Chimica nella Scuola 2 - 2024
ISSN: 0392-8942

Crea il tuo bot-chimico telegram

Giovanni Merola, Maria Cristina Mancini e Maria Laura Alessandroni
Istituto Tecnico Industriale Statale "Stanislao Cannizzaro", Colleferro (RM)
e-mail: giovanni.merola@itiscannizzarocolleferro.it

Abstract. In the digital age we live in, interactive learning has become a fundamental approach to engaging students effectively and stimulating their interest. Chemistry, with its complex formulas and molecular properties, can be a challenging subject for many students. However, recent advancements in artificial intelligence and the widespread availability of scientific data enable us to harness the potential of messaging bots to create innovative educational tools. A Telegram bot is a software application that operates within the Telegram messaging platform. Bots are designed to automate interactions and provide services to users through text-based chats. In the context of this project, the chemical bot on Telegram is developed with the aim of functioning as an interactive chemistry expert. Users can send questions and requests to the bot via chat, and the bot will respond by providing relevant chemical information. The educational significance of creating a Telegram bot extends beyond students specialized in computer science to include chemistry students. The process of developing a chemical bot involves acquiring cross-disciplinary skills that are valuable in various fields of study. By creating a bot, chemistry students can enhance their understanding of programming concepts and gain practical experience in integrating technology into their domain of study. This project provides a guide on creating a chemical bot using Python code executed on Google Colab. It explores how the bot utilizes the PubChemPy library to retrieve chemical information from the PubChem database. Users can interact with the bot, obtaining answers to basic questions such as retrieving the molecular mass of a compound based on its formula. Through the development of a chemical bot, students can develop critical thinking, problem-solving, and communication skills. Moreover, this innovative approach to studying chemistry fosters a deeper understanding of the subject matter and encourages active engagement with technology.

Keywords: intelligenza artificiale nella didattica; apprendimento interattivo; bot Telegram, competenze STEM

1. Introduzione

In questo articolo presentiamo un nuovo Bot Telegram progettato per supportare l'apprendimento della chimica in modo interattivo. Il nostro bot sfrutta la libreria PubChemPy [1-3] per accedere a un vasto database di composti chimici e fornire informazioni fondamentali come la massa molare e il nome IUPAC di un composto, sia in base alla formula inserita dall'utente sia in base al nome del composto stesso. Il bot offre agli studenti una piattaforma intuitiva per esplorare le proprietà chimiche dei composti, facilitando la comprensione delle relazioni tra struttura molecolare e comportamento chimico. Attraverso una semplice interfaccia di chat, gli studenti possono interagire con il bot, formulando domande e ottenendo risposte immediate e accurate. L'obiettivo di questo articolo è quello di fornire agli insegnanti e agli studenti uno strumento pratico per arricchire l'esperienza di apprendimento della chimica.

Un bot Telegram è un'applicazione software che interagisce con gli utenti all'interno della piattaforma di messaggistica Telegram. Questo articolo esplora come la costruzione di un bot su Telegram possa offrire agli studenti delle scuole superiori l'opportunità di sviluppare competenze STEM (Science, Technology, Engineering, and Mathematics) in modo interdisciplinare e approfondire la comprensione dei concetti di programmazione e di chimica. Negli ultimi anni, l'utilizzo del coding Python e di strumenti come Telegram ha acquisito sempre maggiore rilevanza nel campo dell'istruzione. L'uso di Python come strumento didattico e la sua crescente importanza nel campo della formazione sono stati oggetto di studio e ricerca da parte di esperti del settore. Lana e Mazzoli [4] hanno affrontato l'argomento nel loro articolo *Il coding e le sue potenzialità didattiche*, evidenziando come il coding, con l'utilizzo di Python, possa promuovere lo sviluppo di competenze didattiche e il pensiero computazionale nei giovani studenti. Altri autori [5, 6] hanno analizzato l'uso di Python nella ricerca scientifica, sottolineandone l'efficacia e la flessibilità nell'elaborazione e nell'analisi dei dati scientifici.

I risultati di questi studi dimostrano come Python abbia assunto un ruolo sempre più rilevante nell'istruzione chimica, permettendo agli studenti di sviluppare competenze STEM fondamentali come il pensiero critico, la risoluzione dei problemi e l'analisi dei dati. Inoltre, l'utilizzo di piattaforme di messaggistica come Telegram, in accordo con quanto evidenziato da Ismawati et al. [7, 8], si è dimostrato efficace nel fornire un ambiente didattico interattivo e accessibile per gli studenti. Attraverso la creazione di un bot chimico interattivo, gli studenti avranno l'opportunità di esplorare il mondo della chimica in modo coinvolgente e innovativo, sviluppando al contempo competenze chiave in ambito STEM. Oltre all'apprendimento dei concetti di programmazione e all'uso della libreria PubChemPy, gli studenti acquisiranno abilità trasversali, come il pensiero critico, la risoluzione dei problemi e la comunicazione. Il

nostro lavoro si inserisce in questo contesto dinamico, mirando a esplorare il potenziale positivo dell'integrazione dell'Intelligenza Artificiale (IA) nella didattica. Tuttavia, riconosciamo la necessità di un approccio cauto e di una costante revisione di queste affermazioni. Sebbene diversi studi [9-11] abbiano evidenziato i benefici dell'IA nell'ambito educativo, dobbiamo ammettere che questa è un'area in continua evoluzione. Le tecnologie emergenti portano con sé sfide e opportunità e il loro impatto sulla didattica non può essere considerato conclusivo. Con l'avanzamento della tecnologia, l'IA ha raggiunto una fase di sviluppo senza precedenti, influenzando profondamente vari aspetti della vita. L'educazione, come evidenziato, è diventata un campo cruciale per l'implementazione dell'IA, con impatti significativi sui metodi di insegnamento, l'ambiente di apprendimento, il management scolastico e la valutazione dell'insegnamento. È altrettanto fondamentale riconoscere che, con l'ampia adozione di strumenti digitali, realtà aumentata e canali di comunicazione come Telegram nell'ambito educativo, emergono questioni cruciali. La nostra indagine mira a stimolare una riflessione critica sul modo in cui queste tecnologie possono migliorare l'apprendimento, ma allo stesso tempo, riconosciamo la necessità di sottolineare con forza l'importanza di un'educazione informatica e della consapevolezza dei rischi associati a un uso improprio di tali strumenti come indicato da alcuni autori [12]; questa trasformazione, infatti, non è priva di sfide, con impatti etici, legali e sulla sicurezza dei dati che richiedono un'attenzione particolare. In tale contesto, esploriamo le dinamiche di creare un "bot chimico" su Telegram, riconoscendo che il docente, oltre alla sua utilità educativa, debba comprendere appieno l'impatto degli strumenti digitali sulla formazione degli studenti [13].

L'articolo illustra i passaggi necessari per la creazione di un bot chimico su Telegram utilizzando il codice Python eseguito su Google Colab [14]. Saranno presentate le funzionalità del bot, come il recupero delle informazioni chimiche dal database PubChem, e verranno forniti esempi di interazioni tra gli utenti e il bot. L'integrazione della tecnologia nel campo di studio della chimica contribuirà a un apprendimento più significativo e coinvolgente, fornendo agli studenti strumenti pratici per esplorare e applicare concetti scientifici in un contesto reale. Verrà fornita una guida pratica per aiutare gli studenti delle scuole superiori a sviluppare il proprio bot chimico interattivo, incoraggiando l'apprendimento attivo e lo sviluppo delle competenze necessarie per le future carriere in campo scientifico e tecnologico. In questo modo si offre una panoramica degli strumenti e delle metodologie disponibili per sfruttare al meglio il potenziale del coding e in particolare il linguaggio Python nell'ambito dell'istruzione chimica nelle scuole superiori. Il percorso educativo presentato in questo lavoro potrebbe essere più adatto negli istituti tecnici, oppure nei nuovi percorsi come liceo "*Made in italy*", dove il tempo dedicato all'insegnamento della chimica è più ampio, favorendo progetti STEM multidisciplinari con in-

segnanti di informatica. Al contrario, nelle scuole con un numero limitato di ore settimanali per l'insegnamento della chimica, l'implementazione di un approccio di questo tipo potrebbe risultare più problematica. In particolare, negli istituti tecnici si riconoscerà il ruolo chiave degli insegnanti e degli studenti dell'indirizzo informatico o elettronico come risorsa preziosa per lo sviluppo del codice e l'utilizzo degli strumenti necessari per il progetto. Questa collaborazione sinergica può non solo facilitare il processo di apprendimento, ma anche potenziare la creatività degli studenti nell'implementazione pratica dei concetti di chimica attraverso la programmazione. La possibilità che gli studenti di informatica diventino i principali artefici del bot, integrando le loro competenze nell'ottica STEM, rappresenta un'opportunità unica per l'apprendimento interdisciplinare della chimica.

Il Bot Telegram

Un bot Telegram può essere progettato per eseguire una vasta gamma di funzioni, come rispondere a domande degli utenti, fornire informazioni, eseguire calcoli e persino interagire con fonti esterne di dati come Wikipedia o PubChem. Ad esempio, un bot Telegram potrebbe essere programmato per rispondere alle domande degli studenti relative alla chimica. Utilizzando il linguaggio di programmazione Python, il bot può essere istruito per interpretare e comprendere il testo delle domande inviate dagli utenti. Successivamente, il bot può utilizzare il codice Python per eseguire una serie di operazioni, come cercare informazioni su una reazione chimica specifica, o calcolare la massa molare di un composto. Per creare un bot su Telegram è necessario registrarlo utilizzando il BotFather [15], uno strumento fornito da Telegram.

BotFather

BotFather è uno strumento fornito da Telegram che permette agli utenti di creare e configurare i propri bot Telegram. È fondamentalmente un bot Telegram, sviluppato da Telegram stesso, che funziona come un assistente per la creazione e la gestione dei bot. Con BotFather, gli utenti possono registrare i propri bot e ottenere un token unico per identificarli. Il token generato da BotFather è una sorta di "chiave" che identifica il tuo bot Telegram. È come una password segreta che permette al tuo bot di accedere all'API di Telegram e comunicare con gli utenti. Il token è necessario per configurare il tuo bot e consentire al codice che hai scritto di interagire con Telegram. Inoltre, BotFather offre diverse funzionalità di configurazione per il bot, come la possibilità di impostare un'immagine del profilo, una descrizione, comandi personalizzati e risposte predefinite. Gli utenti possono anche impostare le condizioni di privacy del bot, come limitare l'accesso solo agli utenti specifici o consentire l'accesso a tutti. In appendice è presente una mini-guida sull'utilizzo di BotFather e su come ottenere il token.

Python

Python è un linguaggio di programmazione creato nel 1991 da Guido van Rossum [16, 17]. È ampiamente utilizzato nel campo scientifico grazie alle sue librerie specializzate. È anche popolare nell'ambito didattico grazie alla sua sintassi intuitiva e alla facilità di apprendimento. Il codice Python è un insieme di istruzioni scritte nel linguaggio di programmazione Python. Python è noto per la sua sintassi semplice e leggibile, che lo rende adatto sia per i principianti che per i programmatori esperti. Con Python, è possibile creare programmi per una vasta gamma di applicazioni, come sviluppo web, analisi dei dati, automazione dei processi e molto altro ancora. Il codice Python può essere organizzato in moduli, che sono file separati contenenti funzioni, classi o variabili che possono essere riutilizzati all'interno di altri programmi. È anche possibile creare pacchetti, che sono insiemi di moduli correlati. Python supporta una vasta gamma di librerie e framework che consentono di ampliare le funzionalità di base del linguaggio.

Un semplice esempio di codice Python è quello di seguito riportato.

```
# Calcola e stampa la somma di due numeri
def somma(a, b):
return a + b

num1 = 5
num2 = 3

risultato = somma(num1, num2)
print("La somma di", num1, "e", num2, "è", risultato)
```

Questo codice definisce una funzione somma che accetta due argomenti e restituisce la somma dei due numeri. Vengono quindi definiti due numeri (num1 e num2) e viene chiamata la funzione somma per calcolare il risultato. Infine, il risultato viene stampato a schermo. Questo è solo un esempio molto semplice, ma Python può essere utilizzato per creare programmi complessi e interattivi.

Google Colab

Google Colaboratory, o più semplicemente Colab (**https://colab.google/**), è un ambiente di sviluppo integrato basato sul cloud che consente di scrivere, eseguire e condividere codice Python. È particolarmente utile per gli studenti di una scuola perché offre diversi vantaggi. In primo luogo, Colab fornisce un'interfaccia intuitiva e facile da usare, che rende l'apprendimento della programmazione accessibile anche ai principianti. Gli studenti possono scrivere il codice Python direttamente nel browser senza dover installare nulla sul proprio computer. Inoltre, Colab offre una potente capacità di esecuzione in tempo reale, consentendo agli studenti di testare immediatamente il loro codice e visualizzare i risultati. Colab supporta anche la creazione di documenti interattivi chiamati "notebook", che permettono agli studenti di integrare

il codice con testo, immagini, grafici e altri elementi per creare un'esperienza di apprendimento più coinvolgente. Infine, Colab offre la possibilità di eseguire codice su risorse di calcolo potenti e permette di condividere facilmente i notebook con gli insegnanti o i compagni di classe, favorendo la collaborazione e la revisione del codice. Complessivamente, Google Colab è uno strumento completo e accessibile per gli studenti di una scuola per imparare e utilizzare il codice Python in modo efficace [18].

Per accedere a Colab e creare un nuovo notebook occorre accedere con il browser all'indirizzo URL **https://colab.new/** e assicurarsi di essere connessi al proprio account Google; in mancanza di un account personale, è necessario crearne uno.

Nella pagina principale di Google Colab, si può creare un nuovo notebook o aprirne uno già esistente.

Creazione di un nuovo notebook

a) File → Nuovo Notebook:
   - una volta su Google Colab, andare su "File" e selezionare "Nuovo notebook".

b) Assegnazione di un Nome:
   - il nuovo notebook verrà creato con un nome generico come "Untitled0.ipynb"; si può fare clic sul nome e assegnarne uno personalizzato.

c) Aggiunta di celle:
   - un notebook è diviso in celle; si possono aggiungere nuove celle facendo clic su **+** CODE o + TEXT nella parte superiore o inferiore di una cella esistente.

d) Tipo di Cella:
   - ogni cella può essere di tipo "CODE" per il codice Python, o "TEXT" per il testo (Markdown).

e) Esecuzione di Codice:
   - per eseguire il codice in una cella, si può premere Shift + Enter o fare clic sul pulsante di riproduzione (PLAY) sulla sinistra della cella.

f) Salvataggio del Notebook:
   - si può salvare il proprio notebook su Google Drive o scaricarlo localmente; andare su "File" → "Salva una copia su Drive" per salvarlo su Google Drive, o "File" → "Scarica" per scaricarlo sul proprio dispositivo.

g) Condivisione del Notebook:
   - si può condividere il proprio notebook con altri utenti facendo clic su "Condividi" in alto a destra, concedendo loro l'accesso in sola lettura o in modalità di modifica.

h) Ambiente di Esecuzione:

- Google Colab fornisce un ambiente di esecuzione gratuito con accelerazione hardware GPU e TPU; si può selezionare l'ambiente di esecuzione desiderato andando su "Ambiente di esecuzione" → "Cambia tipo di ambiente di esecuzione".

Per maggiori informazioni si può consultare la guida in linea di Colab [19].

2. Telegram-bot

Uno dei modi più semplici per iniziare a creare un bot Telegram con Python è utilizzare la libreria **python-telegram-bot**. Con poche righe di codice, è possibile creare un bot che risponde ad alcune interazioni di base.

Ad esempio, un semplice codice Python per un bot che risponde a un saluto dell'utente è mostrato di seguito.

Questo codice crea un gestore di messaggi che controlla ogni messaggio inviato al bot. Se il messaggio è "ciao" o "salve", il bot risponderà con "Ciao! Come posso aiutarti?" In caso contrario, risponderà con "Mi dispiace, non ho capito."

Per eseguire il nostro codice abbiamo bisogno di installare le librerie con il comando:

```
!pip install python-telegram-bot==13.13
```

Tale comando installerà nell'archivio locale di Google Colab le librerie necessarie, ovvero una serie di script di codice già scritti che implementa delle funzioni standard che possono essere richiamate da un altro codice.

```
from telegram.ext import Updater, MessageHandler, Filters
def handle_message(update, context):
    text = update.message.text.lower()
    if text == 'ciao' or text == 'salve':
        response = 'Ciao! Come posso aiutarti?'
    else:
        response = 'Mi dispiace, non ho capito.'
    update.message.reply_text(response)
updater = Updater('TOKEN')  # Inserisci il token del tuo bot
updater.dispatcher.add_handler(MessageHandler(Filters.text, handle_message))
updater.start_polling()
updater.idle()
```

*il codice è commentato in appendice

Oltre alle interazioni di base, i bot Telegram possono anche utilizzare API diverse da Wikipedia o PubChem per ottenere informazioni aggiuntive. Ad esem-

pio, un bot potrebbe utilizzare l'API di OpenWeatherMap per dare informazioni meteo, l'API di Google Translate per traduzioni, o l'API di YouTube per cercare e riprodurre video. Si può utilizzare l'AI di ChatGPT per creare un codice adatto, considerando però che attualmente l'AI di ChatGPT si riferisce a dati del 2021 e che, quindi, i comandi e le librerie nel frattempo potrebbero essere cambiate.

3. Il bot chimico

Il bot chimico (Figura 1) sfrutta la libreria PubChemPy per accedere a un vasto database di composti chimici e fornire informazioni fondamentali come la massa molare e il nome IUPAC di un composto, sia in base alla formula inserita dall'utente sia in base al nome del composto stesso.

Il bot offre agli studenti una piattaforma intuitiva per esplorare le proprietà chimiche dei composti, facilitando la comprensione delle relazioni tra struttura molecolare e comportamento chimico. Attraverso una semplice interfaccia di chat, gli studenti possono interagire con il bot, formulando domande e ottenendo risposte immediate e accurate.

Il nostro bot Telegram offre un approccio interattivo e coinvolgente, permettendo agli studenti di esplorare in modo autonomo i concetti chiave della chimica e di ottenere informazioni accurate e tempestive. Siamo convinti che questa innovativa combinazione tra tecnologia e didattica possa contribuire a rendere l'apprendimento della chimica un'esperienza stimolante e accessibile a tutti.

Figura 1. Pagina iniziale del bot chimico

Il bot è in grado di restituire il nome IUPAC, il nome comune e la massa molare di un composto, sia che venga fornita la formula che il nome del composto

(Figura 2). Si noti che il nome del composto deve essere fornito in inglese e possibilmente il nome IUPAC; in alternativa, se si usa un sinonimo e ci sono casi di ambiguità sulla possibile struttura e nome, potrebbe restituire un errore.

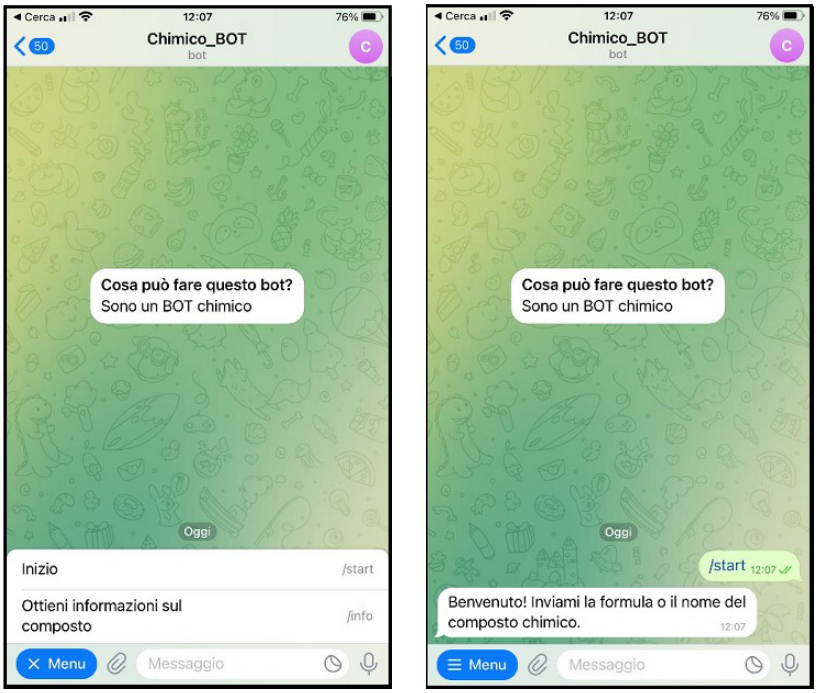

Figura 2. Esempio di utilizzo del bot chimico

Ecco come utilizzare il bot.

1. Inserire il token personale del bot Telegram nella variabile **TOKEN**.
2. Avviare il bot eseguendo il codice; il bot si metterà in ascolto per i comandi su Telegram.
3. Inviare il comando **/start** al bot su Telegram per iniziare la conversazione.
4. Dopo aver inviato il comando **/start**, il bot risponderà con un messaggio di benvenuto e chiederà di inviare la formula o il nome del composto chimico.
5. Si può inviare la formula o il nome del composto chimico al bot utilizzando il comando **/info**; ad esempio, inviare il comando **/info H2O** per ottenere informazioni sull'acqua.
6. Il bot eseguirà una ricerca nel database di PubChem, utilizzando la formula o il nome del composto fornito; se il composto viene trovato, il bot

restituirà il nome IUPAC, il nome comune (se disponibile) e la massa molare del composto.

7. Se il composto non viene trovato nel database, il bot invierà un messaggio di avviso.
8. Si può continuare a interagire con il bot inviando altri comandi o richieste di informazioni sui composti chimici.

Questo il codice da eseguire in Colab e da lasciare in esecuzione per avviare il bot chimico

L'installazione delle librerie aggiuntive è necessaria per la prima esecuzione del codice

```
!pip install python-telegram-bot==13.13
!pip install pubchempy
```

Una volta installate le librerie si può eseguire il codice

```
import telegram
from telegram.ext import Updater, CommandHandler
from pubchempy import get_compounds
# Inserisci il tuo token del bot Telegram qui
TOKEN = 'xxxxxxxxxxxxxxxx'
def start(update, context):
    context.bot.send_message(chat_id=update.effective_chat.id, text="Benvenuto! Inviami la formula o il nome del composto chimico.")
def compound_info(update, context):
    compound_input = ' '.join(context.args)
    try:
        compounds = get_compounds(compound_input, 'name' if compound_input.isalpha() else 'formula')
        if compounds:
        compound=compounds[0]
        compound_name_iupac = compound.iupac_name if compound.iupac_name else compound.synonyms[0]
        # Se il nome comune è presente tra i sinonimi, lo utilizzi; altrimenti, utilizzi il codice CAS.
                compound_name_common = next((synonym for synonym in compound.synonyms if synonym != compound_name_iupac), "N/A")
                compound_mass = compound.molecular_weight
                response = f"Nome IUPAC: {compound_name_iupac}\n" \
                           f"Nome comune: {compound_name_common}\n" \
                           f"Massa molecolare: {compound_mass} g/mol"
                context.bot.send_message(chat_id=update.effective_chat.id, text=response)
```

```
        else:
        context.bot.send_message(chat_id=update.effective_chat.id,  text="Nessuna informazione trovata per il composto inserito.")
        except Exception as e:
        context.bot.send_message(chat_id=update.effective_chat.id, text="Si è verificato un errore. Riprova più tardi.")
def main():
        updater = Updater(token=TOKEN, use_context=True)
        dispatcher = updater.dispatcher
        start_handler = CommandHandler('start', start)
        compound_info_handler = CommandHandler('info', compound_info)
        dispatcher.add_handler(start_handler)
        dispatcher.add_handler(compound_info_handler)
        updater.start_polling()
        updater.idle()
if __name__ == '__main__':
        main()
```

*il codice è commentato in appendice

Si ricorda che PubChemPy restituisce una lista di sinonimi per il nome comune del composto, ad esempio: etanolo, alcool etilico, ecc. Se il nome comune non è presente tra i sinonimi verrà utilizzato il codice CAS (Figura 3).

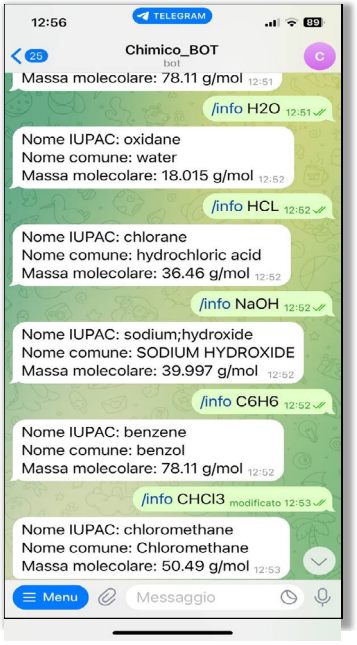

Figura 3. Utilizzo del bot

Il bot chimico rappresenta un'innovativa soluzione per l'apprendimento interattivo della chimica nell'era digitale. Questo permette agli utenti di avere una panoramica immediata delle caratteristiche principali di un composto. Tuttavia, le potenzialità del bot non si fermano qui. Esistono molte possibilità di espandere le sue funzionalità, come l'aggiunta di altre proprietà chimiche, la ricerca avanzata dei composti, il supporto per altre lingue e l'integrazione con altre fonti di dati scientifici. Ciò consentirebbe agli utenti di accedere a una gamma più ampia di informazioni e migliorare l'esperienza di apprendimento. Inoltre, il bot potrebbe essere arricchito con funzionalità interattive, come calcoli chimici e gestione degli errori più informativa. L'obiettivo finale è offrire agli studenti uno strumento didattico completo, intuitivo e coinvolgente per esplorare il mondo della chimica in modo efficace e stimolante.

3.1 Bot chimico avanzato

In questo esempio più complesso, modificato con l'aiuto dell'IA di ChatGPT, il bot è in grado di ricevere la formula, o il nome di un composto chimico da un utente e restituisce diverse informazioni su quel composto. Utilizzando la libreria PubChemPy, il bot recupera le proprietà del composto, come la formula molecolare, la massa molare, lo SMILES canonico, l'InChI, l'InChIKey, il nome IUPAC, il valore XLogP e la massa esatta. Inoltre, il bot restituisce fino a dieci nomi comuni associati al composto. I messaggi di risposta includono tutte queste informazioni, consentendo agli utenti di ottenere rapidamente dettagli sui composti chimici di interesse.

Librerie da installare (se non installate in precedenza):

```
!pip install python-telegram-bot==13.13
!pip install pubchempy
```

Codice del BOT:

```
import telegram
from telegram.ext import Updater, CommandHandler
from pubchempy import get_compounds

# Inserisci il tuo token del bot Telegram qui
TOKEN = "INSERISCI_IL_TUO_TOKEN"
def start(update, context):
    context.bot.send_message(chat_id=update.effective_chat.id,        text="Benvenuto! Inviami la formula o il nome del composto chimico per ottenere le informazioni.")

def compound_info(update, context):
    compound_input = ' '.join(context.args)

try:
    # Cerca il composto in base alla formula o al nome
        compounds = get_compounds(compound_input, 'name' if compound_input.isalpha() else 'formula')
```

```
if compounds:
        compound = compounds[0]

                # Recupera le proprietà del composto
                properties = {
                        "Molecular Formula": compound.molecular_formula,
                        "Molecular Weight": compound.molecular_weight,
                        "Canonical SMILES": compound.canonical_smiles,
                        "InChI": compound.inchi,
                        "InChIKey": compound.inchikey,
                        "IUPAC Name": compound.iupac_name,
                        "XLogP": compound.xlogp,
                        "Exact Mass": compound.exact_mass
                }
                # Recupera i nomi comuni del composto (massimo 10)
                common_names = compound.synonyms[:10]

                # Costruisci il messaggio di risposta con le proprietà e i nomi comuni del
                composto
                response = f"Informazioni sul composto:\n\n"
                for key, value in properties.items():
                        response += f"{key}: {value}\n"
                response += "\nNomi comuni:\n"
                if common_names:
                        for name in common_names:
                                response += f"- {name}\n"
                else:
                        response += "Nessun nome comune trovato."
                        context.bot.send_message(chat_id=update.effective_chat.id,
                        text=response)
                else:
                        context.bot.send_message(chat_id=update.effective_chat.id,
                        text="Nessuna informazione trovata per il composto inserito.")
        except Exception as e:
                        context.bot.send_message(chat_id=update.effective_chat.id,
                        text="Si è verificato un errore. Riprova più tardi.")
def main():
        updater = Updater(token=TOKEN, use_context=True)
        dispatcher = updater.dispatcher
        start_handler = CommandHandler('start', start)

        compound_info_handler = CommandHandler('info', compound_info)
```

```
        dispatcher.add_handler(start_handler)
        dispatcher.add_handler(compound_info_handler)

        updater.start_polling()
        updater.idle()

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

3.2 Bot chimico con foto

Se vogliamo aggiungere una foto della molecola del composto nel risultato del bot Telegram, è necessario utilizzare il concetto di "Inline Query" per recuperare un'immagine della molecola da una fonte esterna e inviarla come risposta (Figura 4).

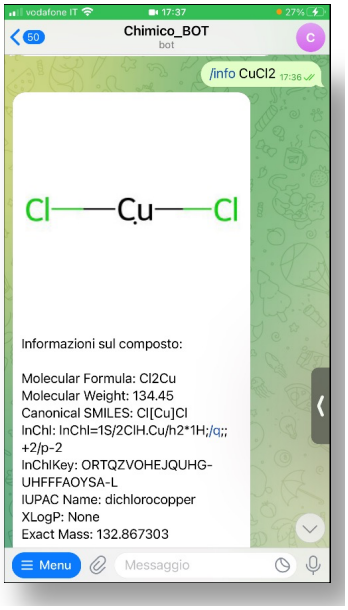

Figura 4. Risposta del bot con immagine della molecola

Tuttavia, il modulo pubchempy non fornisce un modo diretto per ottenere l'immagine di una molecola.

Per risolvere questo problema, possiamo utilizzare la libreria rdkit per generare un'immagine della molecola in formato PNG. Ecco una versione aggiornata del codice che include l'invio dell'immagine del composto:

```
!pip install rdkit pillow
!pip install python-telegram-bot==13.13
!pip install pubchempy
import telegram
from telegram import InlineQueryResultPhoto, InputTextMessageContent
from telegram.ext import Updater, CommandHandler, InlineQueryHandler
from pubchempy import get_compounds
from rdkit import Chem
from rdkit.Chem import Draw
# Inserisci il tuo token del bot Telegram qui
TOKEN = "INSERISCI_IL_TUO_TOKEN”

def start(update, context):
        context.bot.send_message(chat_id=update.effective_chat.id,    text="Benvenu-
to! Inviami la formula o il nome del composto chimico per ottenere le informazioni.")

def compound_info(update, context):
        compound_input = ' '.join(context.args)
        try:
        # Cerca il composto in base alla formula o al nome
                compounds = get_compounds(compound_input, 'name' if compound_
                input.isalpha() else 'formula')
                if compounds:
                        compound = compounds[0]
                        # Recupera le proprietà del composto
                        properties = {
                                "Molecular Formula": compound.molecular_formula,
                                "Molecular Weight": compound.molecular_weight,
                                "Canonical SMILES": compound.canonical_smiles,
                                "InChI": compound.inchi,
                                "InChIKey": compound.inchikey,
                                "IUPAC Name": compound.iupac_name,
                                "XLogP": compound.xlogp,
                                "Exact Mass": compound.exact_mass
                        }
                        # Recupera i nomi comuni del composto (massimo 10)
                        common_names = compounds.synonyms[:10]
                        # Genera un'immagine della molecola
                        mol = Chem.MolFromSmiles(compound.canonical_smiles)
                        img = Draw.MolToImage(mol)
                        img_path = "compound.png"
                        mg.save(img_path)
```

```
# Costruisci il messaggio di risposta con le proprietà e i nomi comuni del composto
                        response = f"Informazioni sul composto:\n\n"
                        for key, value in properties.items():
                                response += f"{key}: {value}\n"
                        response += "\nNomi comuni:\n"
                        if common_names:
                                for name in common_names:
                                        response += f"- {name}\n"
                        else:
                        response += "Nessun nome comune trovato."
                # Invia il messaggio di risposta con l'immagine della molecola
                context.bot.send_photo(chat_id=update.effective_chat.id, photo=o-
                pen(img_path, 'rb'), caption=response)
                else:
                        context.bot.send_message(chat_id=update.effective_chat.id,
                        text="Nessuna informazione trovata per il composto inseri-
                        to.")
        except Exception as e:
                context.bot.send_message(chat_id=update.effective_chat.id, text="Si è
                verificato un errore. Riprova più tardi.")
def main():
updater = Updater(token=TOKEN, use_context=True)
dispatcher = updater.dispatcher

start_handler = CommandHandler('start', start)
compound_info_handler = CommandHandler('info', compound_info)

dispatcher.add_handler(start_handler)
dispatcher.add_handler(compound_info_handler)

updater.start_polling()
updater.idle()

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

In questo codice, è stata aggiunta la generazione dell'immagine della molecola utilizzando il modulo rdkit e salvato l'immagine su disco come "compound.png". Successivamente, si è utilizzato il metodo context.bot.send_photo per inviare l'immagine insieme al messaggio di risposta.

Il codice risulta ancora incompleto per quanto riguarda la ricerca per il nome comune e, per non fare confusione fra formule e nome comune che potrebbero causare ambiguità nel bot, possiamo utilizzare comandi diversi:

```
import telegram

                    if len(compounds) == 1:
                        compound = compounds[0]

                # Recupera le proprietà del composto
                properties = {
                    "Molecular Formula": compound.molecular_formula,
                    "Molecular Weight": compound.molecular_weight,
                    "Canonical SMILES": compound.canonical_smiles,
                    "InChI": compound.inchi,
                    "InChIKey": compound.inchikey,
                    "IUPAC Name": compound.iupac_name,
                    "XLogP": compound.xlogp,
                    "Exact Mass": compound.exact_mass
                }

                # Recupera i nomi comuni del composto (massimo 10)
                common_names = compound.synonyms[:10]

                # Genera un'immagine della molecola
                mol = Chem.MolFromSmiles(compound.canonical_smiles)
                img = Draw.MolToImage(mol)
                img_path = "compound.png"  # Sostituisci con il percorso completo
                img.save(img_path)

                # Costruisci il messaggio di risposta con le proprietà e i nomi comuni del
                composto
                response = f"Informazioni sul composto:\n\n"
                for key, value in properties.items():
                    response += f"{key}: {value}\n"
                response += "\nNomi comuni:\n"
                if common_names:
                    for name in common_names:
                        response += f"- {name}\n"
                else:
                    response += "Nessun nome comune trovato."

                # Invia il messaggio di risposta con l'immagine della molecola

                context.bot.send_photo(chat_id=update.effective_chat.id,   photo=open(img_
                path, 'rb'), caption=response)

        else:
            # Gestione dell'ambiguità quando ci sono più composti corrispondenti al
```

```
nome comune
    # Verifica se il nome comune inserito corrisponde a un pattern con più parole
    words = re.findall(r'\w+', compound_input.lower())
    if len(words) > 1 and len(words) <= 5:
        compounds_filtered = [compound for compound in compounds if any(word in compound.synonyms_lower for word in words)]

    if compounds_filtered:
        if len(compounds_filtered) == 1:
            compound = compounds_filtered[0]

        # Restituisci le informazioni per il composto trovato
        properties = {
            "Molecular Formula": compound.molecular_formula,
            "Molecular Weight": compound.molecular_weight,
            "Canonical SMILES": compound.canonical_smiles,
            "InChI": compound.inchi,
            "InChIKey": compound.inchikey,
            "IUPAC Name": compound.iupac_name,
            "XLogP": compound.xlogp,
            "Exact Mass": compound.exact_mass
        }

        # Recupera i nomi comuni del composto (massimo 10)
        common_names = compound.synonyms[:10]

        # Genera un'immagine della molecola
        mol = Chem.MolFromSmiles(compound.canonical_smiles)
        img = Draw.MolToImage(mol)
        img_path = "compound.png"  # Sostituisci con il percorso completo
        img.save(img_path)

        # Costruisci il messaggio di risposta con le proprietà e i nomi comuni del composto
        response = f"Informazioni sul composto:\n\n"
        for key, value in properties.items():
            response += f"{key}: {value}\n"

        response += "\nNomi comuni:\n"
        if common_names:
            for name in common_names:
                response += f"- {name}\n"
        else:
```

```
                        response += "Nessun nome comune trovato."
                    # Invia il messaggio di risposta con l'immagine della molecola
                    context.bot.send_photo(chat_id=update.effective_chat.id, photo=open(img_path, 'rb'), caption=response)
                    else:
                        # Se ci sono ancora più composti corrispondenti al nome comune, chiedi di riformulare la domanda
                        context.bot.send_message(chat_id=update.effective_chat.id, text="Ci sono più composti corrispondenti al nome comune inserito. Riformula la domanda in modo più specifico.")
                    else:
                        context.bot.send_message(chat_id=update.effective_chat.id, text=f"Non ho trovato informazioni per il nome comune '{compound_input}'.")
                    else:
                        context.bot.send_message(chat_id=update.effective_chat.id, text="Il nome comune inserito deve essere formato da 2 a 5 parole.")
                    else:
                        context.bot.send_message(chat_id=update.effective_chat.id, text=f"Non ho trovato informazioni per il composto '{compound_input}'.")
        except ValueError as ve:
        print(f"Errore: {str(ve)}")
            context.bot.send_message(chat_id=update.effective_chat.id, text="Il nome comune inserito non è valido.")
        except Exception as e:
            print(f"Errore: {str(e)}")
            context.bot.send_message(chat_id=update.effective_chat.id, text="Si è verificato un errore. Riprova più tardi.")

def main():
        updater = Updater(token=TOKEN, use_context=True)
        dispatcher = updater.dispatcher

        start_handler = CommandHandler('start', start)
        cerca_nome_comune_handler = CommandHandler('info', compound_info)

        dispatcher.add_handler(start_handler)
        dispatcher.add_handler(cerca_nome_comune_handler)

        updater.start_polling()
        updater.idle()

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

Ora il bot risponderà ai comando **/info** per effettuare la ricerca delle sostanze chimiche rispettivamente tramite nome comune, formula e nome IUPAC (Figura 5).

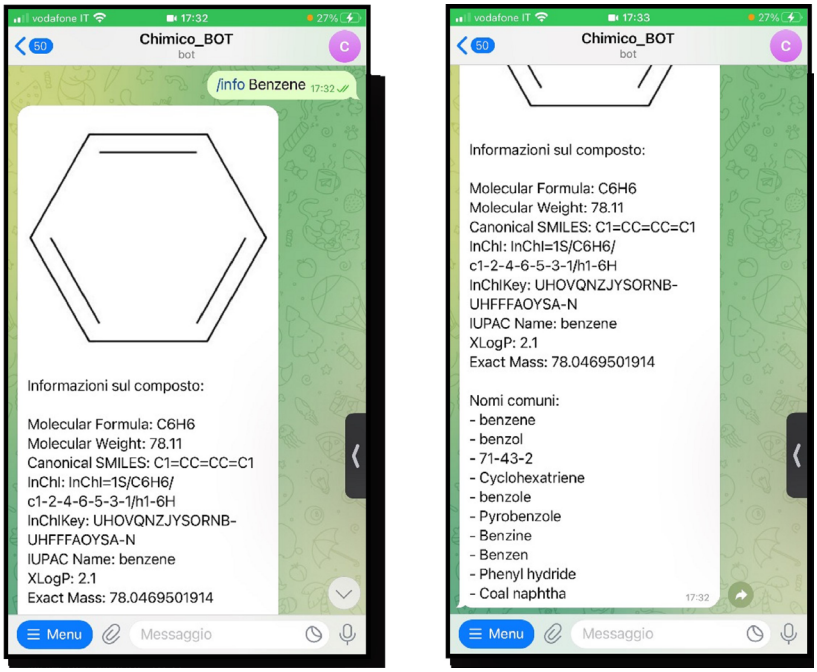

Figura 5. Risposta del bot

3.3 Risoluzione dei problemi

Se gli script non si avviano, assicurarsi che siano state installate le librerie necessarie:

```
pip install --upgrade python-telegram-bot==13.13
pip install --upgrade pubchempy
pip install rdkit  # richiesto dal bot con immagini
```

Assicurarsi che nello script sia inserito il token giusto al posto della frase “YOUR_BOT_TOKEN” o “TOKEN”

Esempio:

```
import telegram
from telegram.ext import Updater, CommandHandler
from pubchempy import get_compounds
# Inserisci il token personale del bot Telegram qui
TOKEN = '6263886992:AA5D3ffijCRO0YmIIjldUWLxjA29HQkj7oE'
```

```
def start(update, context):
context.bot.send_message(chat_id=update.effective_chat.id, text="Benvenuto! Inviami la formula o il nome del composto chimico.")
def compound_info(update, context):
compound_input = ' '.join(context.args)
...
```

3.4 Eccezioni non gestite (debug)

Un problema nell'esecuzione di get_compounds dalla libreria PubChemPy potrebbe generare un'eccezione non gestita. Verificare se l'eccezione specifica viene catturata correttamente e mostrata nella finestra di output dello script.

```
except Exception as e:
print(f"Errore: {str(e)}")
context.bot.send_message(chat_id=update.effective_chat.id, text="Si è verificato un errore. Riprova più tardi.")
```

Una volta identificato l'errore specifico, si potranno prendere le misure necessarie per risolverlo.

4. Conclusioni

L'esperienza con il bot chimico può essere affascinante e utile per gli utenti interessati alla chimica. Tuttavia, è importante notare che il bot è ancora in fase di sviluppo e potrebbe non funzionare correttamente in tutte le situazioni. Essendo stato realizzato da principianti, potrebbero esserci alcuni difetti o limitazioni nel codice.

È possibile incontrare alcune problematiche durante l'utilizzo del bot, come la mancanza di informazioni per alcuni composti, errori di ricerca o problemi nel riconoscimento dei nomi comuni. Questi problemi potrebbero derivare dalla complessità delle query chimiche, dalle possibili ambiguità nei nomi dei composti, o da errori nell'implementazione del codice stesso. Tuttavia, nonostante questi eventuali problemi, l'esperienza con il bot può comunque fornire informazioni utili sulla chimica e sulle proprietà dei composti. È importante considerare il bot come uno strumento di apprendimento in via di sviluppo e non come una fonte di informazioni completamente affidabile. Per migliorare l'efficienza e l'accuratezza del bot, potrebbe essere necessario effettuare ulteriori sviluppi, come l'implementazione di algoritmi più sofisticati per la ricerca e l'elaborazione dei dati chimici, l'ottimizzazione delle query e la gestione delle possibili ambiguità. È innegabile che il nostro approccio possa essere più agevolmente integrato negli istituti tecnici oppure nei nuovi percorsi come liceo "*Made in italy*", dove le ore dedicate alla chimica sono più numerose e progetti multidisciplinari, soprattutto nell'ottica STEM, sono più facilmente attuabili.

L'esperienza con il bot chimico può avere diversi risvolti didattici e coinvolgere i ragazzi in modo interattivo nel campo della chimica. Essendo un bot accessibile tramite Telegram, offre un'opportunità di apprendimento divertente e stimolante per i giovani interessati alla materia. Attraverso l'utilizzo del bot, i ragazzi possono sperimentare l'interazione diretta con un programma informatico che fornisce informazioni sulla massa molare, le proprietà e altre caratteristiche dei composti chimici. Siamo convinti che questa innovativa combinazione tra tecnologia e didattica possa contribuire a rendere l'apprendimento della chimica un'esperienza stimolante e accessibile a tutti. Questo coinvolgimento attivo può aiutare a rafforzare la comprensione dei concetti chimici e consentire agli studenti di approfondire i loro studi in modo autonomo. Lo sviluppo del codice del bot chimico potrebbe presentare un'opportunità per coinvolgere i ragazzi nel processo di apprendimento della chimica attraverso l'interazione con un programma informatico. L'approccio graduale nel descrivere lo sviluppo del codice e l'accento sull'aspetto didattico possono rendere l'esperienza interessante e accessibile per tutti, fornendo al contempo una solida base di conoscenza sulla chimica e sullo sviluppo di applicazioni bot. È imperativo sottolineare che l'integrazione di strumenti digitali, realtà aumentata e risorse online, compresi canali come Telegram, richiedono una considerazione ponderata. Dobbiamo porre una forte enfasi sull'importanza di educare gli utenti su eventuali rischi e limiti associati a questi strumenti, cosa che include una consapevolezza critica su temi come la sicurezza online, la privacy e l'accuratezza delle informazioni.

Riferimenti

[1] B. Dahlgren, ChemPy: A package useful for chemistry written in Python, *Journal of Open Source Software,* 2018, **3**(24), 565.

[2] Chempy: **https://pythonhosted.org/chempy/**.

[3] PubchemPy: **https://pubchempy.readthedocs.io/en/latest/.**

[4] L. Lana, V. Mazzoli, Il coding e le sue potenzialità didattiche, *Educare. it,* 2021, 21(9), 98-105.

[5] J. E. Menke, C. S. Reese, T. R. Martinez, Hierarchical models for estimating individual ratings from group competitions, in *American Statistical Association*, 2007 (**https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=293122079779b53acca7b83d7ba3eb3a0cbd1032**).

[6] F. Nati, Python e il suo utilizzo nella ricerca scientifica, in *Linux Day*, Roma, 2002.

[7] M. I. Ardimansyah, M. H. Widianto, Development of online learning media based on Telegram Chatbot (Case studies: Programming courses), in *Journal of Physics: Conference Series*, 2021, **1987**, 012006 (**https://iopscience.iop.org/article/10.1088/1742-6596/1987/1/012006/pdf**).

[8] D. Ismawati, Studing while playing: a new method of e-learning for mille-

nials using social media, *International Journal of Learning, Management and Digitalization (IJLMD),* 2020, **1**(1), 8-13.

[9] M. L. Owac, A. Sawicka, P. Weichbroth, Artificial intelligence technologies in education: benefits, challenges and strategies of implementation, in *Artificial Intelligence for Knowledge Management. AI4KM 2019. IFIP Advances in Information and Communication Technology*, (M. L. Owoc, M. Pondel, Eds.), Springer, 2019, vol. 599 (**https://doi.org/10.1007/978-3-030-85001-2_4**).

[10] K. M. Al-Tkayneh, E. M. Alghazo, D. Tahat, The Advantages and Disadvantages of Using Artificial Intelligence in Education, *Journal of Educational and Social Research,* 2023, **13**(4), 105-117.

[11] A. Y. Kenchakkanavar, Exploring the Artificial Intelligence Tools: Realizing the Advantages in Education and Research, *Journal of Advances in Library and Information Science,* 2023, **12**(4), 218-224.

[12] F. J. Hinojo-Lucena, I. Aznar-Díaz, M. P. Cáceres-Reche, J. M. Trujillo-Torres, J. M. Romero-Rodríguez, Problematic Internet Use as a Predictor of Eating Disorders in Students: A Systematic Review and Meta-Analysis Study, *Nutrients,* 2019, **9**(11), 2151.

[13] Y. Liu, S. Saleh, J. Huang, Artificial Intelligence in Promoting Teaching and Learning Transformation in Schools, *International Journal of Innovation, Creativity and Change,* 2021, **3**(15), 892-900.

[14] E. Bisong, Google Colaboratory, in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners* by E. Bisong, Apress Berkeley, CA, 2019, 59-64.

[15] From BotFather to 'Hello World', Telegram.org, 2023: **https://core.telegram.org/bots/tutorial**.

[16] G. Van Rossum, Python Programming language, *USENIX annual technical conference*, 2007.

[17] G. Van Rossum, F. L. Drake, *Python reference manual*, Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[18] G. Merola, J. Campagna, S. Celletti, M. Maglia, Imparare la chimica analitica con il coding in Python, *CnS Chimica nella Scuola,* 2023, **4**, 49-81.

[19] Ti diamo il benvenuto in Colab: **https://colab.research.google.com/.**

APPENDICE

Guida dettagliata all'utilizzo di BotFather per creare e configurare un Bot Telegram

Passo 1: Trovare e avviare BotFather

- Aprire Telegram e cercare BotFather
    - Avviare l'app Telegram sul proprio dispositivo.
    - Nella barra di ricerca, digitare "BotFather" e avviare la chat con BotFather.

Passo 2: Avviare e creare un Nuovo Bot

- Inviare il comando /newbot a BotFather per iniziare a creare un nuovo bot.
- Seguire le istruzioni per assegnare un nome al tuo bot.
- Dopo aver scelto un nome, BotFather chiederà di assegnare un username al bot personale (deve terminare con "bot").

Passo 3: Ottenere il Token del Bot

- Dopo aver completato la creazione, BotFather fornirà un messaggio di conferma contenente il token del bot personale.
- Copiare il token del bot (ad es.: 1234567890:ABCdefghIJKlmnOpqrStuvWxyz12345678)

Passo 4: Impostare comandi personalizzati

- Usare il comando /setcommands per definire una lista di comandi personalizzati e relative descrizioni.
- Ad esempio: start - Avvia il bot
  info – Cerca le proprietà del composto

Passo 5: Configurare menu e bottoni (opzionale)

- Utilizzare comandi come /setinline, /setinlinefeedback, e /setcommands per creare tastiere personalizzate e pulsanti inline.

Passo 6: Testare il Bot personale

- Tornare alla chat con il bot personale su Telegram.
- Inviare il comando /start per avviare il bot e verificare che risponda correttamente.

Passo 7: Personalizzare il proprio Bot

- Utilizzare una serie di comandi come /setname, /setdescription, e altri per gestire le impostazioni del proprio bot.

Considerazioni Aggiuntive:

- Immagini Esplicative:
    - Inserire le immagini nei punti appropriati, ad esempio, uno screen-

shot della ricerca di BotFather e un altro dello screenshot che mostra il token del bot.

- Documentazione BotFather:
    - Per ulteriori dettagli e opzioni, consultare la documentazione ufficiale di BotFather su BotFather Documentation.

Seguendo questa guida dettagliata, lo studente sarà in grado di creare e configurare il proprio bot Telegram in modo efficace, personalizzando comandi, tastiere e impostazioni per fornire un'esperienza utente ottimale.

Commenti al codice Telegram Bot

- from telegram.ext import Updater, MessageHandler, Filters

Importa le classi ovvero i comandi necessari (**Updater, MessageHandler, Filters)** dalla libreria **telegram.ext** per creare e gestire il bot Telegram.

- def handle_message(update, context):

Definisce una funzione di gestione dei messaggi del bot e prende in input l'oggetto **update**, che contiene le informazioni sul messaggio ricevuto, e **context**, che fornisce ulteriori contestualizzazioni; all'interno di questa funzione vengono eseguiti i seguenti comandi.

- **text = update.message.text.lower()** - estrae il testo del messaggio ricevuto e lo converte in minuscolo per semplificare la gestione delle risposte.
- **if text == 'ciao' or text == 'salve':** - controlla se il testo del messaggio è "ciao" o "salve"; se la condizione è verificata esegue
    - **response = 'Ciao! Come posso aiutarti?'**: se il messaggio contiene "ciao" o "salve", viene assegnata una risposta di saluto.
- **else:** - se il messaggio non corrisponde a "ciao" o "salve", allora esegue
    - **response = 'Mi dispiace, non ho capito.'** e viene assegnata una risposta di default per gestire le domande non riconosciute.
- **update.message.reply_text(response)** - in entrambi i casi invia la risposta al mittente del messaggio originale.

- updater = Updater('TOKEN')

Crea un oggetto **Updater** utilizzando il token del bot Telegram personale. È necessario sostituire **'TOKEN'** con il token reale del bot personale fornito da BotFather.

- updater.dispatcher.add_handler(MessageHandler(Filters.text, handle_message))

Aggiunge un gestore dei messaggi al dispatcher dell'updater. Il gestore dei messaggi utilizza la funzione **handle_message** per gestire i messaggi di testo.

- updater.start_polling()

Avvia il polling per ricevere i messaggi dal bot Telegram.

- updater.idle()

Mantiene l'esecuzione del programma fino a quando non viene interrotto manualmente.

Commenti al codice del bot chimico

1. **import telegram**
   Importa il modulo **telegram (**che deve essere stato installato in precedenza con il comando "pip install…") che fornisce le funzionalità per interagire con l'API di Telegram.
2. **from telegram.ext import Updater, CommandHandler**
   Importa la classe **Updater** e la classe **CommandHandler** dal modulo **telegram.ext**. Queste classi sono necessarie per gestire gli aggiornamenti del bot e i comandi ricevuti.
3. **from pubchempy import get_compounds**
   Importa la funzione **get_compounds** dal modulo **pubchempy**. Questa funzione sarà utilizzata per ottenere informazioni sui composti chimici.
4. **TOKEN = '...'**
   Definisce la chiave di accesso del bot Telegram; è importante sostituire **'...'** con il token reale (quello personale).
5. **def start(update, context):**
   Definisce una funzione chiamata **start** che viene eseguita quando il comando **/start** viene inviato al bot. Questa funzione invia un messaggio di benvenuto al chat_id corrispondente.
6. **def compound_info(update, context):**
   Definisce una funzione chiamata **compound_info** che viene eseguita quando il comando **/info** viene inviato al bot. Questa funzione gestisce la ricerca e l'invio delle informazioni sul composto chimico richiesto.
7. **def main():**
   Definisce la funzione principale **main** che crea un'istanza di **Updater** e **Dispatcher** per gestire gli aggiornamenti del bot.
8. **if __name__ == '__main__':**
   Verifica se lo script viene eseguito direttamente e, in tal caso, chiama la funzione **main()** per avviare il bot.

Le linee di codice all'interno delle funzioni **start** e **compound_info** sono responsabili dell'invio delle risposte ai messaggi ricevuti. Utilizzano il modulo **telegram.ext** per ottenere l'oggetto **context** che fornisce il metodo **send_message** per inviare i messaggi di risposta.

La funzione **compound_info** viene eseguita quando viene inviato il comando **/info** al bot. Questa funzione gestisce la richiesta di informazioni su un composto chimico specifico, che può essere fornito tramite nome o formula.
La riga **compound_input = ' '.join(context.args)** recupera l'input dell'utente, che può essere una stringa contenente il nome o la formula del composto. Viene utilizzata la funzione **join** per unire tutti gli argomenti della lista **context.args** in una singola stringa.
Successivamente, viene utilizzata la funzione **get_compounds** del modulo **pubchempy** per ottenere informazioni sui composti chimici corrispondenti alla stringa **compound_input**. La funzione accetta due parametri: la stringa **compound_input** e il tipo di ricerca, che viene determinato in base al tipo di input fornito dall'utente (nome o formula).
Se vengono trovati composti corrispondenti, la funzione seleziona il primo composto dalla lista dei risultati e recupera il suo nome IUPAC utilizzando l'attributo **iupac_name**. Se l'attributo **iupac_name** è vuoto, viene utilizzato il primo sinonimo presente nell'attributo **synonyms**. Inoltre, viene verificata la presenza di un secondo sinonimo nell'attributo **synonyms** e, se presente, viene assegnato alla variabile **compound_name_common**. Altrimenti, viene assegnato il valore "N/A" per indicare l'assenza di un nome comune. Successivamente, viene estratta la massa molecolare del composto utilizzando l'attributo **molecular_weight**. Infine, viene costruita una risposta che contiene il nome IUPAC, il nome comune (se disponibile) e la massa molare del composto. Questa risposta viene inviata all'utente utilizzando il metodo **send_message** dell'oggetto **context.bot** per inviare il messaggio al **chat_id** corrispondente.
Se non vengono trovate informazioni per il composto inserito, viene inviato un messaggio indicando l'assenza di informazioni. In caso di errori durante l'esecuzione della funzione, viene inviato un messaggio di errore generico.