RAG e Árvore sintática de código para geração de documentação

Dickson Alves de Souza

Especialização em Engenharia de Software – Oferta 7 – Turma 1

PUC Minas

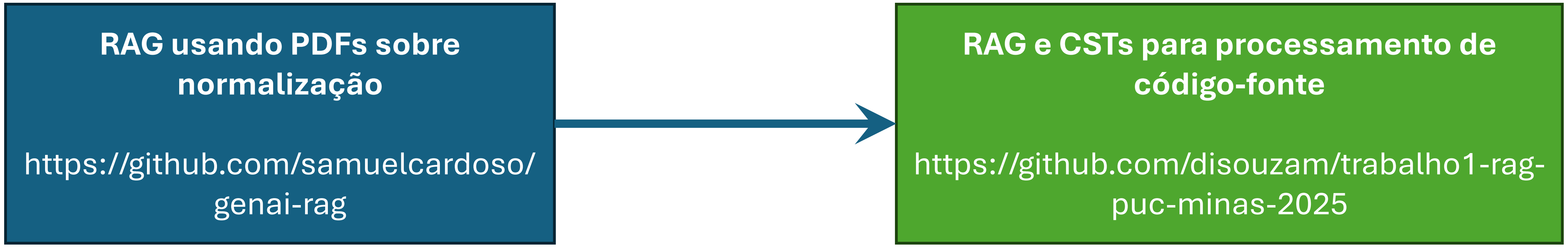
10 de Março de 2025

Professor Samuel Almeida Cardoso

Resumo

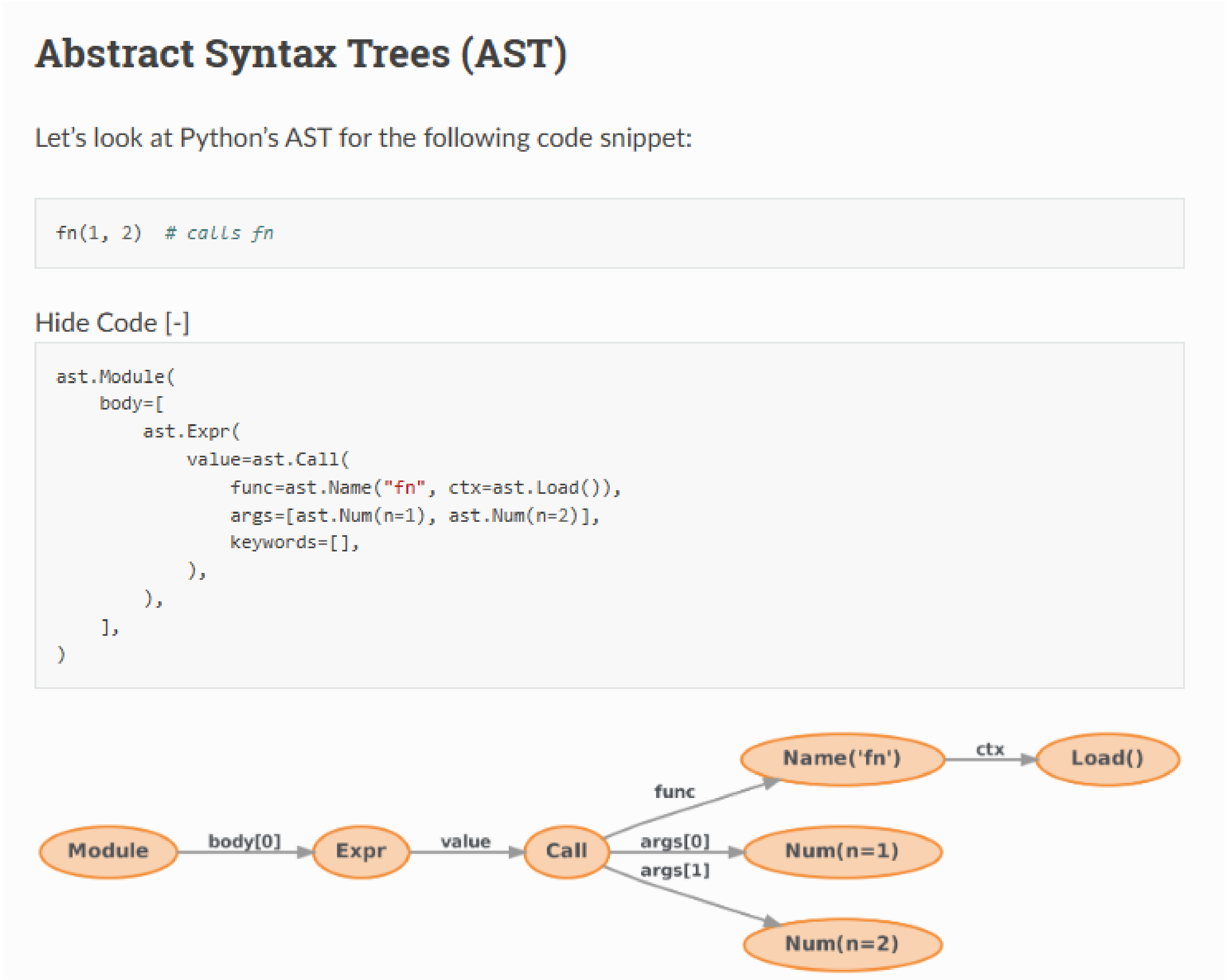
1. Customização de RAG para processamento de PDFs
2. Uso de árvore sintática concreta para processamento semântico de código-fonte em Python
3. Separação de chunks com e sem docstrings
4. Criação do index com os chunks com docstrings
5. Submissão de chunks sem docstrings para geração automatizada de docstrings
6. Inserção das docstrings no arquivo original

Visão geral


RAG usando PDFs sobre normalização
https://github.com/samuelcardoso/genai-rag
RAG e CSTs para processamento de código-fonte
https://github.com/disouzam/trabalho1-rag-puc-minas-2025

Árvore sintática concreta

- Biblioteca libCST do Instagram permite processar e percorrer a árvore sintática de um módulo
- O uso da CST (concrete syntax tree) permite processamento semântico de um arquivo
- É possível, por exemplo, identificar funções com e sem docstrings e ainda retornar o conteúdo inteiro dela
- Essas árvores permitem ferramentas que formatam código funcionar respeitando as regras da linguagem
- Refatorações de código também são possíveis



Abstract Syntax Trees (AST)

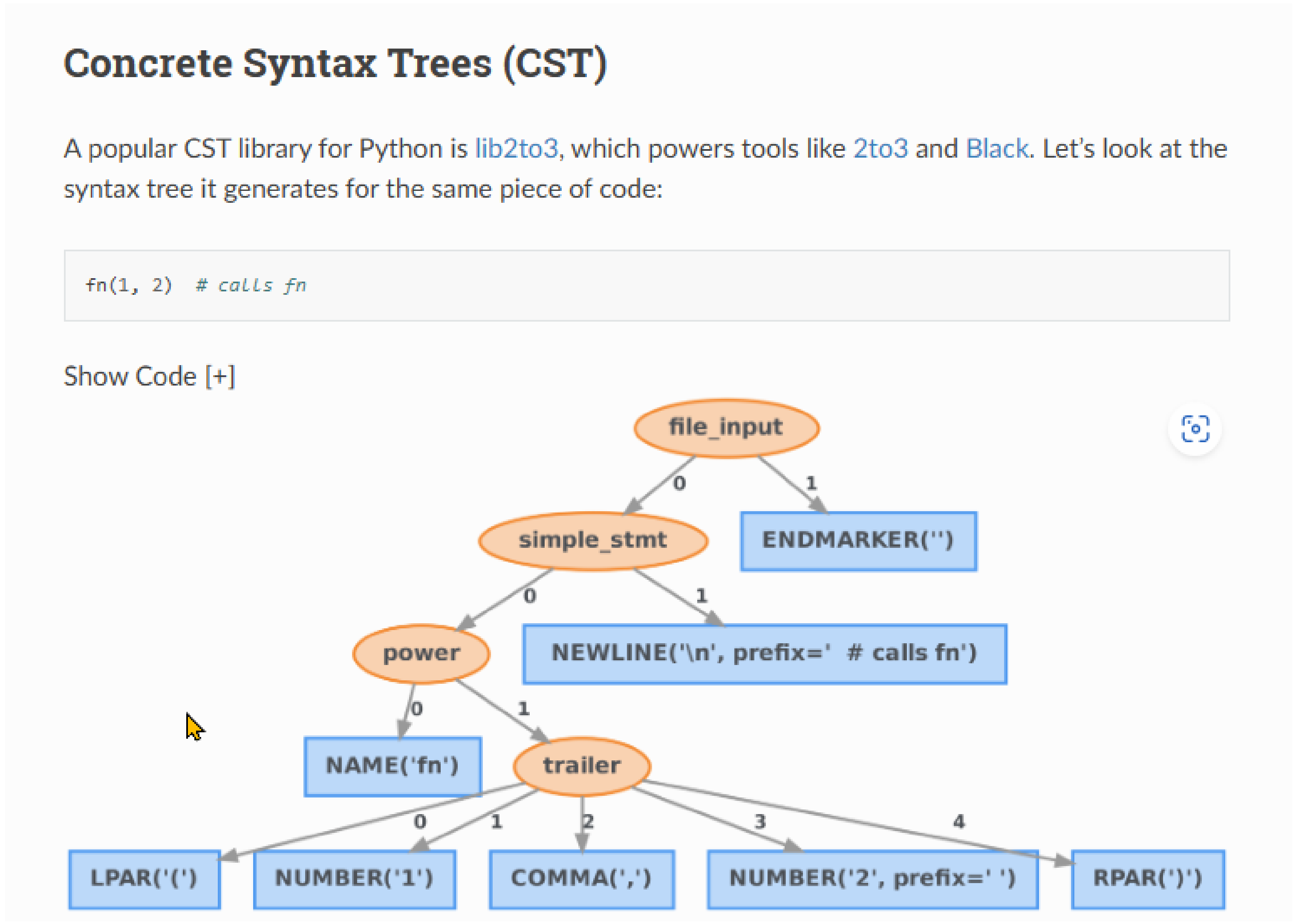
Let's look at Python's AST for the following code snippet:

```
fn(1, 2)  # calls fn
```

Hide Code [-]

```
ast.Module(
    body=[
        ast.Expr(
            value=ast.Call(
                func=ast.Name("fn", ctx=ast.Load()),
                args=[ast.Num(n=1), ast.Num(n=2)],
                keywords=[],
            ),
        ),
    ],
)
```

Figura 1: Árvore sintática abstrata
Fonte: Why LibCST? — LibCST documentation



Concrete Syntax Trees (CST)

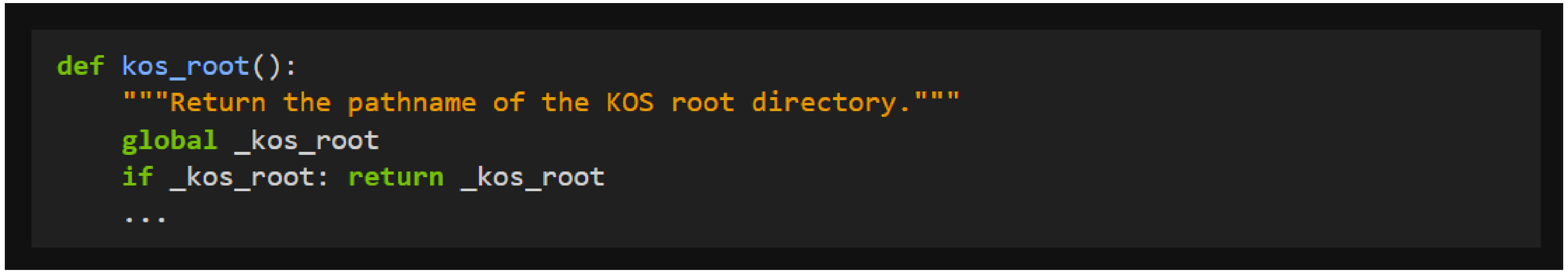
A popular CST library for Python is lib2to3, which powers tools like 2to3 and Black. Let's look at the syntax tree it generates for the same piece of code:

```
fn(1, 2)  # calls fn
```

Show Code [+]

Figura 2: Árvore sintática concreta
Fonte: Why LibCST? — LibCST documentation

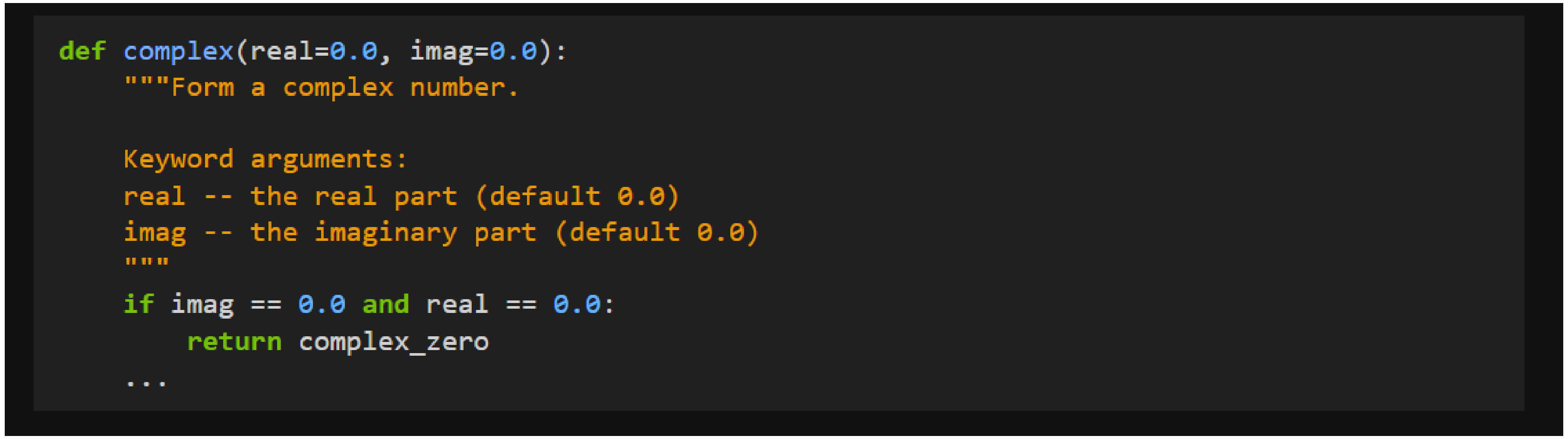
Docstrings – PEP 257



```python
def kos_root():
    """Return the pathname of the KOS root directory."""
    global _kos_root
    if _kos_root: return _kos_root
    ...
```

Docstrings de uma linha só

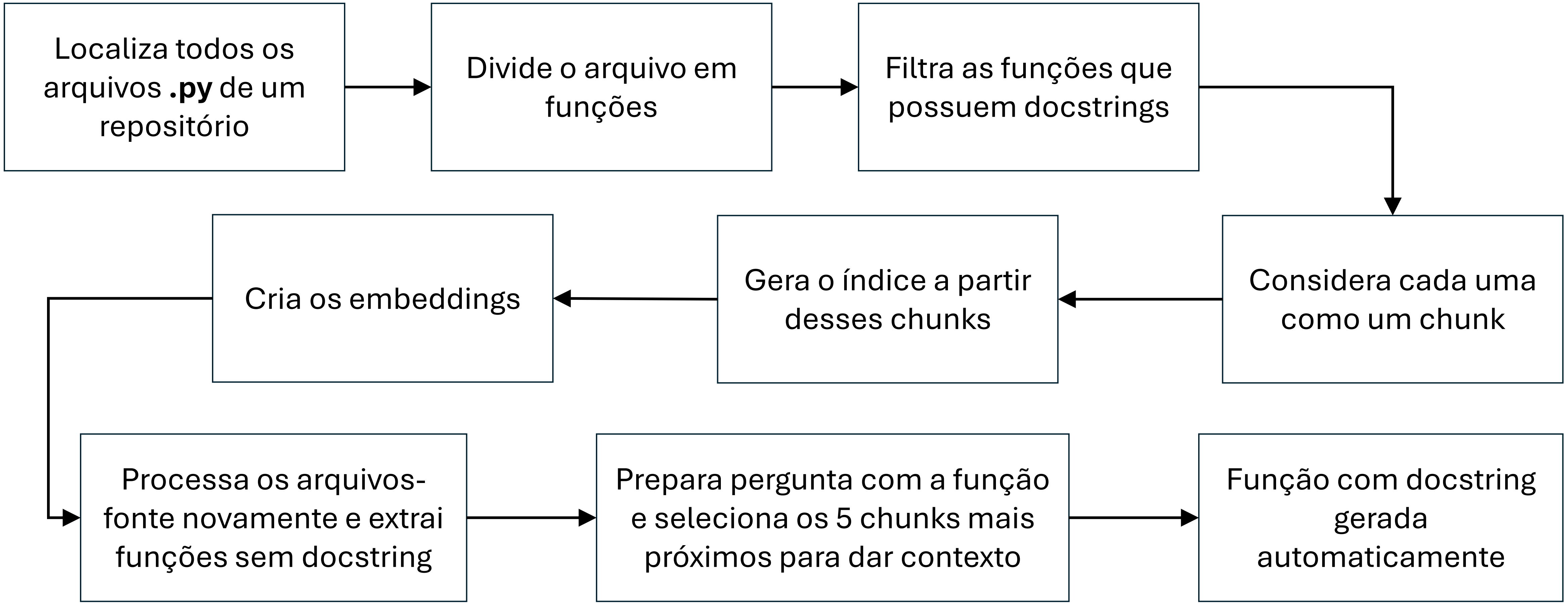
Docstrings de múltiplas linhas

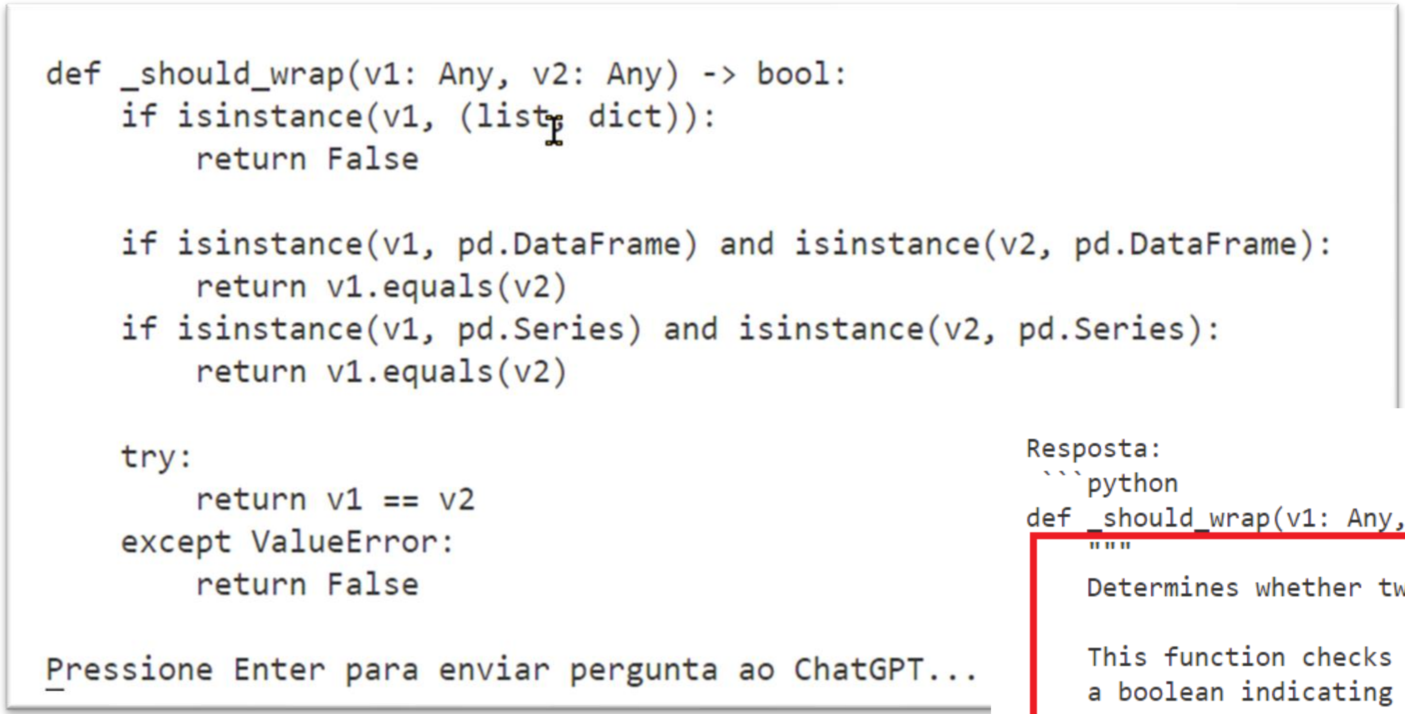


```python
def complex(real=0.0, imag=0.0):
    """Form a complex number.

    Keyword arguments:
    real -- the real part (default 0.0)
    imag -- the imaginary part (default 0.0)
    """
    if imag == 0.0 and real == 0.0:
        return complex_zero
    ...
```

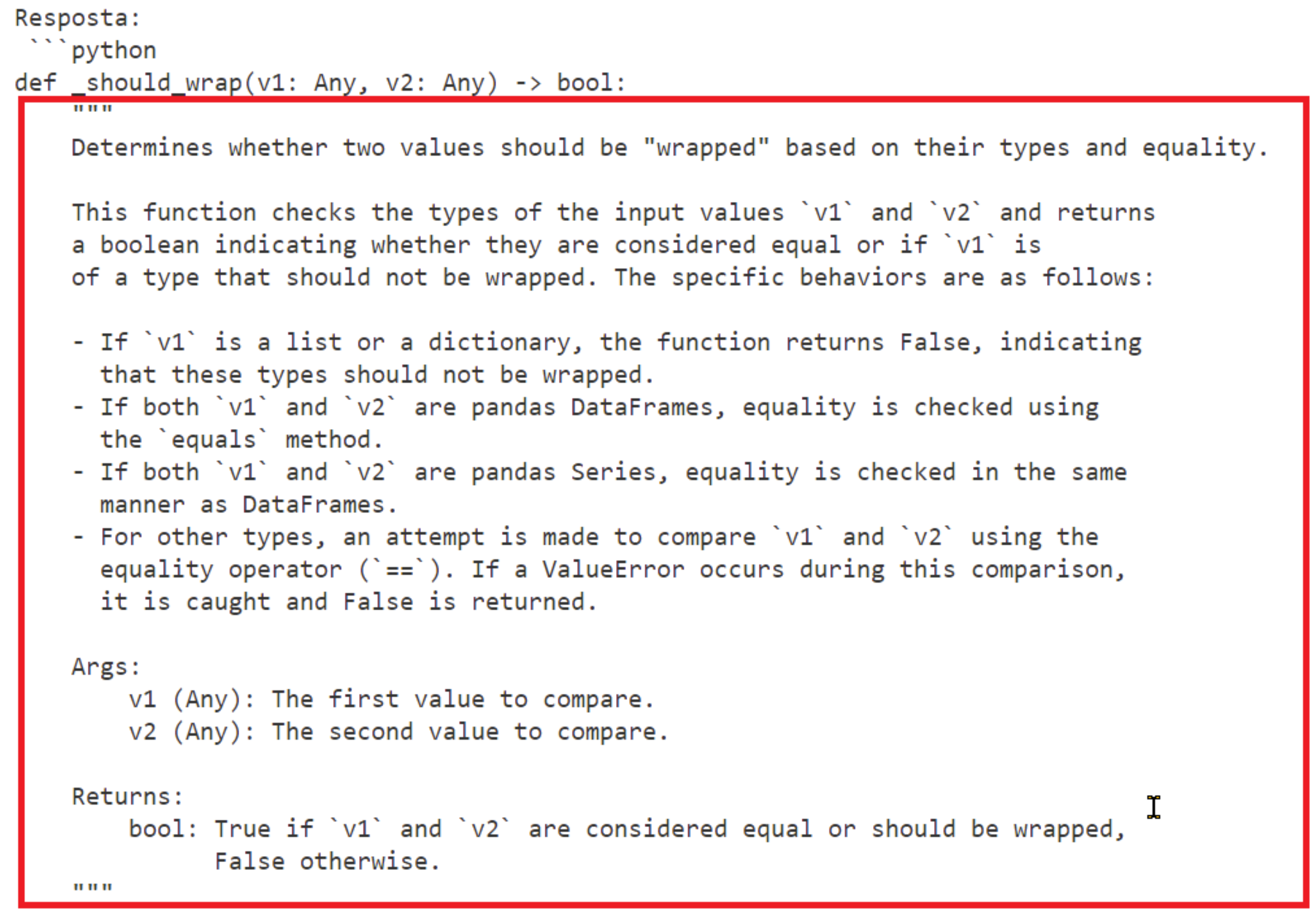
PEP 257 – Docstring Conventions | peps.python.org

Pipeline RAG + Resposta a perguntas

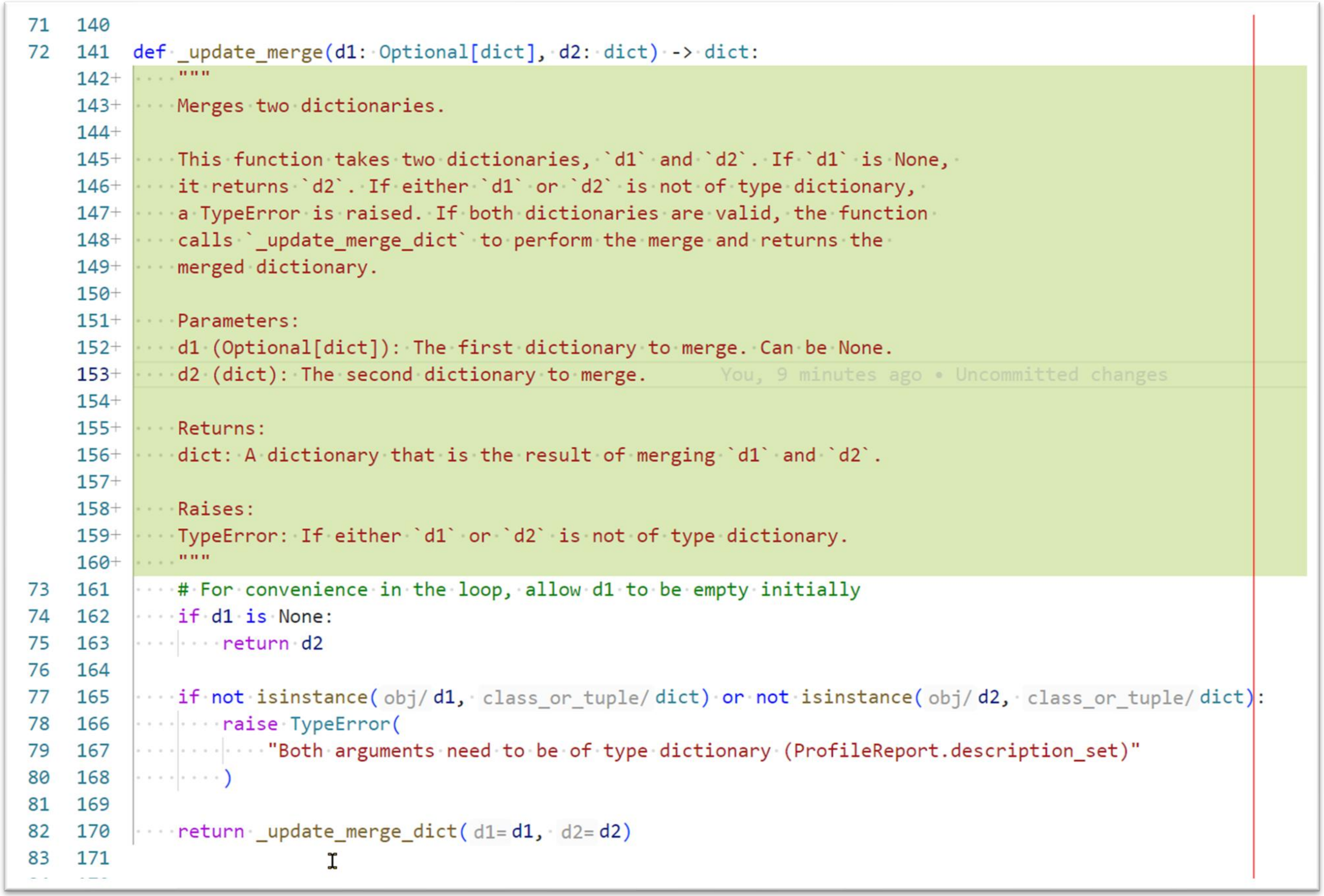

Localiza todos os arquivos **.py** de um repositório
Divide o arquivo em funções
Filtra as funções que possuem docstrings
Considera cada uma como um chunk
Gera o índice a partir desses chunks
Cria os embeddings
Processa os arquivos-fonte novamente e extrai funções sem docstring
Prepara pergunta com a função e seleciona os 5 chunks mais próximos para dar contexto
Função com docstring gerada automaticamente



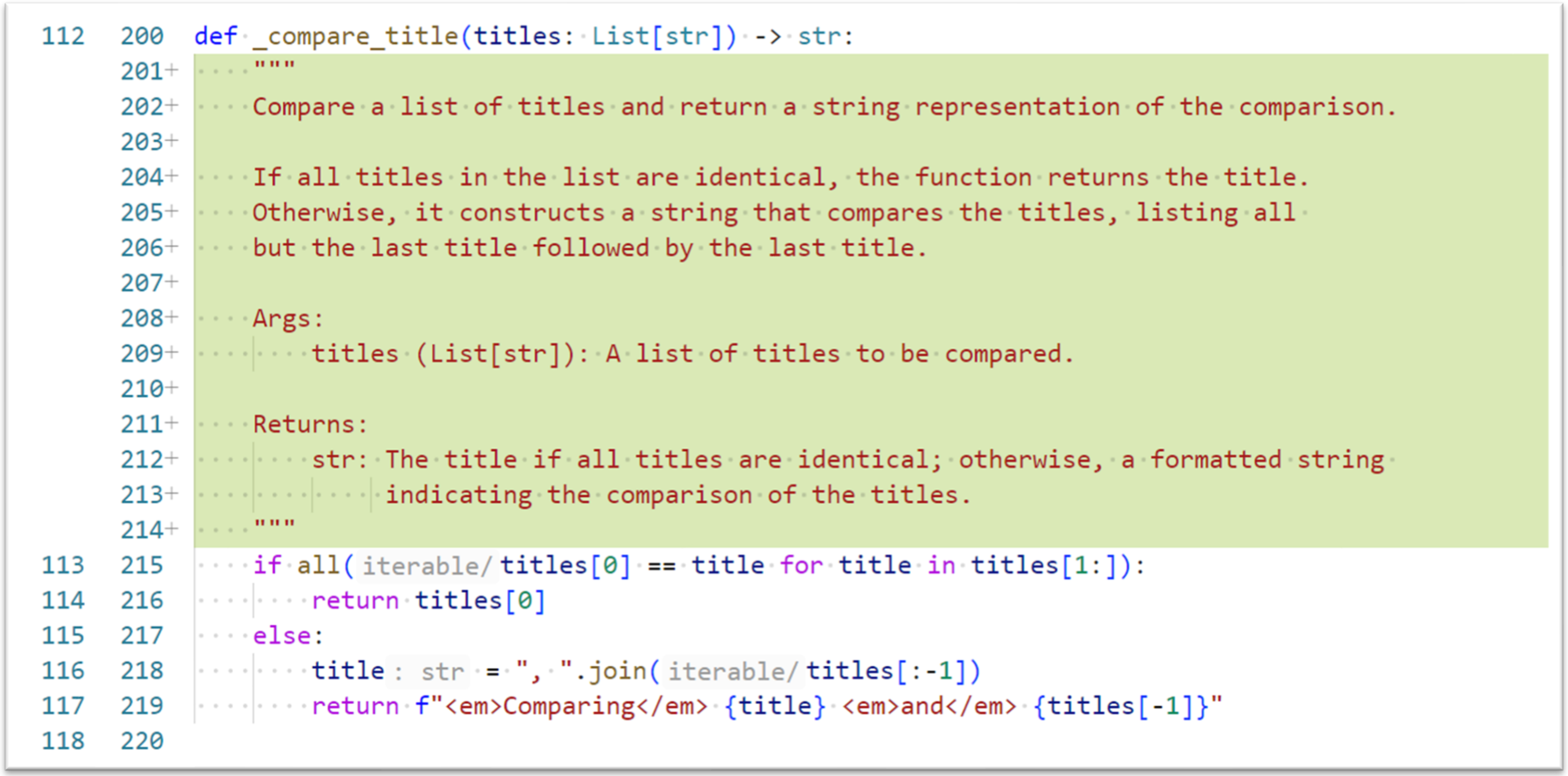
```
def _should_wrap(v1: Any, v2: Any) -> bool:
    if isinstance(v1, (list, dict)):
        return False

    if isinstance(v1, pd.DataFrame) and isinstance(v2, pd.DataFrame):
        return v1.equals(v2)
    if isinstance(v1, pd.Series) and isinstance(v2, pd.Series):
        return v1.equals(v2)

    try:
        return v1 == v2
    except ValueError:
        return False

Pressione Enter para enviar pergunta ao ChatGPT...
```



~~~
Resposta:
```python
def _should_wrap(v1: Any, v2: Any) -> bool:
    """
    Determines whether two values should be "wrapped" based on their types and equality.

    This function checks the types of the input values `v1` and `v2` and returns
    a boolean indicating whether they are considered equal or if `v1` is
    of a type that should not be wrapped. The specific behaviors are as follows:

    - If `v1` is a list or a dictionary, the function returns False, indicating
      that these types should not be wrapped.
    - If both `v1` and `v2` are pandas DataFrames, equality is checked using
      the `equals` method.
    - If both `v1` and `v2` are pandas Series, equality is checked in the same
      manner as DataFrames.
    - For other types, an attempt is made to compare `v1` and `v2` using the
      equality operator (`==`). If a ValueError occurs during this comparison,
      it is caught and False is returned.

    Args:
        v1 (Any): The first value to compare.
        v2 (Any): The second value to compare.

    Returns:
        bool: True if `v1` and `v2` are considered equal or should be wrapped,
              False otherwise.
    """
~~~

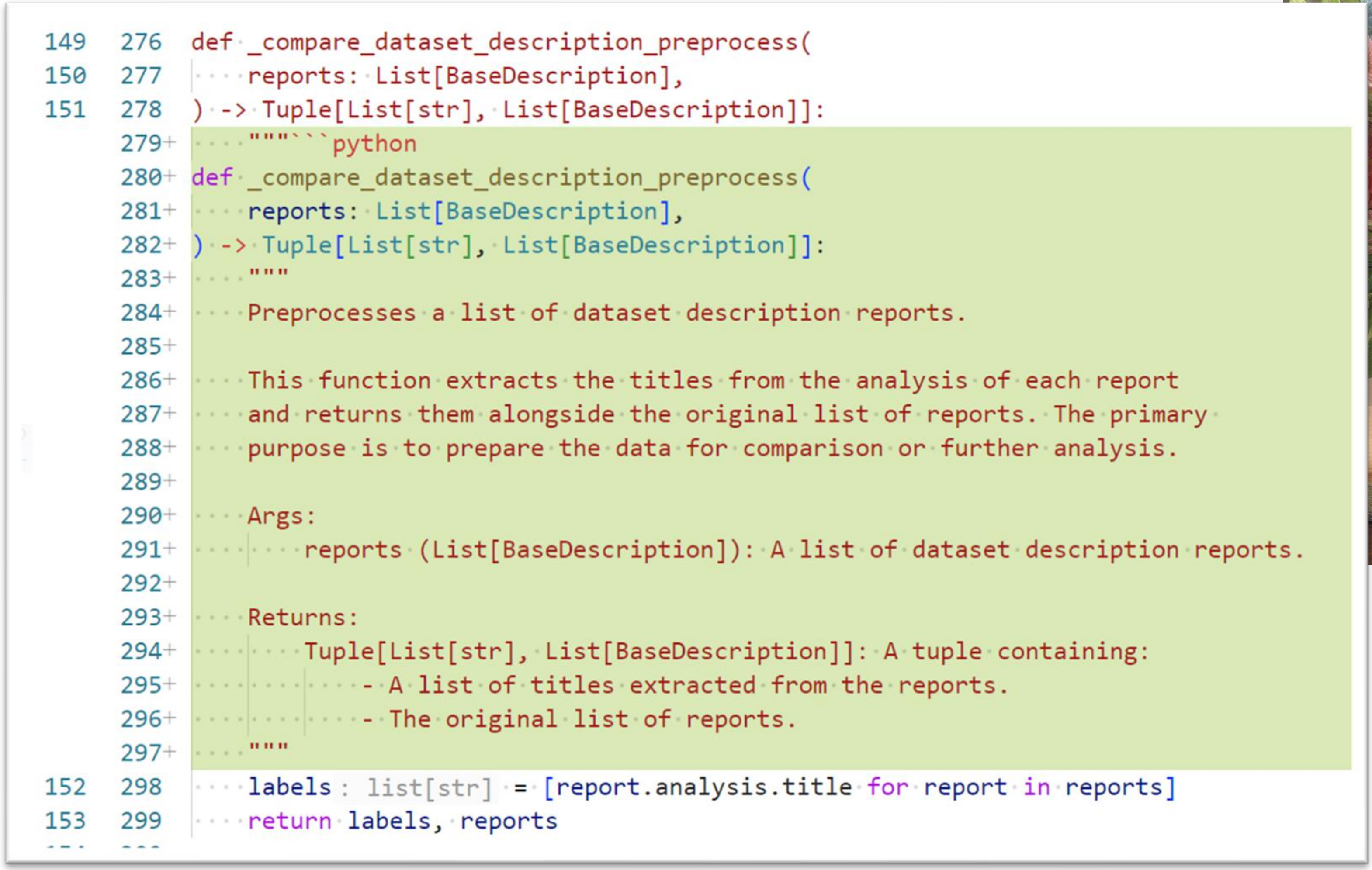


```python

def _update_merge(d1: Optional[dict], d2: dict) -> dict:
    """
    Merges two dictionaries.

    This function takes two dictionaries, `d1` and `d2`. If `d1` is None,
    it returns `d2`. If either `d1` or `d2` is not of type dictionary,
    a TypeError is raised. If both dictionaries are valid, the function
    calls `_update_merge_dict` to perform the merge and returns the
    merged dictionary.

    Parameters:
    d1 (Optional[dict]): The first dictionary to merge. Can be None.
    d2 (dict): The second dictionary to merge.      You, 9 minutes ago • Uncommitted changes

    Returns:
    dict: A dictionary that is the result of merging `d1` and `d2`.

    Raises:
    TypeError: If either `d1` or `d2` is not of type dictionary.
    """
    # For convenience in the loop, allow d1 to be empty initially
    if d1 is None:
        return d2

    if not isinstance(obj/ d1, class_or_tuple/ dict) or not isinstance(obj/ d2, class_or_tuple/ dict):
        raise TypeError(
            "Both arguments need to be of type dictionary (ProfileReport.description_set)"
        )

    return _update_merge_dict(d1= d1, d2= d2)

```

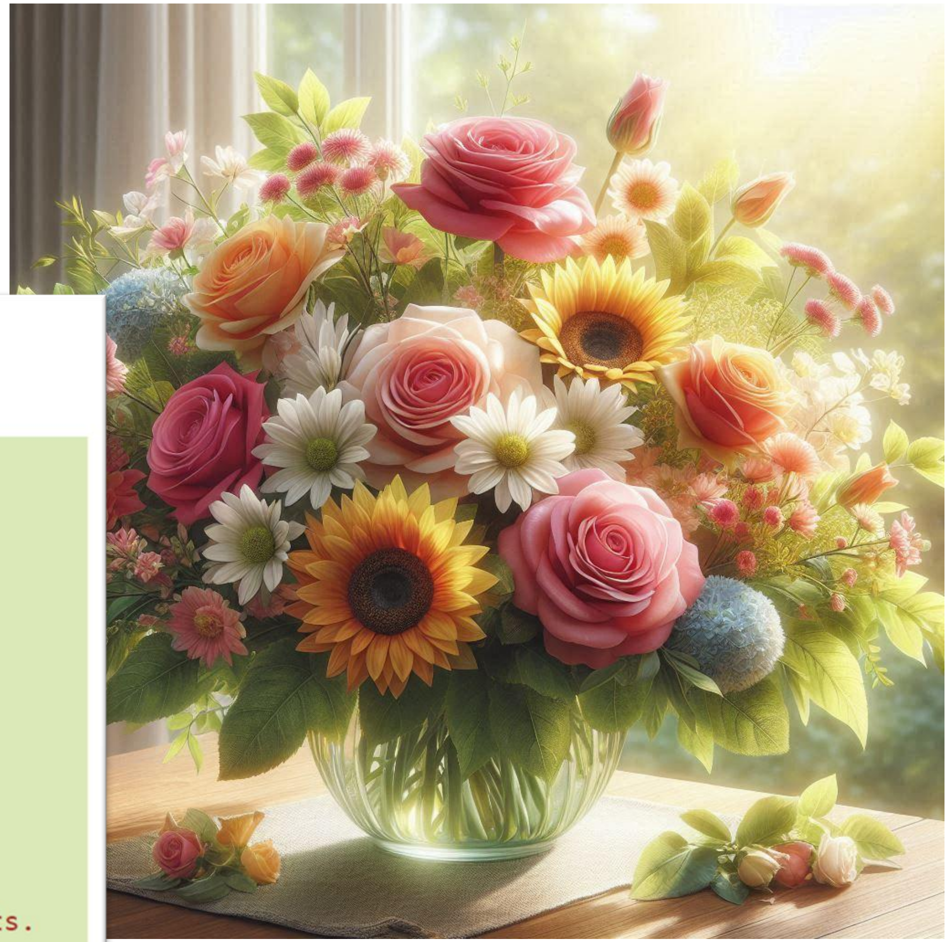


```python
def _compare_title(titles: List[str]) -> str:
    """
    Compare a list of titles and return a string representation of the comparison.

    If all titles in the list are identical, the function returns the title.
    Otherwise, it constructs a string that compares the titles, listing all
    but the last title followed by the last title.

    Args:
        titles (List[str]): A list of titles to be compared.

    Returns:
        str: The title if all titles are identical; otherwise, a formatted string
            indicating the comparison of the titles.
    """
    if all(iterable/ titles[0] == title for title in titles[1:]):
        return titles[0]
    else:
        title : str = ", ".join(iterable/ titles[:-1])
        return f"<em>Comparing</em> {title} <em>and</em> {titles[-1]}"
```

Nem tudo são flores...



```python
def _compare_dataset_description_preprocess(
    reports: List[BaseDescription],
) -> Tuple[List[str], List[BaseDescription]]:
    """```python
def _compare_dataset_description_preprocess(
    reports: List[BaseDescription],
) -> Tuple[List[str], List[BaseDescription]]:
    """
    Preprocesses a list of dataset description reports.

    This function extracts the titles from the analysis of each report
    and returns them alongside the original list of reports. The primary
    purpose is to prepare the data for comparison or further analysis.

    Args:
        reports (List[BaseDescription]): A list of dataset description reports.

    Returns:
        Tuple[List[str], List[BaseDescription]]: A tuple containing:
            - A list of titles extracted from the reports.
            - The original list of reports.
    """
    labels : list[str] = [report.analysis.title for report in reports]
    return labels, reports
```



... Mas essas foram geradas pelo Copilot integrado no Microsoft 365...

Perguntas não respondidas nesse trabalho:

- Separar em chunks semânticos teve realmente diferença no resultado?
- Como a biblioteca testada ydata-profiling (antiga pandas-profiling - https://github.com/ydataai/ydata-profiling) é antiga, será que o RAG fez alguma diferença na construção da resposta?
- A docstring gerada é consistente com todas as funções passadas como pergunta?

Obrigado pela atenção!

Gerado com o Copilot integrado usando o prompt:
“Generate an image of only one software engineer using LLM late at night”