# GPTtutor Setup & Learning Manual

## Section 1: Installation and Setup Workflow

### 1.1 Project Folder Setup

**Directory structure:**

GPTTutor-Decision/  
│  
├── main.py  
├── query\_engine.py  
├── process\_documents.py  
├── requirements.txt  
├── .env  
├── venv/ (virtual environment)

### 1.2 Command Line Basics

* To open Command Prompt inside a folder:
  1. Navigate to the folder in File Explorer.
  2. Type cmd in the address bar and press Enter.

### 1.3 Python Setup Notes

* **Virtual Environment:** A self-contained Python environment that avoids conflicts between projects.
  + Create: python -m venv venv
  + Activate (Windows): venv\Scripts\activate
* **Install dependencies:** pip install -r requirements.txt

### 1.4 requirements.txt Contents

Make sure the file includes:

streamlit  
openai  
sentence-transformers  
faiss-cpu  
PyMuPDF  
python-dotenv

## Section 2: Core Concepts Behind GPTtutor

### What is a Vector Embedding?

* Embedding = a list of numbers (vector) that represents the meaning of text.
* Used for **semantic search**: comparing how *similar* two pieces of text are.
* Typically 768-dimensional (e.g., OpenAI’s text-embedding-3-small).
* Each dimension encodes a nuanced aspect of meaning.

### Why Use Vectors?

* Vectors let the system **search by meaning**, not exact words.
* GPTtutor turns your course content into vectors so it can:
  + Find the most relevant info
  + Answer follow-up questions based on semantic context

### How Vector Comparison Works

* Uses **cosine similarity**: a measure of the angle between two vectors.
* Small angle = high similarity → the system picks that chunk as relevant.

### What is a Transformer?

* A type of neural network designed to **focus attention** on all words in a sentence.
* Unlike older models (which read word-by-word), transformers compare all words at once.
* They compute relationships via *attention scores*.

### How GPT Works

* At each step, it:
  1. Turns context into vectors
  2. Uses attention layers to weigh relevance between tokens
  3. Predicts the **next most likely word** using a probability distribution

### Why GPT Feels Coherent

* It balances short-term memory (recent tokens) and long-term patterns (training data).
* It doesn’t “think” like humans but uses **statistical patterns**.

### How GPT Chooses the Final Output

* For each word prediction, it samples from a probability distribution.
* You can tune randomness:
  + temperature = 0: always pick highest probability
  + temperature > 0: allow some randomness, more creative

### When Does GPT Stop?

* When it generates a special end-of-sequence token
* Or reaches token limit (e.g., 4,096 or 8,192 tokens depending on model)

## Section 3: Key Q&A from Setup and Learning

**Q: What’s a virtual environment and why use it?**  
A: It keeps your project isolated, avoids package conflicts, and makes deployment easier.

**Q: Why do I need to activate the environment each time?**  
A: Because by default, your system Python is used. Activating tells it to use the local one.

**Q: What is** `\*\*?\*\*\ A: A list of dependencies. Usepip install -r requirements.txt` to install them.

**Q: What’s inside** ``**?**  
A: Usually API keys and settings you want to keep private and out of GitHub.

**Q: What’s the difference between Replit, VS Code, Cursor, Render, GitHub Pages?**  
A:

* **Replit**: Easy online editor, good for quick demos.
* **VS Code**: Full-featured local IDE. Best for long-term dev.
* **Cursor**: VS Code with AI help built-in.
* **Render**: Deploy backend code (e.g. GPT API calls).
* **GitHub Pages**: Host static frontends (HTML/CSS/JS), no server logic.

**Q: Can I mix them?**  
A: Yes. You can code in Cursor, push to GitHub, host front on GitHub Pages, and deploy backend via Render.

Let me know if you’d like to add visuals, examples, or connect this with your students’ guide.