	ЗВО: Національний університет «Львівська політехніка» **Навчальний рік:** 2024/2025 **Семестр:** весняний	**Навчальна дисципліна:** Розробка кросплатформених додатків **Лабораторна робота № 9:** Розроблення чат-боту із інтелектуальним пошуком
	Кафедра систем автоматизованого проектування **Викладач:** Володимир Гавран	**Група:** ПП-24 **Студент:** Роман Звір **Варіант:** 9

Мета роботи

Розробити чат-бот із інтелектуальним пошуком на основі бібліотеки tkinter.
Додаток повинен містити:

- Текстовий віджет для виведення діалогу з кольоровим форматуванням відповідей.
- Текстове поле для введення повідомлень користувача.
- Пошукове поле з прапорцем чутливості до регістру.
- Кнопки «Send» і «Find All» для надсилання повідомлень і пошуку відповідно.
- Прив’язку подій клавіатури, зокрема:
  - <KeyPress-Return> — для швидкого надсилання повідомлення.
  - <KeyPress> — для очищення попередніх результатів пошуку.
- Обробку тегів Text.tag_config() з пріоритетом tag_raise, що забезпечує виділення тексту поверх інших кольорових міток.

Інструкція до виконання роботи

1. Вибрати тему діалогу для чат-бота згідно із вашим номером варіанту:

1	2	3	4	5	6	7	8
Професійний розвиток і кар'єра	Навчання і саморозвиток	Баланс між роботою та відпочинком	Емоційне здоров’я і добробут	Джерела натхнення	Мрії та життєві цілі	Технології майбутнього	Звички та продуктив-ність
9	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Подорожі та нові враження	Хобі та особисті інтереси	Цінності та переконання	Улюблені книги та автори	Вплив музики на настрій	Вивчення іноземних мов	Спілкування та міжособис-тісні стосунки	Соціальні мережі та цифрова гігієна
17	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Волонтерство та соціальна активність	Екологічна свідомість	Кулінарні уподобання	Спорт і фізична активність	Подолання стресу та емоційне здоров’я	Планування часу та організованість	Роль технологій у повсякденному житті	Подорожі та культурний обмін
25	**26**	**27**	**28**	**29**	**30**	**31**	**32**
Професійні амбіції	Особисті досягнення	Натхнення та джерела мотивації	Етика у цифрову епоху	Майбутнє освіти	Вплив соціальних мереж на спілкування	Улюблені традиції або свята	Враження від останньої культурної події

2. Створити головне вікно програми.
3. Організувати інтерфейс виведення повідомлень:
3.1. Створити Text-віджет у фреймі frame_text для відображення історії спілкування.
3.2. Додати вертикальну смугу прокрутки (Scrollbar).
3.3. Забезпечити кольорове форматування повідомлень за допомогою тегів (повідомлення користувача, відповіді бота, підсвітка знайдених збігів).
4. Реалізувати поле для введення повідомлень користувачем:
4.1. Додати Entry для введення тексту користувача.
4.2. Додати кнопку Send, яка надсилає повідомлення.
5. Зберігати попередньо підготовлений список запитань/відповідей бота.
5.1. Перевіряти індекс перед виведенням нового повідомлення (обмежити до наявного списку).
5.2. При досягненні кінця списку – повідомити користувача про завершення діалогу.

6. Створити пошуковий блок у чаті.
6.1. Додати поле Entry для введення тексту пошуку.
6.2. Додати Checkbutton для врахування регістру (Match case).
6.3. Додати кнопку Find All для пошуку всіх входжень і підсвітки.
6.4. Під час натискання кнопки виконувати пошук та підсвічувати всі входження у полі Text.
7. Налаштувати обробку подій:
7.1. Подія поля пошуку <KeyPress-Return> для відправлення повідомлення клавішею Enter.
7.2. Подія головного вікна <KeyPress> у вікні для очищення попередньої підсвітки.
8. Запустити головний цикл програми.

1. Теоретичні відомості

1. Tkinter

Tkinter — це стандартна бібліотека Python для створення графічного інтерфейсу користувача (GUI). Вона дозволяє створювати вікна, кнопки, текстові поля, меню тощо.

2. Основні компоненти інтерфейсу:

- **Text** — багаторядковий віджет для виведення діалогу. Дозволяє форматувати текст за допомогою тегів (наприклад, кольорове виділення відповідей).
- **Entry** — однорядкове текстове поле для введення повідомлень користувача.
- **Checkbutton** — елемент керування, який дозволяє вибирати опцію (наприклад, чутливість до регістру).
- **Button** — кнопки для виконання дій (наприклад, надсилання повідомлення або пошук).

3. Керування подіями клавіатури:

- **<KeyPress-Return>** — обробляється для швидкої відправки повідомлення (аналог натискання кнопки "Send").
- **<KeyPress>** — використовується для очищення попередніх результатів пошуку під час введення нового тексту.

4. Інтелектуальний пошук:

- Відбувається пошук тексту в діалозі на основі введеного запиту.
- Можна враховувати або ігнорувати регістр за допомогою прапорця.
- Всі знайдені збіги виділяються у вікні Text.

5. Робота з тегами у Text:

- **.tag_config()** — дозволяє створити стиль для певного фрагменту тексту (наприклад, фон або колір шрифту).
- **.tag_add()** — застосовує тег до певного діапазону тексту.
- **.tag_raise()** — піднімає пріоритет відображення тегу, щоб виділення не перекривалося іншими тегами.

2. Лістинг коду програми

```
import tkinter as tk
from tkinter import font as tkfont
import datetime # Add this import at the top
```

```
root = tk.Tk()
root.title("Chat Bot")
root.geometry("400x500") # Set initial size
root.minsize(300, 300)
root.configure(bg="#f0f0f0") # Light background for the app

# Create custom fonts
message_font = tkfont.Font(family="Helvetica", size=10)
header_font = tkfont.Font(family="Helvetica", size=10, weight="bold")

case_var = tk.BooleanVar(value=False)

# Frame for title
frame_title = tk.Frame(root, bg="#4a7abc", pady=10)
frame_title.pack(fill="x")
tk.Label(frame_title, text="Chat Bot", font=("Helvetica", 14, "bold"),
fg="white", bg="#4a7abc").pack()

frame_text = tk.Frame(bg="#f0f0f0")
frame_text.pack(fill="both", expand=True, padx=10, pady=10)

scroll = tk.Scrollbar(frame_text)
scroll.pack(side="right", fill="y")

text_widget = tk.Text(frame_text, width=35, wrap="word", yscrollcommand=scroll.set,
state="disabled", font=message_font, bg="white", bd=1, relief="solid")
text_widget.pack(side="left", fill="both", expand=True)
scroll.config(command=text_widget.yview)

text_widget.tag_config("highlight", foreground="yellow", background="#333333")
text_widget.tag_config("green", background="#e6ffe6")
text_widget.tag_config("blue", background="#e6f2ff")
text_widget.tag_config("user", font=header_font, foreground="#006600")
text_widget.tag_config("bot", font=header_font, foreground="#003366")

frame_send = tk.Frame(root, bg="#f0f0f0")
frame_send.pack(fill="x", padx=10, pady=(0, 10))

id_message = 0
messages = [
"Яке місце ти б хотів(-ла) відвідати?",
"Які враження залишилися від останньої поїздки?",
"Чи любиш більше активний відпочинок чи спокійне проведення часу?",
"Яка поїздка справила на тебе найбільше враження?",
"Чи були подорожі, які змінили твоє бачення світу або себе?",
"Які країни або міста є в твоєму списку бажаних подорожей?",
"Чи плануєш якісь поїздки найближчим часом?",
"Які незвичні страви тобі доводилося куштувати під час мандрів?",
"Що для тебе найважливіше в подорожах — культура, природа, люди чи відпочинок?",
"Які корисні звички або інсайти ти привіз(-ла) з подорожей?",
]
```

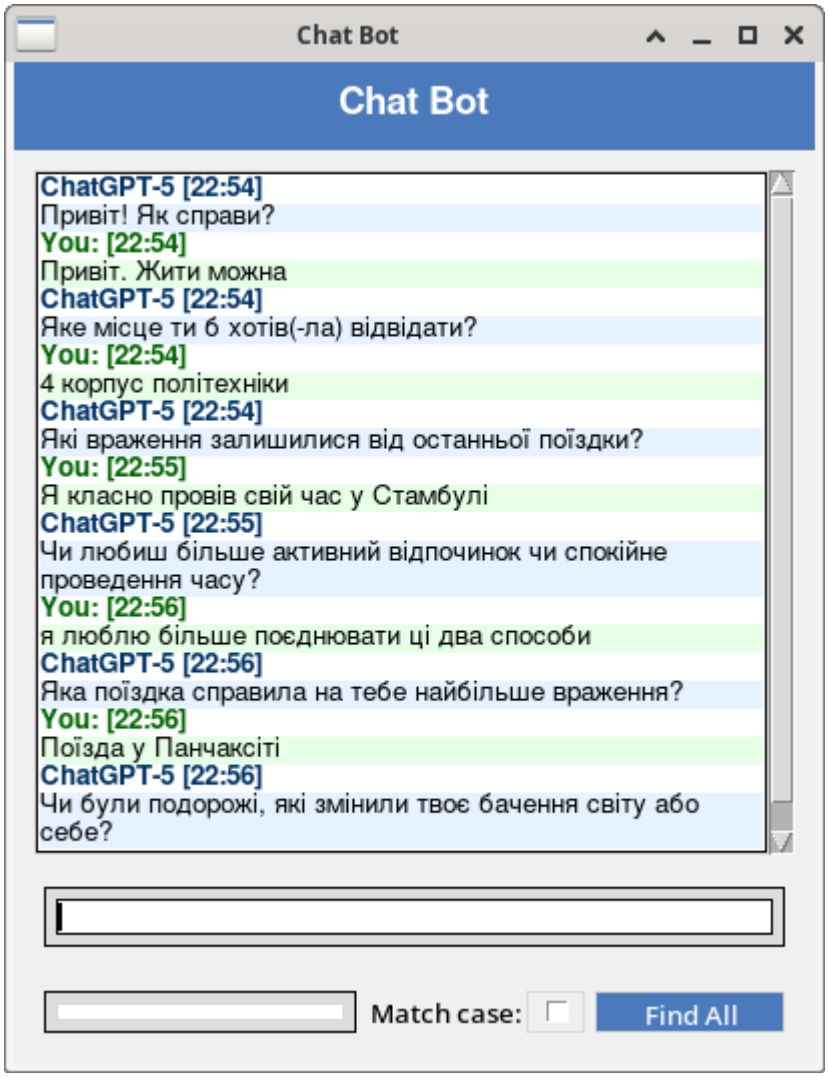
```
def create_line(tag, text):
text_widget.config(state="normal")
# Add timestamp for headers
if text == "You:" or text == "ChatGPT-5":
current_time = datetime.datetime.now().strftime("%H:%M")
display_text = f"{text} [{current_time}]"
else:
display_text = text
if tag == "green":
if display_text.startswith("You:"):
text_widget.insert(tk.END, f"{display_text}\n", "user")
else:
text_widget.insert(tk.END, f"{display_text}\n", tag)
elif tag == "blue":
if display_text.startswith("ChatGPT-5"):
text_widget.insert(tk.END, f"{display_text}\n", "bot")
else:
text_widget.insert(tk.END, f"{display_text}\n", tag)
else:
text_widget.insert(tk.END, f"{display_text}\n", tag)
text_widget.yview_moveto(1.0)
text_widget.config(state="disabled")
def handle_send():
global id_message
post = entry_send.get()
if not post.strip():
return
create_line("green", "You:")
create_line("green", post)
entry_send.delete(0, tk.END)
create_line("blue", "ChatGPT-5")
if id_message >= len(messages):
create_line("blue", "Я втомився, поговоримо пізніше...")
return
create_line("blue", messages[id_message])
id_message += 1

# Entry with rounded corners feel
entry_frame = tk.Frame(frame_send, bg="#dddddd", bd=1, relief="solid")
entry_frame.pack(side="left", fill="x", expand=True, padx=5, pady=5)

entry_send = tk.Entry(entry_frame, width=62, bd=0, font=message_font)
entry_send.pack(side="left", fill="x", expand=True, padx=5, pady=5)

button_send = tk.Button(
frame_send, width=10, text="Send", bg="#4a7abc", fg="white",
command=handle_send, relief="flat")
button_send.pack(side="left", padx=5, pady=5)
```

```
button_clear = tk.Button(
frame_send, width=10, text="Clear Chat", bg="#f44336", fg="white",
command=lambda: clear_chat(), relief="flat")
button_clear.pack(side="left", padx=5, pady=5)

frame_find = tk.Frame(root, bg="#f0f0f0")
frame_find.pack(fill="x", padx=10, pady=(0, 10))

# Search section with better styling
search_frame = tk.Frame(frame_find, bg="#dddddd", bd=1, relief="solid")
search_frame.pack(side="left", fill="x", expand=True, padx=5, pady=5)

entry_search = tk.Entry(search_frame, bd=0, font=message_font)
entry_search.pack(side="left", fill="x", expand=True, padx=5, pady=5)

label_case = tk.Label(frame_find, text="Match case:", bg="#f0f0f0")
label_case.pack(side="left", pady=5)

check_case = tk.Checkbutton(frame_find, variable=case_var, bg="#f0f0f0")
check_case.pack(side="left", pady=5)

def handle_find_all():
text_widget.tag_raise("highlight")
text_widget.config(state="normal")
text_widget.tag_remove("highlight", "1.0", "end")
pattern = entry_search.get()
if not pattern:
return
index = "1.0"
stopindex = "end"
while True:
position = text_widget.search(
pattern, index=index, stopindex=stopindex,
nocase=not case_var.get(),
)
if position:
start = position
finish = f"{start}+{len(pattern)}c"
text_widget.tag_add("highlight", start, finish)
index = finish
else:
break
text_widget.config(state="disabled")


button_find = tk.Button(
frame_find, width="10", text="Find All", bg="#4a7abc", fg="white",
command=handle_find_all, relief="flat")
button_find.pack(side="left", padx=5, pady=5)
```

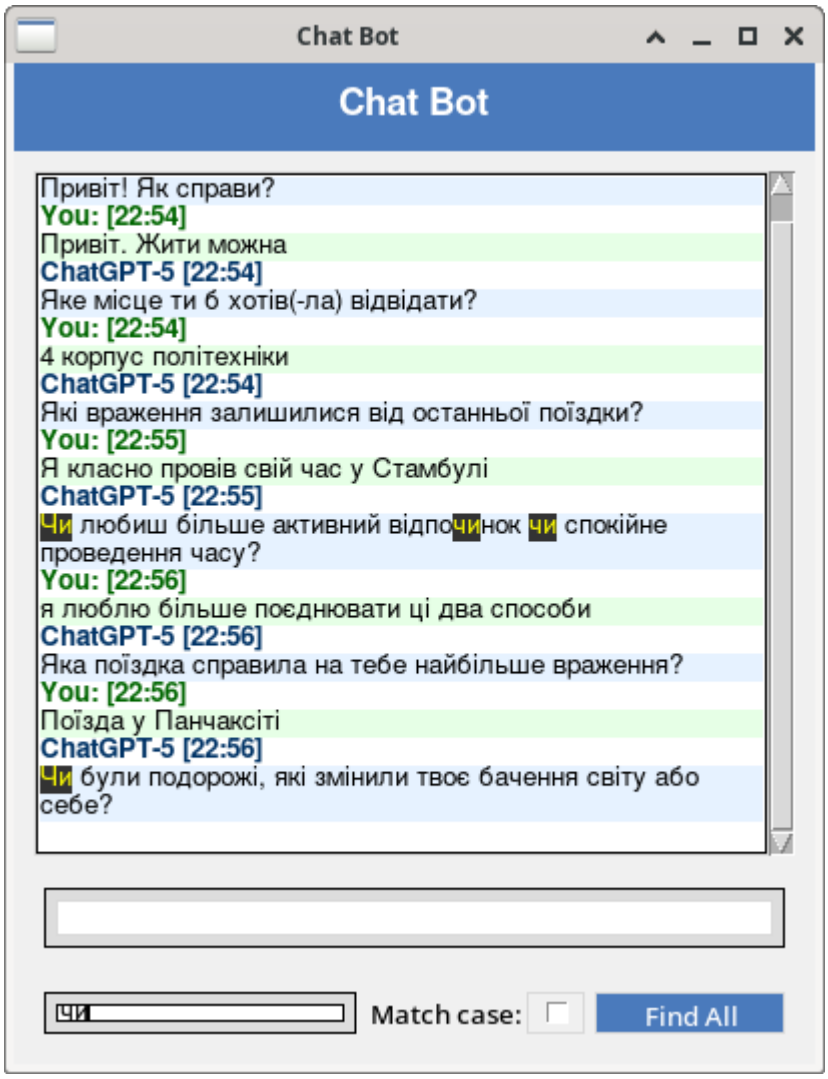
```
entry_send.bind(
"<KeyPress-Return>",
lambda e: handle_send())

root.bind(
"<KeyPress>",
lambda e: text_widget.tag_remove("highlight", "1.0", "end"))

def clear_chat():
global id_message
text_widget.config(state="normal")
text_widget.delete("1.0", tk.END)
text_widget.config(state="disabled")
id_message = 0
create_line("blue", "ChatGPT-5:")
create_line("blue", "Привіт! Як справи?")

# Status bar
status_frame = tk.Frame(root, bg="#e0e0e0", height=20)
status_frame.pack(fill="x", side="bottom")
status_label = tk.Label(status_frame, text="Ready", bg="#e0e0e0", anchor="w", padx=10)
status_label.pack(fill="x")

create_line("blue", "ChatGPT-5")
create_line("blue", "Привіт! Як справи?")
entry_send.focus_set()
root.mainloop()
```

3. Перевірка та тестування програми

3.1. Тестування відповідей



3.2. Тестування пошуку



Висновок

Висновок має відповісти на запитання «Що зроблено?», «Як зроблено?», «Що це дало?».

Що зроблено?

У даному проєкті розроблено програму "Chat Bot" з графічним інтерфейсом користувача, що імітує діалог між користувачем та ботом на тему подорожей. Програма дозволяє обмінюватись повідомленнями, здійснювати пошук тексту та очищати історію спілкування.

Як зроблено?

Програму реалізовано мовою Python з використанням бібліотеки Tkinter для створення графічного інтерфейсу. Розробка включає:

- •Структуровану організацію інтерфейсу з використанням фреймів
- •Налаштування стилів тексту та кольорів для розрізнення повідомлень користувача та бота
- •Додавання часових міток до повідомлень
- •Реалізацію функції пошуку з підсвічуванням знайденого тексту
- •Систему попередньо визначених відповідей бота українською мовою
- •Обробку подій клавіатури для зручності користування

Що це дало?

Отримано функціональний чат-бот з інтуїтивно зрозумілим інтерфейсом, який:

- •Забезпечує імітацію діалогу з ботом на тему подорожей
- •Має візуально приємний дизайн з чітким розділенням повідомлень
- •Пропонує корисні функції для взаємодії (пошук, очищення чату)
- •Демонструє практичне застосування принципів розробки GUI-додатків з використанням Tkinter
- •Показує реалізацію базової логіки діалогової системи з обмеженим набором відповідей

Розроблена програма є прикладом успішної імплементації простого чат-боту з використанням стандартних інструментів Python.

Орієнтовний перелік питань під час захисту:

- Якість та повнота оформлення звіту.
- Переконатися, що параметри менеджерів геометрії налаштовані таким чином, щоб елементи інтерфейсу коректно адаптувалися до зміни розмірів вікна.
- Додати часові мітки для кожного повідомлення.
- Додати можливості вставляти смайлики для покращення емоційної складової спілкування.
- Додати можливість перемикання між темною і світлою темою інтерфейсу.
- Додати швидке меню для комунікації, в якому користувач може вибирати з набору заготовлених відповідей або запитів, що прискорить комунікацію.