API Documentation

cli

cli

docgen: generate docs & summarize code

generate

Generate docs from comments & docstrings

summarize

Summarize a .py or .ipynb file using OpenAI

full

Generate docs for all .py files under SRC, summarize every .py and .ipynb there, and write a single file to OUT in the specified format.

config

formatter

BaseFormatter

Base class for all formatters

render

Convert parsed items into the target format

get_template

Load a Jinja2 template from the templates directory

MarkdownFormatter

Format parsed items as Markdown

render

HtmlFormatter

Format parsed items as HTML

render

LaTeXFormatter

Format parsed items as LaTeX

render

PDFFormatter

Format parsed items as PDF (via ReportLab)

render

get_formatter

Factory function to get the appropriate formatter

generator

MarkdownGenerator

render

parser

DocItem

extract_comments

Map ending line number → collected comment block above it.

parse_file

summarize

read_code_from_file

Read a Python (.py) or Jupyter notebook (.ipynb) file and return its code as a single string. - .py: returns the entire file. - .ipynb: concatenates all code-cell sources, separated by blank lines.

split_code_by_function_or_class

Split the full code into top■level chunks, each beginning with a `def ` or `class `. Returns up to `max_chunks` segments to avoid overly long prompts.

build_combined_prompt

Build a single user prompt for the LLM by enumerating each chunk. Prepends a header comment "# Chunk i" to each segment for clarity.

summarize_code_file

Orchestrates the full summarization: 1. Reads code from the given file. 2. Splits into manageable chunks. 3. Builds a combined prompt. 4. Calls the OpenAI API to get a summary. 5. Returns the cleaned summary text.

__init__

{'name': 'Summary of cli.py', 'docstring': 'This codebase is a command-line tool for generating documentation from Python files, summarizing code files using OpenAI, and combining these outputs into a single file in different formats like markdown, html, pdf, or latex. It allows generating docs for individual Python files, summarizing them, and creating a comprehensive document. The tool utilizes Click for command-line interface and includes functionalities for generating, summarizing, and combining documentation.', 'params': [], 'returns': ''}

{'name': 'Summary of config.py', 'docstring': 'This code chunk imports necessary modules, loads environment variables from a .env file, and sets default values for variables related to OpenAI API key, model, temperature, max tokens, and maximum code chunks.', 'params': [], 'returns': ''}

{'name': 'Summary of formatter.py', 'docstring': 'These code chunks define a system for generating documentation in different formats (Markdown, HTML, LaTeX, and PDF) using Jinja2 templates and ReportLab for PDF generation. The system includes base classes for formatters, specific formatter classes for each format, and a factory function to obtain the appropriate formatter based on the

desired output format. Each formatter class implements a `render` method to convert parsed items into the corresponding format.', 'params': [], 'returns': ''}

{'name': 'Summary of generator.py', 'docstring': 'The code defines a MarkdownGenerator class with a render method that takes a list of DocItem objects and generates Markdown content based on their type, name, comments, and docstring.', 'params': [], 'returns': ''}

{'name': 'Summary of parser.py', 'docstring': 'The code chunks define a parser module for extracting documentation items from Python files, including functions, classes, and modules. The parser reads the source code, extracts comments, and generates DocItem objects containing information such as name, type, line number, docstring, and comments. The main functions include extract_comments to parse comments and parse_file to extract documentation items from a given file.', 'params': [], 'returns': ''}

{'name': 'Summary of summarize.py', 'docstring': 'The code provided is a Python script that reads code

from a file, splits it into manageable chunks based on function or class definitions, creates a combined prompt for the OpenAI language model, calls the OpenAI API to summarize the code, and returns the cleaned summary text. The script ensures the file exists, reads the code, splits it into chunks, generates a prompt for the language model, queries the OpenAI API for summarization, and provides the resulting summary of the code.', 'params': [], 'returns': ''}

{'name': 'Summary of __init__.py', 'docstring': 'Unfortunately, there is no code chunk provided for me to summarize. If you could provide a code snippet, I would be happy to help summarize it for you.', 'params': [], 'returns': ''}