade Options**: If your bot needs more resources than the free tier provides (e.g., more CPU time, storage), consider upgrading to a paid plan on PythonAnywhere.

By following these steps, you should be able to successfully host your Python Telegram bot on PythonAnywhere and keep it operational 24/7. Adjust the instructions as per your specific bot's requirements and PythonAnywhere's interface updates.

https://**www.pythonanywhere.com**/user/*username*/files/home/*username*/mysite/tgbot.py?edit