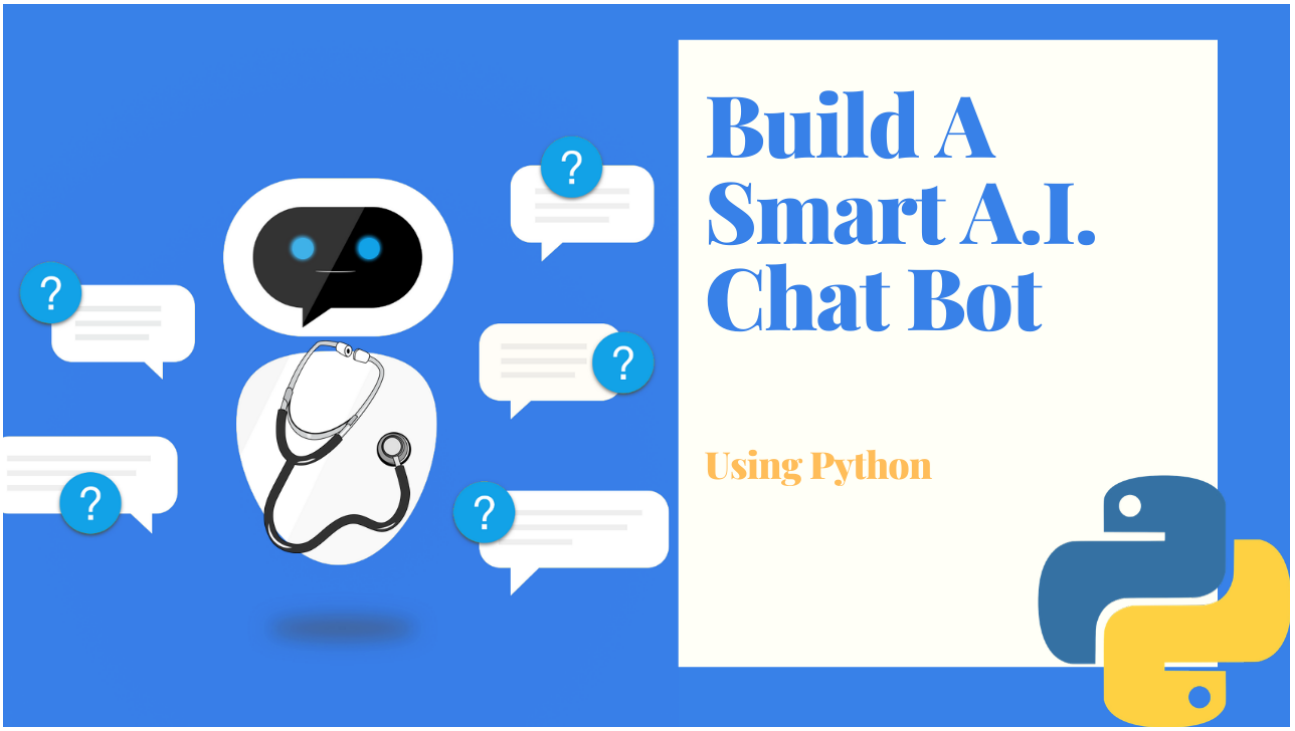
Create a chatbot in Python

Phase 2 Submission Document

Project: Create a chatbot in python



Introduction:

Creating a chatbot in Python is an exciting and practical venture that enables developers to design intelligent conversational agents. In this tutorial, we'll explore the step-by-step process to build a chatbot from scratch. By the end of this journey, you'll have a functional bot capable of engaging users in natural language conversations.

Paragraph 1: Understanding the Basics Before diving into the code, it's crucial to grasp the fundamental concepts. A chatbot is a computer program designed to simulate human conversation. Python, with its extensive libraries, is an ideal choice for chatbot development. We will explore the essentials of Natural Language Processing (NLP) and chatbot architecture to set the stage for our project.

Paragraph 2: Data and Tools Building a chatbot demands access to suitable data and tools. We'll discuss how to collect and preprocess training data, which is crucial for training the chatbot's language model. Python offers a wide range of NLP libraries, such as NLTK and spaCy, that simplify tasks like tokenization and sentiment analysis. Additionally, we'll explore external APIs and libraries like TensorFlow or PyTorch for machine learning purposes.

Paragraph 3: Building the Bot's Brain The core of any chatbot is its intelligence, which comes from its underlying algorithms and models. We'll delve into various NLP techniques, including tokenization, word embeddings, and sequence-to-sequence models, to construct the chatbot's "brain." By integrating machine learning algorithms, we'll train the bot to understand and generate human-like responses.

Paragraph 4: Designing User Interaction User interaction is a pivotal aspect of chatbot development. We will create a user-friendly interface where users can input their queries. Python's Flask framework is a valuable resource for building web-based interfaces, while libraries like tkinter or PyQt can be used for desktop applications. We'll explore how to process user input and make the chatbot respond appropriately.

Paragraph 5: Testing and Deployment To ensure the chatbot functions as intended, rigorous testing is essential. We'll cover techniques for testing the chatbot's responses and fine-tuning its performance. Once satisfied, we'll discuss deployment options, such as hosting the chatbot on a web server or integrating it into messaging platforms like Slack or Facebook Messenger. This final step will bring your Python chatbot to life, ready to assist and engage with users in real-world scenarios.

Data Source:

Dataset Link: (Blended Skill Talk (1 On 1 Conversations) (kaggle.com))

Data Collection and Preprocessing:

➢Importing the dataset: Obtain a comprehensive dataset containing relevant features such as square footage, number of bedrooms, location, amenities, etc.

➢Data preprocessing: Clean the data by handling missing values, outliers, and categorical variables. Standardise or normalise numerical features.

Exploratory Data Analysis (EDA):

➢Visualise and analyse the dataset to gain insights into the relationships between variables.

➢Identify correlations and patterns that can inform feature selection and engineering.

➢Present various data visualisations to gain insights into the dataset.

➢Explore correlations between features and the target variable (house prices).

➢Discuss any significant findings from the EDA phase that inform feature selection.

Program:

```
import nltk

from nltk.chat.util import Chat, reflections



pairs = [
```

```
    [
        r"hi|hello|hey",
        ["Hello!", "Hi there!", "How can I help you today?"]
    ],
    [
        r"what is your name?",
        ["I'm a chatbot.", "You can call me Bot."]
    ],
    [
        r"bye|goodbye",
        ["Goodbye!", "See you later!", "Have a great day!"]
    ],
]
chatbot = Chat(pairs, reflections)
print("Hello! I'm a chatbot. You can start a conversation with me or type 'bye' to exit.")
while True:
    user_input = input("You: ")
    if user_input.lower() == 'bye':
        print("Chatbot: Goodbye!")
```

```
        break

    response = chatbot.respond(user_input)

    print("Chatbot:", response)
```

Conclusion:

Our "Create a Chatbot in Python" project provides a fundamental introduction to chatbot development. We explored the basics of natural language processing, data preprocessing, and the design of chatbot interactions. While this project provides a solid foundation, building a sophisticated chatbot for blended skill talk would require more advanced techniques and datasets. Developing a chatbot is a dynamic journey with the potential to enhance user engagement and automate responses in various domains, opening doors to exciting possibilities in the field of AI and user interaction.