Using python to send automatic messages directly scrapped from websites.

Introduction:

In today's data-driven world, accessing information from the web has become an essential part of various projects. In this article, we'll explore how to leverage Python to scrape data from a website and seamlessly send it to a Telegram bot. This integration can be incredibly powerful, providing real-time updates and insights to users right within their Telegram chat.

Getting Started with Web Scraping:

1. **Choose a Website:**
   Identify the website from which you want to scrape data. Ensure that you are compliant with the website's terms of service and respect its policies.
2. **Install Required Libraries:**
   Use Python's powerful libraries for web scraping, such as `requests` for making HTTP requests and `BeautifulSoup` for parsing HTML. Install them using the following:

   ```
   pip install requests beautifulsoup4
   ```

3. **Inspect the Website's Structure:**
   Use your browser's developer tools to inspect the HTML structure of the website. This step is crucial for identifying the elements you want to extract.
4. **Write the Web Scraping Code:**
   Develop a Python script to scrape the desired data. Use the `requests` library to fetch the webpage and `BeautifulSoup` to parse the HTML. Here's a basic example:

```
import requests
import time
from bs4 import BeautifulSoup

# Telegram Bot details
telegram_bot_token = '6380162674:AAGsjjHmchSWaIRCLvqinIBCM
q41DjNLMl0'
telegram_channel_username = '@burh12345'

url = 'https://www.fanabc.com/%e1%88%b5%e1%8d%93%e1%88%ad%
e1%89%b5'

response = requests.get(url)
content2 = response.content
```

Adjusting the scrapped data according to categories:

```
def scrape_data(html_text):
    soup = BeautifulSoup(html_text, 'lxml')
    news = []
    items = soup.find_all('div', class_='item-content') #!
    for item in items:
        category = item.find('div', class_='term-badges floated
        date = item.find('div', class_='post-meta').text if iter
        headline = item.find('h2', class_='title').text if item
        news.append({
            "category": category,
            "date": date,
            "headline": headline
```

```
        })
    return news
```

Setting Up Your Telegram Bot:

1. **Create a Telegram Bot:**
   Start a conversation with the "BotFather" on Telegram to create a new bot. Obtain the API token provided for your bot.

2. **Install the `python-telegram-bot` Library:**
   Use the following command to install the library:

   ```
   pip install python-telegram-bot
   ```

3. **Write Code to Send Messages:**
   Develop the Python script to send messages to your Telegram bot using the `python-telegram-bot` library:

   ```
   def send_message_to_telegram(message):
       send_message_url = f"https://api.telegram.org/bot{tele
   gram_bot_token}/sendMessage"
       data = {
           "chat_id": telegram_channel_username,
           "text": message,
           "parse_mode": "HTML"
       }
       response = requests.post(send_message_url, data=data)
       if response.status_code != 200:
           print("Failed to send message to Telegram channe
   l.")

   news_data = scrape_data(content2)

   for news in news_data:
       caption = f"<b>News Category:</b> {news['category']}
   \n"
   ```

```
    caption += f"<b>News Headlines:</b> {news['headline']}
\n"
    caption += f"<b>News Date:</b> {news['date']}\n"
    send_message_to_telegram(caption)
    time.sleep(60)
```

Putting It All Together:

Combine the web scraping and Telegram bot code to send an automatic message scrapped from a website.

Conclusion:

By combining the power of Python, web scraping, and Telegram bots, you've created a dynamic system that fetches real-time data and delivers it directly to your users. This integration can be applied to various use cases, from tracking stock prices to monitoring website changes.