PYTHON V3.12.1 *https://www.python.org/doc/versions/*

Quote reuse: in Python 3.11, reusing the same quotes as the enclosing f-string raises a [SyntaxError](https://docs.python.org/release/3.12.1/library/exceptions.html" \l "SyntaxError" \o "SyntaxError), forcing the user to either use other available quotes (like using double quotes or triple quotes if the f-string uses single quotes). In Python 3.12, you can now do things like this:

>>>

**>>>** songs = ['Take me back to Eden', 'Alkaline', 'Ascensionism']

**>>>** f"This is the playlist: **{**", ".join(songs)**}**"

'This is the playlist: Take me back to Eden, Alkaline, Ascensionism'

Multi-line expressions and comments: In Python 3.11, f-string expressions must be defined in a single line, even if the expression within the f-string could normally span multiple lines (like literal lists being defined over multiple lines), making them harder to read. In Python 3.12 you can now define f-strings spanning multiple lines, and add inline comments:

Backslashes and unicode characters: before Python 3.12 f-string expressions couldn’t contain any \ character. This also affected unicode [escape sequences](https://docs.python.org/release/3.12.1/reference/lexical_analysis.html#escape-sequences) (such as \N{snowman}) as these contain the \N part that previously could not be part of expression components of f-strings. Now, you can define expressions like this:

**>>>** print(f"This is the playlist: **{**"**\n**".join(songs)**}**")

This is the playlist: Take me back to Eden

Alkaline

Ascensionism

The bitwise inversion operator (~) on bool is deprecated. It will throw an error in Python 3.14. Use not for logical negation of bools instead. In the rare case that you really need the bitwise inversion of the underlying int, convert to int explicitly: ~int(x).

**Code Generation**

The Retrieval-Augmented Generation model can also be used in code generation tasks. In this case, the retrieval model retrieves the relevant code snippets, and the generation model adapts and extends the code to meet specific project requirements.

The code generation models use RAG to fetch relevant information from the existing code repositories, utilize it to develop accurate code and documentation, and even fix code errors.

**RAG -**

* Converts the natural language descriptions into code implications.
* Predicts the next code bit
* It also converts the code into natural language descriptions
* Generates and runs new code to perform a comprehensive analysis

**RECENT UPDATES:**

**Certainly! Here are 8 Python modules that have recently undergone significant updates, and many large language models (LLMs) might not be fully aware of these changes due to the rapid pace of development:**

1. **FastAPI:**
   * **Recent Updates: FastAPI has seen updates improving its support for Pydantic v2, better async capabilities, and enhanced OpenAPI documentation generation.**
   * **Why LLMs Might Not Know: The updates are relatively recent, and the integration of Pydantic v2 brings many breaking changes and new features.**
2. **Pydantic:**
   * **Recent Updates: Pydantic v2 was released with a complete rewrite, offering better performance, more flexible validation, and improved type hints.**
   * **Why LLMs Might Not Know: The transition from v1 to v2 involves significant changes that might not be fully documented in training data up to 2023.**
   * **https://github.com/pydantic/pydantic/blob/main/docs/errors/validation\_errors.md**
3. **Polars:**
   * **Recent Updates: Polars, a fast DataFrame library, has seen major updates in its API, performance optimizations, and new features like lazy evaluation and better integration with other data science tools.**
   * **Why LLMs Might Not Know: Polars is a relatively new library, and its rapid development means that many updates might not be well-known.**
4. **LangChain:**
   * **Recent Updates: LangChain has added support for more LLMs, improved chain construction, and better integration with external tools and APIs.**
   * **Why LLMs Might Not Know: LangChain is evolving quickly, and new features are being added frequently, which might not be captured in older training data.**
   * **https://github.com/langchain-ai/langchain/tree/master/docs/docs/troubleshooting/errors**
5. **Hugging Face Transformers:**
   * **Recent Updates: The library has seen updates in model support, optimizations for training and inference, and better integration with other Hugging Face tools like Datasets and Tokenizers.**
   * **Why LLMs Might Not Know: The Hugging Face ecosystem is vast and constantly updated, making it hard for LLMs to keep up with every change.**
6. **Ray:**
   * **Recent Updates: Ray has improved its support for distributed computing, better integration with machine learning frameworks, and enhanced fault tolerance.**
   * **Why LLMs Might Not Know: Ray's updates are often focused on performance and scalability, which might not be immediately visible in documentation or tutorials.**
7. **Dask:**
   * **Recent Updates: Dask has seen updates in its DataFrame API, better integration with cloud services, and improved performance for large-scale data processing.**
   * **Why LLMs Might Not Know: Dask's updates are often incremental and focused on specific use cases, which might not be widely covered in general documentation.**
8. **Streamlit:**
   * **Recent Updates: Streamlit has added new components, better state management, and improved performance for building interactive web applications.**
   * **Why LLMs Might Not Know: Streamlit's updates are often focused on user experience and new features, which might not be immediately reflected in older training data.**

| **Module** | **Latest Version (as of 2024)** | **Description** |
| --- | --- | --- |

|  |  |  |
| --- | --- | --- |
| numpy | 1.26.x | Numerical computing |

|  |  |  |
| --- | --- | --- |
| pandas | 2.2.x | Data analysis and manipulation |

|  |  |  |
| --- | --- | --- |
| matplotlib | 3.8.x | Data visualization |

|  |  |  |
| --- | --- | --- |
| seaborn | 0.13.x | Statistical data visualization |

|  |  |  |
| --- | --- | --- |
| scipy | 1.12.x | Scientific computing |

|  |  |  |
| --- | --- | --- |
| tensorflow | 2.15.x | Machine learning framework |

|  |  |  |
| --- | --- | --- |
| torch (PyTorch) | 2.2.x | Deep learning framework |

|  |  |  |
| --- | --- | --- |
| scikit-learn | 1.4.x | Machine learning library |

|  |  |  |
| --- | --- | --- |
| requests | 2.31.x | HTTP requests |

|  |  |  |
| --- | --- | --- |
| beautifulsoup4 | 4.12.x | Web scraping |

|  |  |  |
| --- | --- | --- |
| lxml | 4.9.x | XML and HTML parsing |

|  |  |  |
| --- | --- | --- |
| flask | 3.0.x | Lightweight web framework |

|  |  |  |
| --- | --- | --- |
| django | 4.2.x | Full-stack web framework |

|  |  |  |
| --- | --- | --- |
| fastapi | 0.110.x | High-performance web API framework |

|  |  |  |
| --- | --- | --- |
| pytest | 8.0.x | Testing framework |

|  |  |  |
| --- | --- | --- |
| sqlalchemy | 2.0.x | Database ORM |

|  |  |  |
| --- | --- | --- |
| pillow | 10.2.x | Image processing |

|  |  |  |
| --- | --- | --- |
| openpyxl | 3.1.x | Excel file handling |

|  |  |  |
| --- | --- | --- |
| pytesseract | 0.3.x | OCR (Optical Character Recognition) |

|  |  |  |
| --- | --- | --- |
| opencv-python | 4.9.x | Computer vision |
|  |  |  |

**NUMPY**

<https://chatgpt.com/share/67ca64c8-2a30-8005-a1dc-d6f48f2e5be9>

<https://chatgpt.com/share/67cbcfdf-c190-8005-8b77-232714d1b10a>

<https://chatgpt.com/share/67cfa81b-3d9c-8005-b645-c20f053ea99e>