```
   /     \     |                   |-       |- |         / \           /=\         |-
   | =|= | /=\ |  /=: /=\ /=\=\ /=\   |  /=\   |  |=\ /=\   |\| /=\ | | /=|  \  | | /== |  /=\ /=\=\
    \/ \/  \=  \= \=: \=/ | | | \=    \= \=/   \= | | \=    \ / \=/ \\/ \=| \=/ \=| ==/ \= \=  | | |
                                                                       \=|
aiohttp==3.8.6
aiosignal==1.3.1
astroid==3.0.0
async-timeout==4.0.3
attrs==23.1.0
bandit==1.7.5
bidict==0.22.1
blinker==1.6.3
certifi==2023.7.22
charset-normalizer==3.3.0
click==8.1.7
dill==0.3.7
dparse==0.6.3
Flask==3.0.0
Flask-SocketIO==5.3.6
frozenlist==1.4.0
gitdb==4.0.10
GitPython==3.1.37
h11==0.14.0
idna==3.4
isort==5.12.0
itsdangerous==2.1.2
Jinja2==3.1.2
lazy-object-proxy==1.9.0
markdown-it-py==3.0.0
MarkupSafe==2.1.3
mccabe==0.7.0
mdurl==0.1.2
multidict==6.0.4
netifaces==0.11.0
openai==0.28.1
packaging==23.2
pbr==5.11.1
pipdeptree==2.13.0
platformdirs==3.11.0
Pygments==2.16.1
pylint==3.0.1
pyparsing==3.1.1
python-dotenv==1.0.0
python-engineio==4.7.1
python-socketio==5.9.0
PyYAML==6.0.1
requests==2.31.0
rich==13.6.0
ruamel.yaml==0.17.35
ruamel.yaml.clib==0.2.8
safety==2.3.5
simple-websocket==1.0.0
smmap==5.0.1
stevedore==5.1.0
tomlkit==0.12.1
tqdm==4.66.1
urllib3==2.0.6
```

```
Werkzeug==3.0.0
wrapt==1.15.0
wsproto==1.2.0
yarl==1.9.2
```

Extracted Code Report

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/conftest.py

```
import pytest

@pytest.fixture
def director():
    return Director()  # Replace with the actual object creation logic
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/map_file_structure.py

```
import os
import time
import argparse
import pwd
import grp
import hashlib
import mimetypes
import subprocess
import stat
from pathlib import Path

from stream_to_console import stc



def get_file_details(file_path):
    """
    Retrieves comprehensive details about a file.

    Args:
    file_path (str): Path to the file.

    Returns:
    dict: A dictionary containing various file details.
    """
    try:
        # Basic file stats
        stats = os.stat(file_path)
        file_info = {
            "size": stats.st_size,
            "last_modified": time.ctime(stats.st_mtime),
            "last_accessed": time.ctime(stats.st_atime),
            "created": time.ctime(stats.st_ctime)
        }

        # Owner and permissions
        file_info["owner"] = stat.filemode(stats.st_mode)
        file_info["uid"] = stats.st_uid
        file_info["gid"] = stats.st_gid
        # File hash for integrity
        hasher = hashlib.sha256()
        with open(file_path, 'rb') as file:
            buf = file.read()
            hasher.update(buf)
        file_info["hash"] = hasher.hexdigest()

        # Additional details based on file type
        if file_path.endswith('.py'):  # Example for Python files
            with open(file_path, 'r') as file:
                lines = file.readlines()
                file_info["line_count"] = len(lines)
                # Additional Python-specific analysis can be done here

        return file_info

    except Exception as e:
        return {"error": str(e)}


def parse_arguments():
    parser = argparse.ArgumentParser(description='Generate file structure with optional verbosity and deep s
    parser.add_argument('-v', '--verbose', action='store_true', help='Enable verbose output')
    parser.add_argument('-d', '--deep_scan', action='store_true', help='Enable deep scanning for additional fil
    return parser.parse_args()

def print_verbose(message, status):
    if status == "info":
        # Green for file details, blue for summary headers
        parts = message.split(":")
        colored_message = "\033[32m" + parts[0] + "\033[0m" if len(parts) == 1 else "\033[34m" + parts[0] + ":\0
        print(colored_message)
    else:
        # Default colors for other statuses
        color_code = "32" if status == "success" else "31"
        print(f"\033[{color_code}m{message}\033[0m")

def print_verbose_info(message):
    # Blue color for variable content
    print(f"\033[34m{message}\033[0m", end="")

def print_verbose_label(message):
    # Grey color for non-variable text
    print(f"\033[90m{message}\033[0m", end="")

def color_text(text, color_code):
    return f"\033[{color_code}m{text}\033[0m"

def format_file_size(size):
    for unit in ['B', 'KB', 'MB', 'GB', 'TB']:
        if size < 1024:
            return f"{size:.2f}{unit}"
        size /= 1024

def get_file_permissions(file_path):
    permissions = oct(os.stat(file_path).st_mode)[-3:]
    return permissions

def get_file_owner(file_path):
    uid = os.stat(file_path).st_uid
    gid = os.stat(file_path).st_gid
    user = pwd.getpwuid(uid).pw_name
    group = grp.getgrgid(gid).gr_name
    return user, group

def get_file_hash(file_path):
    sha256_hash = hashlib.sha256()
    with open(file_path, "rb") as f:
        for byte_block in iter(lambda: f.read(4096), b""):
            sha256_hash.update(byte_block)
    return sha256_hash.hexdigest()

def get_file_type(file_path):
    if os.path.islink(file_path):
        return "Symbolic Link"
    elif os.path.isdir(file_path):
        return "Directory"
    elif os.path.isfile(file_path):
        return "File"
    else:
        return "Other"

def get_mime_type(file_path):
    mime_type, _ = mimetypes.guess_type(file_path)
    return mime_type if mime_type else "Unknown"

def get_git_commit_history(file_path, script_dir):
    try:
        cmd = f"git log -n 3 --pretty=format:'%h - %s (%cr)' -- {file_path}"
        process = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, cwd
        stdout, stderr = process.communicate()
        return stdout.decode().strip() if stdout else "Not available"
    except Exception as e:
        return str(e)

def print_summary(total_files, file_types, mime_types):
    print_verbose("File summary:", "info")

    # Print the summary details directly from the variables, not from the file
    print_verbose(f"Total files processed: \033[34m{total_files}\033[32m", "info")
    print_verbose("File types distribution:", "info")
    for f_type, count in file_types.items():
        print_verbose(f"  \033[32m{f_type}: \033[34m{count}\033[32m", "info")
    print_verbose("MIME types distribution:", "info")
    for m_type, count in mime_types.items():
        print_verbose(f"  \033[32m{m_type}: \033[34m{count}\033[32m", "info")
    print('')

def print_deep_scan_summary(deep_scan_details):
    if deep_scan_details:  # Check if there are any deep scan details
        print_verbose("Deep scan details:", "info")
        for file_path, details in deep_scan_details.items():
            detail_parts = [color_text(f"File: {os.path.basename(file_path)}", 34)]
            for key, value in details.items():
                detail_parts.append(color_text(f"{key.capitalize()}: {value}", 34))
            print_verbose(", ".join(detail_parts), "info")
        print('')


def generate_file_structure(script_dir, run_name, base_output_dir='file_tree/runs',
                            skip_dirs=None, include_hidden=False, deep_scan=False, verbose=False):
    if skip_dirs is None:
        skip_dirs = ['bin', 'lib', 'include', 'your_lib_folder', 'archive', '.git', '__pycache__']

    output_dir = os.path.join(base_output_dir, run_name)
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    output_file = os.path.join(output_dir, 'file_structure.txt')
    summary_file = os.path.join(output_dir, 'summary.txt')
    error_log_file = os.path.join(output_dir, 'error_log.txt')

    total_files = 0
    file_types = {}
    mime_types = {}
    deep_scan_details = {}

    def update_distributions(file_type, mime_type):
        nonlocal total_files, file_types, mime_types
        total_files += 1
        file_types[file_type] = file_types.get(file_type, 0) + 1
        mime_types[mime_type] = mime_types.get(mime_type, 0) + 1

    with open(output_file, 'w') as file_out, open(summary_file, 'w') as summary_out, open(error_log_file, 'w') a
        for root, dirs, files in os.walk(script_dir):
            if not include_hidden:
                dirs[:] = [d for d in dirs if not d.startswith('.')]
                files = [f for f in files if not f.startswith('.')]
            dirs[:] = [d for d in dirs if d not in skip_dirs]

            for f in files:
                file_path = os.path.join(root, f)
                try:
                    file_stat = os.stat(file_path)
                    file_size = format_file_size(file_stat.st_size)
                    mod_time = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(file_stat.st_mtime))
                    file_type = get_file_type(file_path)
                    mime_type = get_mime_type(file_path)
                    update_distributions(file_type, mime_type)

                    if deep_scan:
                        file_hash = get_file_hash(file_path) if deep_scan else "N/A"
                        git_history = get_git_commit_history(file_path, script_dir) if deep_scan else "N/A"
                        deep_scan_details[file_path] = {'hash': file_hash, 'git_history': git_history}

                    file_detail = f'{os.path.join(root.replace(script_dir, ""), f)} - Type: {file_type} MIME: {mime_type} S
                    file_out.write(f'{file_detail} | {deep_scan_details} Modified: {mod_time}\n')

                    if verbose:
                        print_verbose_label("Name: ")
                        print_verbose_info(f"{os.path.basename(file_path)}, ")
                        print_verbose_label("Type: ")
                        print_verbose_info(f"{file_type} ")
                        print_verbose_label("MIME: ")
                        print_verbose_info(f"{mime_type} ")
                        print_verbose_label("Size: ")
                        print_verbose_info(f"{file_size} ")
                        print_verbose_label("Last Modified: ")
                        print_verbose_info(f"{mod_time} ")
                        if deep_scan:
                            for file_path, details in deep_scan_details.items():
                                detail_parts = [color_text(f"File: {os.path.basename(file_path)}", 34)]
                                for key, value in details.items():
                                    detail_parts.append(color_text(f"{key.capitalize()}: {value}", 34))
                                print_verbose(", ".join(detail_parts), "info")
                        print('')
                except Exception as e:
                    error_log.write(f"Error processing file {file_path}: {e}\n")
                    if verbose:
                        print_verbose(f"\rError processing file {file_path}: {e}", "error")

        summary_out.write(f'Total files processed: {total_files}\n')
        summary_out.write('File types distribution:\n')
        for f_type, count in file_types.items():
            summary_out.write(f'  {f_type}: {count}\n')
        summary_out.write('MIME types distribution:\n')
        for m_type, count in mime_types.items():
            summary_out.write(f'  {m_type}: {count}\n')



        if deep_scan:
            print_verbose("Deep scan details:", "info")
            for file, details in deep_scan_details.items():
                print_verbose(f"\033[32mFile: \033[34m{file}\033[32m Hash: \033[34m{details['hash']}\033[32m Git

        if verbose:
            print_verbose(f"\rFile structure generation complete. Total files processed: {total_files}", "success")
            print_summary(total_files, file_types, mime_types)

        if verbose and deep_scan:
            print_deep_scan_summary(deep_scan_details)

        # At the end, just print the total files processed
        print(f"File map complete. Total files processed: {total_files}")

if __name__ == '__main__':
    args = parse_arguments()
    verbose = args.verbose
    deep_scan = args.deep_scan
    script_directory = os.getcwd()
    current_time = time.strftime("%Y%m%d_%H%M%S")
    run_name = f'run_{current_time}'
    generate_file_structure(script_directory, run_name, deep_scan=deep_scan, verbose=verbose)
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/test_design_patterns.py

```
# # test_design_patterns.py
# New way of testing

import importlib
import sys

def run_all_tests():
    patterns = [
        "factory", "builder", "prototype", "singleton", "adapter",
        "bridge", "composite", "decorator", "facade", "flyweight",
        "proxy", "chain_of_responsibility", "command", "iterator",
        "observer", "memento", "mediator", "memoize", "state",
        "strategy", "template", "visitor"
        # Add more patterns as needed
    ]

    for pattern in patterns:
        try:
            test_module = importlib.import_module(f"{pattern}.test_{pattern}")
            print(f"\nTesting {pattern.capitalize()} Pattern:")
            test_module.run_tests()
        except ModuleNotFoundError:
            print(f"Test module for {pattern} not found.", file=sys.stderr)
        except AttributeError:
            print(f"No run_tests() function in test_{pattern}.", file=sys.stderr)

if __name__ == "__main__":
    run_all_tests()



# Old way of testing with unittest
# # Test standard design patterns
# from factory.test_factory import main as test_factory_main
# from builder.test_builder import main as test_builder_main
# from prototype.test_prototype import main as test_prototype_main
# from singleton.test_singleton import main as test_singleton_main
# from adapter.test_adapter import main as test_adapter_main
# from bridge.test_bridge import main as test_bridge_main
# from composite.test_composite import main as test_composite_main
# from decorator.test_decorator import main as test_decorator_main
# from facade.test_facade import main as test_facade_main
# from flyweight.test_flyweight import main as test_flyweight_main
# from proxy.test_proxy import main as test_proxy_main
# from chain_of_responsibility.test_chain_of_responsibility import main as test_chain_of_responsibility_mai
# from command.test_command import main as test_command_main
# from iterator.test_iterator import main as test_iterator_main
# from observer.test_observer import main as test_observer_main
# from memento.test_memento import main as test_memento_main
# from mediator.test_mediator import main as test_mediator_main
# from memoize.test_memoize import main as test_memoize_main
# from state.test_state import main as test_state_main
# from strategy.test_strategy import main as test_strategy_main
# from template.test_template import main as test_template_main
# from visitor.test_visitor import main as test_visitor_main
# # Test composite design patterns
# # from event_queue.test_queue import main as test_event_queue_main
# # from thread_pool.test_thread_pool import main as test_thread_pool_main
# # from web_scraper.test_scraper import main as test_scraper_main
# # from zip_file.test_zip_file import main as test_zip_file_main

# def run_all_tests():
#     print("Testing Factory Pattern:")
#     test_factory_main()

#     print("\n\nTesting Builder Pattern:")
#     test_builder_main()

#     print("\n\nTesting Prototype Pattern:")
#     test_prototype_main()

#     print("\n\nTesting Singleton Pattern:")
#     test_singleton_main()

#     print("\n\nTesting Adapter Pattern:")
#     test_adapter_main()

#     print("\n\nTesting Bridge Pattern:")
#     test_bridge_main()

#     print("\n\nTesting Composite Pattern:")
#     test_composite_main()

#     print("\n\nTesting Decorator Pattern:")
#     test_decorator_main()

#     print("\n\nTesting Facade Pattern:")
#     test_facade_main()

#     print("\n\nTesting Flyweight Pattern:")
#     test_flyweight_main()

#     print("\n\nTesting Proxy Pattern:")
#     test_proxy_main()

#     print("\n\nTesting Chain of Responsibility Pattern:")
#     test_chain_of_responsibility_main()

#     print("\n\nTesting Command Pattern:")
#     test_command_main()
#     print("\n\nTesting Iterator Pattern:")
#     test_iterator_main()

#     print("\n\nTesting Observer Pattern:")
#     test_observer_main()

#     print("\n\nTesting State Pattern:")
#     test_state_main()

#     print("\n\nTesting Strategy Pattern:")
#     test_strategy_main()

#     print("\n\nTesting Template Method Pattern:")
#     test_template_main()

#     print("\n\nTesting Visitor Pattern:")
#     test_visitor_main()

#     print("\n\nTesting Memento Pattern:")
#     test_memento_main()

#     print("\n\nTesting Mediator Pattern:")
#     test_mediator_main()

#     print("\n\nTesting Memoize Pattern:")
#     test_memoize_main()

#     # print("\n\nTesting Event Queue Pattern:")
#     # test_event_queue_main()

#     # print("\n\nTesting Thread Pool Pattern:")
#     # test_thread_pool_main()

#     # print("\n\nTesting Web Scraper Pattern:")
#     # test_scraper_main()

#     # print("\n\nTesting Zip File Pattern:")
#     # test_zip_file_main()

#     #===============================================================================#
#     print()

# if __name__ == "__main__":
#     run_all_tests()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/copy_codebase_to_file.py

```
import os
import argparse
import logging
from datetime import datetime
import zipfile
import json
import xml.etree.ElementTree as ET
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

def write_to_pdf(file_data, output_file):
    c = canvas.Canvas(output_file, pagesize=letter)
    width, height = letter
    c.drawString(30, height - 30, "Extracted Code Report")

    y_position = height - 50
    for file_path, content in file_data.items():
        c.drawString(30, y_position, file_path)
        y_position -= 20
        for line in content.split('\n'):
            c.drawString(40, y_position, line)
            y_position -= 15
            if y_position < 40:
                c.showPage()
                y_position = height - 50

    c.save()

def ensure_directory_exists(directory):
    if not os.path.exists(directory):
        os.makedirs(directory)

def write_to_json(file_data, output_file):
    ensure_directory_exists(os.path.dirname(output_file))
    with open(output_file, 'w') as json_file:
        json.dump(file_data, json_file, indent=4)

def write_to_xml(file_data, output_file):
    ensure_directory_exists(os.path.dirname(output_file))
    root = ET.Element("files")
    for file_path, content in file_data.items():
        file_elem = ET.SubElement(root, "file", path=file_path)
        content_elem = ET.SubElement(file_elem, "content")
        content_elem.text = content

    tree = ET.ElementTree(root)
    tree.write(output_file)
```

```
def setup_logging(log_level, log_file=None):
    log_format = '%(asctime)s - %(levelname)s - %(message)s'
    logging.basicConfig(filename=log_file if log_file else None,
                        level=log_level, format=log_format)

def parse_arguments():
    parser = argparse.ArgumentParser(description='Extract Python code from a directory into separate files ir
    parser.add_argument('--directory', type=str, help='Directory to scan for Python files (relative or absolute).')
    parser.add_argument('--output_folder', type=str, help='Folder to write extracted code into separate files (re
    parser.add_argument('--log', type=str, help='Optional log file')
    parser.add_argument('--log_level', type=str, default='INFO', choices=['DEBUG', 'INFO', 'WARNING', 'ERR
                        help='Set the logging level (default: INFO)')
    parser.add_argument('--min_size', type=int, default=0, help='Minimum file size in bytes')
    parser.add_argument('--max_size', type=int, default=None, help='Maximum file size in bytes')
    parser.add_argument('--before_date', type=str, default=None, help='Filter files modified before this date (Y
    parser.add_argument('--format', type=str, default='txt', choices=['txt', 'json', 'xml', 'pdf'],
                        help='Output format: txt, json, xml, or pdf (default: txt)')
    return parser.parse_args()

def is_python_file(file_path):
    return file_path.endswith('.py')

def filter_files(file_path, min_size, max_size, before_date):
    try:
        file_stat = os.stat(file_path)
        file_size = file_stat.st_size
        file_mod_time = datetime.fromtimestamp(file_stat.st_mtime)

        if (min_size is not None and file_size < min_size) or \
           (max_size is not None and file_size > max_size) or \
           (before_date is not None and file_mod_time > before_date):
            return False
        return True
    except Exception as e:
        logging.error(f"Error filtering file {file_path}: {e}")
        return False

def recursive_traverse_directory(directory, min_size, max_size, before_date):
    for root, dirs, files in os.walk(directory):
        for file in files:
            file_path = os.path.join(root, file)
            if is_python_file(file_path) and filter_files(file_path, min_size, max_size, before_date):
                yield file_path

def read_file(file_path):
    try:
        with open(file_path, 'r') as infile:
            content = infile.read()
            return content
    except IOError as e:
        logging.error(f"Error reading file {file_path}: {e}")
        return None

def write_to_single_file(file_path, content, outfile):
    outfile.write(f"\n\n# File: {file_path}\n\n")
    outfile.write(content)

def extract_python_code(directory, output_file, min_size, max_size, before_date):
    try:
        with open(output_file, 'w') as outfile:
            for file_path in recursive_traverse_directory(directory, min_size, max_size, before_date):
                content = read_file(file_path)
                if content:
                    write_to_single_file(file_path, content, outfile)
    except Exception as e:
        logging.error(f"Error while writing to file {output_file}: {e}")

def convert_date_string(date_str):
    return datetime.strptime(date_str, '%Y-%m-%d') if date_str else None

def zip_folder(output_folder, zip_file_name):
    try:
        with zipfile.ZipFile(zip_file_name, 'w', zipfile.ZIP_DEFLATED) as zipf:
            for root, dirs, files in os.walk(output_folder):
                for file in files:
                    file_path = os.path.join(root, file)
                    zipf.write(file_path, os.path.relpath(file_path, output_folder))
        logging.info(f"Folder zipped into: {zip_file_name}")
    except Exception as e:
        logging.error(f"Error zipping folder {output_folder}: {e}")

if __name__ == '__main__':
    print("Running script...")
    args = parse_arguments()
    current_dir = os.getcwd()

    # Set default for directory
    args.directory = args.directory or current_dir

    # Set default for output folder
    cwd_name = os.path.basename(current_dir)
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    args.output_folder = args.output_folder or os.path.join(current_dir, f"{cwd_name}_codebase_copies", f"co
    # Ensure the output folder exists
    ensure_directory_exists(args.output_folder)

    # Generate output filename based on chosen format
    args.format = args.format or 'txt'
    output_file = os.path.join(args.output_folder, f"extracted_code_{timestamp}.{args.format}")

    setup_logging(args.log_level, args.log)

    # Convert date string to datetime object
    before_date = convert_date_string(args.before_date) if args.before_date else None

    # Process files based on the chosen format
    if args.format in ['json', 'xml', 'pdf']:
        file_data = {}
        for file_path in recursive_traverse_directory(args.directory, args.min_size, args.max_size, before_date)
            content = read_file(file_path)
            if content:
                file_data[file_path] = content

        if args.format == 'json':
            write_to_json(file_data, output_file)
        elif args.format == 'xml':
            write_to_xml(file_data, output_file)
        elif args.format == 'pdf':
            write_to_pdf(file_data, output_file)
    else:
        # Default to text format
        extract_python_code(args.directory, output_file, args.min_size, args.max_size, before_date)

    print("Script execution completed.")
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/stream_to_console.py

```
# NovaSystem/src/utils/stream_to_console.py

import traceback
import sys
import time
import art
import random
from colorama import Fore, Back, Style, init
# from ..utils.border_maker import border_maker

# ANSI Escape Codes for additional styles
ANSI_STYLES = {
    "underline": "\033[4m",
    "double_underline": "\033[21m",
    "invert_colors": "\033[7m",
    "italic": "\033[3m",
    "strikethrough": "\033[9m",
    "reset": "\033[0m"
}

# Initialize colorama
init(autoreset=True)

def apply_color(text, foreground_color=None, background_color=None, style=None):
    """Applies color and style to text."""
    colored_text = text
    if foreground_color:
        if hasattr(Fore, foreground_color.upper()):
            colored_text = getattr(Fore, foreground_color.upper()) + colored_text
        else:
            raise ValueError(f"Invalid foreground color: {foreground_color}")

    if background_color:
        if hasattr(Back, background_color.upper()):
            colored_text = getattr(Back, background_color.upper()) + colored_text
        else:
            raise ValueError(f"Invalid background color: {background_color}")

    if style:
        colored_text = style + colored_text
    return colored_text

font_options = [
    "block", "caligraphy","graffiti", "colossal",
    "sub-zero", "slant", "fancy1", "fancy2", "fancy3",
    "fancy4", "fancy5", "fancy6", "fancy7", "fancy8", "fancy9",
    "fancy10", "fancy11", "fancy12", "fancy13", "fancy14",
    "fancy15", "fancy16", "fancy17", "fancy18", "fancy19",
    "fancy20", "banner", "big", "bubble", "digital", "ivrit",
    "mirror", "script", "shadow", "speed", "stampatello",
    "term", "avatar", "barbwire", "bear", "bell", "benjamin",
    "bigchief", "binary", "broadway", "bubblebath", "bulbhead",
    "chunky", "coinstak", "contessa", "contrast", "cosmic",
    "cosmike", "cricket", "cyberlarge", "cybermedium", "cybersmall",
    "decimal", "diamond", "dietcola", "digital", "doh",
    "doom", "dotmatrix", "double", "drpepper", "eftichess",
    "eftifont", "eftipiti", "eftirobot", "eftitalic", "eftiwall",
    "eftiwater", "epic", "fender", "fourtops", "fraktur",
    "goofy", "gothic", "graceful", "gradient", "helv",
    "hollywood", "invita", "isometric1", "isometric2", "isometric3",
    "isometric4", "italic", "jazmine", "jerusalem", "katakana",
    "kban", "keyboard", "knob", "larry3d", "lcd",
    "lean", "letters", "linux", "lockergnome", "madrid",
    "marquee", "maxfour", "mike", "mini", "mirror",
    "mnemonic", "morse", "moscow", "mshebrew210", "nancyj",
    "nancyj-fancy", "nancyj-underlined", "nipples", "ntgreek", "nvscript",
    "o8", "ogre", "pawp", "peaks", "pebbles",
    "pepper", "poison", "puffy", "pyramid", "rectangles",
    "relief", "relief2", "rev", "roman", "rot13",
    "rounded", "rowancap", "rozzo", "runic", "runyc",
    "sblood", "script", "serifcap", "shadow", "short",
    "slscript", "small", "smisome1", "smkeyboard", "smscript",
    "smshadow", "smslant", "smtengwar", "speed", "stampatello",
    "standard", "starwars", "stellar", "stop", "straight",
    "tanja", "tengwar", "term", "thick", "thin",
    "threepoint", "ticks", "ticksslant", "tinker-toy", "tombstone",
    "trek", "tsalagi", "twopoint", "univers", "usaflag",
    "wavy", "weird"
]

def apply_colorama_style(bold=False, underline=False, invert_colors=False, double_underline=False, hidde
    """Returns the combined style string based on flags."""
    style_str = ''
    if bold:
        style_str += Style.BRIGHT
    if hidden:
        style_str += Style.DIM
    if underline:
        style_str += ANSI_STYLES["underline"]
    if double_underline:
        style_str += ANSI_STYLES["double_underline"]
    if invert_colors:
        style_str += ANSI_STYLES["invert_colors"]
    if italic:
        style_str += ANSI_STYLES["italic"]
    if strikethrough:
        style_str += ANSI_STYLES["strikethrough"]
    if fg_style:
        if fg_style == "DIM":
            style_str += Style.DIM
        if fg_style == "BRIGHT":
            style_str += Style.BRIGHT
        if fg_style == "NORMAL":
            style_str += Style.NORMAL
        if fg_style == "RESET_ALL":
            style_str += Style.RESET_ALL
    if bg_style:
        if bg_style == "DIM":
            style_str += Style.DIM
        if bg_style == "BRIGHT":
            style_str += Style.BRIGHT
        if bg_style == "NORMAL":
            style_str += Style.NORMAL
        if bg_style == "RESET_ALL":
            style_str += Style.RESET_ALL
    if style:
        if style == "DIM":
            style_str += Style.DIM
        if style == "BRIGHT":
            style_str += Style.BRIGHT
        if style == "NORMAL":
            style_str += Style.NORMAL
        if style == "RESET_ALL":
            style_str += Style.RESET_ALL
    return style_str

def stream_to_console(message, delay=0.0035, foreground_color=None, background_color=None, rainbow
    """
    Streams a message to the console character by character with optional delay, colors, and effects.
    """
    # Validate input types
    if not isinstance(message, str):
        raise TypeError("Message must be a string.")
    if not isinstance(delay, (float, int)):
        raise TypeError("Delay must be a number.")

    # Stream function
    try:
        # Validate delay
        delay = max(0.0001, min(delay, 1.0))  # Clamp delay

        # Style string
        style_str = apply_colorama_style(**style_flags)

        # Stream each character
        for char in message:
            if rainbow_effect:
                fg_color = random.choice(["RED", "GREEN", "YELLOW", "BLUE", "MAGENTA", "CYAN"])
                char = apply_color(char, foreground_color=fg_color, background_color=background_color, style=s
            else:
                char = apply_color(char, foreground_color, background_color, style_str)

            sys.stdout.write(char)
            sys.stdout.flush()
            time.sleep(delay)
        # Reset color at the end
        sys.stdout.write(Style.RESET_ALL)
        sys.stdout.flush()
    except Exception as e:
        exc_type, exc_value, exc_traceback = sys.exc_info()
        traceback_details = {
            'filename': exc_traceback.tb_frame.f_code.co_filename,
            'lineno': exc_traceback.tb_lineno,
            'name': exc_traceback.tb_frame.f_code.co_name,
            'type': exc_type.__name__,
            'message': str(exc_value),
        }
        error_message = "Error in stream_to_console: [{}] {}".format(traceback_details['type'], traceback_details
        error_details = "File: {}, Line: {}, In: {}".format(traceback_details['filename'], traceback_details['lineno'], t
        sys.stderr.write(error_message + "\n" + error_details + "\n")
        sys.stderr.flush()
        raise

    print()  # Newline at the end
# Example usage and test cases remain the same

# Example usage
# stream_to_console("Hello, NovaSystem AI!", rainbow_effect=True)

# Test cases as a list of dictionaries
test_cases = [
    {"message": "Simple message with default settings."},
    {"message": "Slower text...", "delay": 0.05},
    {"message": "Red text.", "foreground_color": "red"},
    {"message": "Green text.", "foreground_color": "green"},
    {"message": "Green on blue.", "foreground_color": "green", "background_color": "blue"},
    {"message": "Rainbow effect!", "rainbow_effect": True},
    {"message": "Slower rainbow text...", "delay": 0.07, "rainbow_effect": True},
    {"message": "Green on red, slowly.", "delay": 0.05, "foreground_color": "green", "background_color": "red"
    {"message": "Bold text.", "bold": True},
    {"message": "Underlined text.", "underline": True},
    {"message": "Inverted colors.", "invert_colors": True},
    {"message": "Blue background.", "background_color": "blue"},
    {"message": "Cyan text on yellow.", "foreground_color": "cyan", "background_color": "yellow"},
    {"message": "Double underline.", "double_underline": True},
    {"message": "Hidden text.", "hidden": True},
    {"message": "Slower inverted rainbow text...", "delay": 0.07, "rainbow_effect": True, "invert_colors": True},
    {"message": "Italicized text.", "italic": True},
    {"message": "Strikethrough text.", "strikethrough": True},
]
```

```
def test():
    # Generate ASCII art with a random font
  random_font = random.choice(font_options)
  random_ascii_art = art.text2art("NovaSystem", font=random_font)

  # Stream the ASCII art first
  stream_to_console(random_ascii_art, delay=0.0004)

  # Stream each test case
  for case in test_cases:
      stream_to_console(**case)

stc = stream_to_console

if __name__ == "__main__":
    test()
```
/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/proxy/proxy.py
```
from abc import ABC, abstractmethod
import datetime
now = datetime.datetime.now()

class Subject(ABC):
    """
    The Subject interface declares common operations for both RealSubject and the Proxy.
    """
    @abstractmethod
    def request(self) -> None:
        pass

class RealSubject(Subject):
    """
    The RealSubject contains core business logic.
    """
    def request(self) -> None:
        print("RealSubject: Handling request.")

class Proxy(Subject):
    """
    The Proxy has an interface identical to the RealSubject.
    """
    def __init__(self, real_subject: RealSubject) -> None:
        self._real_subject = real_subject

    def request(self) -> None:
        if self.check_access():
            self._real_subject.request()
            self.log_access()
    def check_access(self) -> bool:
        print("Proxy: Checking access prior to firing a real request.")
        return True

    def log_access(self) -> None:
        print("Proxy: Logging the time of request.", end="")
        print(f"Time: {now.time()}")

# Client code example
def client_code(subject: Subject) -> None:
    subject.request()

# Example usage
if __name__ == "__main__":
    real_subject = RealSubject()
    proxy = Proxy(real_subject)

    print("Client: Executing with RealSubject:")
    client_code(real_subject)

    print("\nClient: Executing with Proxy:")
    client_code(proxy)
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/proxy/__init__.py

```
from .proxy import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/proxy/test_proxy.py

```
from proxy import RealSubject, Proxy, client_code

def test_real_subject():
    print("Testing RealSubject:")
    real_subject = RealSubject()
    client_code(real_subject)

def test_proxy():
    print("\nTesting Proxy:")
    real_subject = RealSubject()
    proxy = Proxy(real_subject)
    client_code(proxy)

def main():
    test_real_subject()
    test_proxy()

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/decorator/decorator.py

```
class AIComponent:
    """
    Base AIComponent interface defines operations that can be altered by decorators.
    """
    def operation(self) -> str:
        pass

class ConcreteAIComponent(AIComponent):
    """
    Concrete AIComponents provide default implementations of the operations.
    """
    def operation(self) -> str:
        return "ConcreteAIComponent"

class Decorator(AIComponent):
    """
    Base Decorator class follows the same interface as other components.
    """
    _component: AIComponent = None

    def __init__(self, component: AIComponent) -> None:
        self._component = component

    def operation(self) -> str:
        return self._component.operation()

class LoggingDecorator(Decorator):
    """
    Concrete Decorator that adds logging functionality.
    """
    def operation(self) -> str:
        # Additional behavior before calling the wrapped object
        result = self._component.operation()
        # Additional behavior after calling the wrapped object
        return f"LoggingDecorator({result})"

class PerformanceDecorator(Decorator):
    """
    Concrete Decorator that adds performance tracking functionality.
    """
    def operation(self) -> str:
        # Performance tracking behavior
        result = self._component.operation()
        # Additional behavior
        return f"PerformanceDecorator({result})"
```

```
# Client code
def client_code(component: AIComponent) -> None:
    print(f"RESULT: {component.operation()}", end="")

# Example usage
if __name__ == "__main__":
    simple_component = ConcreteAIComponent()
    decorated_component = PerformanceDecorator(LoggingDecorator(simple_component))

    print("Client: I've got a component with additional behaviors:")
    client_code(decorated_component)
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/decorator/__init__.py

```
from .decorator import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/decorator/test_decorator.p

```
from .decorator import ConcreteAIComponent, LoggingDecorator, PerformanceDecorator

def test_decorator_pattern():
    # Testing with the basic AI component
    basic_component = ConcreteAIComponent()
    print("Basic AI Component:", basic_component.operation())

    # Adding Logging functionality
    logged_component = LoggingDecorator(basic_component)
    print("Logged AI Component:", logged_component.operation())

    # Adding Performance tracking on top of Logging
    perf_logged_component = PerformanceDecorator(logged_component)
    print("Performance Tracked and Logged AI Component:", perf_logged_component.operation())

def main():
    test_decorator_pattern()

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/bridge/bridge.py

```
# bridge.py

from __future__ import annotations
from abc import ABC, abstractmethod

class Abstraction:
    """
    The Abstraction defines the interface for the 'control' part of the two
    class hierarchies. It maintains a reference to an object of the
    Implementation hierarchy and delegates all of the real work to this object.
    """
    def __init__(self, implementation: Implementation) -> None:
        self.implementation = implementation

    def operation(self) -> str:
        return (f"Abstraction: Base operation with:\n"
                f"{self.implementation.operation_implementation()}")

class ExtendedAbstraction(Abstraction):
    """
    You can extend the Abstraction without changing the Implementation classes.
    """
    def operation(self) -> str:
        return (f"ExtendedAbstraction: Extended operation with:\n"
                f"{self.implementation.operation_implementation()}")

class Implementation(ABC):
    """
    The Implementation defines the interface for all implementation classes. It
    doesn't have to match the Abstraction's interface. In fact, the two
    interfaces can be entirely different. Typically the Implementation interface
    provides only primitive operations, while the Abstraction defines higher-
    level operations based on those primitives.
    """
    @abstractmethod
    def operation_implementation(self) -> str:
        pass

class ConcreteImplementationA(Implementation):
    def operation_implementation(self) -> str:
        return "ConcreteImplementationA: Here's the result on the platform A."

class ConcreteImplementationB(Implementation):
    def operation_implementation(self) -> str:
        return "ConcreteImplementationB: Here's the result on the platform B."

def client_code(abstraction: Abstraction) -> None:
    """
    Except for the initialization phase, where an Abstraction object gets linked
    with a specific Implementation object, the client code should only depend on
    the Abstraction class. This way the client code can support any abstraction-
    implementation combination.
    """
    print(abstraction.operation(), end="")

if __name__ == "__main__":
    """
    The client code should be able to work with any pre-configured abstraction-
    implementation combination.
    """
    implementation = ConcreteImplementationA()
    abstraction = Abstraction(implementation)
    client_code(abstraction)

    print("\n")

    implementation = ConcreteImplementationB()
    abstraction = ExtendedAbstraction(implementation)
    client_code(abstraction)
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/bridge/__init__.py

```
from .bridge import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/bridge/test_bridge.py

```
# test_bridge.py

from bridge import Abstraction, ExtendedAbstraction, ConcreteImplementationA, ConcreteImplementationB

def test_abstraction_with_concrete_implementation_a():
    """
    Test Abstraction with ConcreteImplementationA.
    """
    implementation = ConcreteImplementationA()
    abstraction = Abstraction(implementation)
    result = abstraction.operation()

    assert result == "Abstraction: Base operation with:\nConcreteImplementationA: Here's the result on the pl
        "Abstraction with ConcreteImplementationA failed"

    print("PASS: Abstraction with ConcreteImplementationA")

def test_abstraction_with_concrete_implementation_b():
    """
    Test Abstraction with ConcreteImplementationB.
    """
    implementation = ConcreteImplementationB()
    abstraction = Abstraction(implementation)
    result = abstraction.operation()

    assert result == "Abstraction: Base operation with:\nConcreteImplementationB: Here's the result on the pl
        "Abstraction with ConcreteImplementationB failed"

    print("PASS: Abstraction with ConcreteImplementationB")
```

```
def test_extended_abstraction_with_concrete_implementation_a():
    """
    Test ExtendedAbstraction with ConcreteImplementationA.
    """
    implementation = ConcreteImplementationA()
    abstraction = ExtendedAbstraction(implementation)
    result = abstraction.operation()

    assert result == "ExtendedAbstraction: Extended operation with:\nConcreteImplementationA: Here's the re
        "ExtendedAbstraction with ConcreteImplementationA failed"

    print("PASS: ExtendedAbstraction with ConcreteImplementationA")

def test_extended_abstraction_with_concrete_implementation_b():
    """
    Test ExtendedAbstraction with ConcreteImplementationB.
    """
    implementation = ConcreteImplementationB()
    abstraction = ExtendedAbstraction(implementation)
    result = abstraction.operation()

    assert result == "ExtendedAbstraction: Extended operation with:\nConcreteImplementationB: Here's the re
        "ExtendedAbstraction with ConcreteImplementationB failed"

    print("PASS: ExtendedAbstraction with ConcreteImplementationB")

def main():
    """
    Main function to run the Bridge pattern tests.
    """
    print("Testing Bridge Pattern Implementations:")
    test_abstraction_with_concrete_implementation_a()
    test_abstraction_with_concrete_implementation_b()
    test_extended_abstraction_with_concrete_implementation_a()
    test_extended_abstraction_with_concrete_implementation_b()

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/template/__init__.py

```
from .template import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/template/template.py

```
from abc import ABC, abstractmethod

class AbstractClass(ABC):
    """
    The Abstract Class defines a template method that contains a skeleton of
    some algorithm, composed of calls to (usually) abstract primitive operations.
    Concrete subclasses should implement these operations, but leave the
    template method itself intact.
    """
    def template_method(self) -> None:
        self.base_operation1()
        self.required_operations1()
        self.base_operation2()
        self.hook1()
        self.required_operations2()
        self.base_operation3()
        self.hook2()

    # These operations already have implementations.
    def base_operation1(self):
        print("AbstractClass says: I am doing the bulk of the work")

    def base_operation2(self):
        print("AbstractClass says: But I let subclasses override some operations")

    def base_operation3(self):
        print("AbstractClass says: But I am doing the majority of the work anyway")

    # These operations have to be implemented in subclasses.
    @abstractmethod
    def required_operations1(self):
        pass

    @abstractmethod
    def required_operations2(self):
        pass

    # These are "hooks." Subclasses may override them, but it's not mandatory
    # since the hooks already have default (but empty) implementation.
    def hook1(self):
        pass

    def hook2(self):
        pass

class ConcreteClass1(AbstractClass):
    def required_operations1(self):
        print("ConcreteClass1 says: Implemented Operation1")

    def required_operations2(self):
        print("ConcreteClass1 says: Implemented Operation2")
```

```
class ConcreteClass2(AbstractClass):
    def required_operations1(self):
        print("ConcreteClass2 says: Implemented Operation1")

    def required_operations2(self):
        print("ConcreteClass2 says: Implemented Operation2")

    def hook1(self):
        print("ConcreteClass2 says: Overridden Hook1")

# Example usage
if __name__ == "__main__":
    print("Same client code can work with different subclasses:")
    concrete_class1 = ConcreteClass1()
    concrete_class1.template_method()

    print("\nSame client code can work with different subclasses:")
    concrete_class2 = ConcreteClass2()
    concrete_class2.template_method()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/template/test_template.py

```
import unittest
from unittest.mock import patch
from template import ConcreteClass1, ConcreteClass2

class TestConcreteClass1(unittest.TestCase):
    @patch('sys.stdout')
    def test_required_operations1(self, mock_stdout):
        concrete_class1 = ConcreteClass1()
        concrete_class1.required_operations1()
        mock_stdout.write.assert_called_with("ConcreteClass1 says: Implemented Operation1\n")

    @patch('sys.stdout')
    def test_required_operations2(self, mock_stdout):
        concrete_class1 = ConcreteClass1()
        concrete_class1.required_operations2()
        mock_stdout.write.assert_called_with("ConcreteClass1 says: Implemented Operation2\n")

class TestConcreteClass2(unittest.TestCase):
    @patch('sys.stdout')
    def test_required_operations1(self, mock_stdout):
        concrete_class2 = ConcreteClass2()
        concrete_class2.required_operations1()
        mock_stdout.write.assert_called_with("ConcreteClass2 says: Implemented Operation1\n")

    @patch('sys.stdout')
    def test_required_operations2(self, mock_stdout):
        concrete_class2 = ConcreteClass2()
        concrete_class2.required_operations2()
        mock_stdout.write.assert_called_with("ConcreteClass2 says: Implemented Operation2\n")

    @patch('sys.stdout')
    def test_hook1(self, mock_stdout):
        concrete_class2 = ConcreteClass2()
        concrete_class2.hook1()
        mock_stdout.write.assert_called_with("ConcreteClass2 says: Overridden Hook1\n")

def main():
    unittest.main()

if __name__ == '__main__':
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/facade/facade.py

```
class AIProcessingSubsystem:
    """
    A subsystem that might perform AI-related tasks.
    """
    def initialize(self) -> str:
        return "AIProcessingSubsystem: Initialized and ready to process."

    def process_data(self, data) -> str:
        return f"AIProcessingSubsystem: Processing data - {data}"

class DataAnalysisSubsystem:
    """
    A subsystem for data analysis.
    """
    def analyze(self, data) -> str:
        return f"DataAnalysisSubsystem: Analyzing data - {data}"

class NovaSystemFacade:
    """
    The Facade class provides a simple interface to the complex logic of NovaSystem's subsystems.
    """
    def __init__(self) -> None:
        self._ai_processor = AIProcessingSubsystem()
        self._data_analyzer = DataAnalysisSubsystem()

    def process_and_analyze_data(self, data) -> str:
        results = []
        results.append(self._ai_processor.initialize())
        results.append(self._ai_processor.process_data(data))
        results.append(self._data_analyzer.analyze(data))
        return "\n".join(results)

# Client Code
def client_code(facade: NovaSystemFacade) -> None:
    print(facade.process_and_analyze_data("Sample Data"), end="")

# Example usage
if __name__ == "__main__":
    facade = NovaSystemFacade()
    client_code(facade)
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/facade/__init__.py

```
from .facade import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/facade/test_facade.py

```
from facade import NovaSystemFacade, client_code

def test_novasystem_facade():
    # Creating the Facade instance
    novasystem_facade = NovaSystemFacade()

    # Simulating client interaction with the facade
    print("Testing NovaSystem Facade:")
    client_code(novasystem_facade)

def main():
    test_novasystem_facade()

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/observer/observer.py

```
from abc import ABC, abstractmethod

class Observer(ABC):
    def __init__(self):
        self.is_active = True

    @abstractmethod
    def update(self, subject) -> None:
        pass

    def activate(self):
        self.is_active = True

    def deactivate(self):
        self.is_active = False
    def is_observer_active(self) -> bool:
        return self.is_active

    def handle_error(self, error: Exception):
        print(f"Observer error: {error}")

    def pre_update(self):
        pass

    def post_update(self):
        pass


# Example of a concrete observer class with expanded functionality
class AdvancedObserver(Observer):
    def __init__(self):
        super().__init__()

    def update(self, subject) -> None:
        if not self.is_active:
            return

        if subject is None:
            raise ValueError("Subject cannot be None")  # Directly raise the exception

        try:
            self.pre_update()
            # Ensure 'subject' has attribute 'state' before trying to access it
            state = getattr(subject, 'state', 'No state')  # Default value if 'state' is not present
            print(f"AdvancedObserver updated with new state: {state}")
            self.post_update()
        except Exception as e:
            self.handle_error(e)

    def pre_update(self):
        print("Preparing to update AdvancedObserver.")

    def post_update(self):
        print("AdvancedObserver update complete.")

    def handle_error(self, error: Exception):
        print(f"Error in AdvancedObserver: {error}")
        # Optionally, you can re-raise the exception if needed for tests
        raise error
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/observer/test_observer.py

```
# # DesignPatterns/observer/test_observer.py
```

```
import pytest
from io import StringIO
from unittest.mock import patch
from observer import AdvancedObserver

class MockSubject:
    """ A mock subject class for testing the observer. """
    def __init__(self):
        self.state = None

    def change_state(self, new_state):
        self.state = new_state

@pytest.fixture
def observer():
    """ Fixture to create an AdvancedObserver instance. """
    return AdvancedObserver()

@pytest.fixture
def mock_subject():
    """ Fixture to create a MockSubject instance. """
    return MockSubject()

def test_activation(observer):
    """ Test if the observer activates and deactivates correctly. """
    observer.deactivate()
    assert not observer.is_observer_active()

    observer.activate()
    assert observer.is_observer_active()

def test_update_when_active(observer, mock_subject):
    """ Test if the observer updates its state when active. """
    with patch('sys.stdout', new_callable=StringIO) as mock_stdout:
        observer.activate()
        mock_subject.change_state("new_state")
        observer.update(mock_subject)
        assert "AdvancedObserver updated with new state: new_state" in mock_stdout.getvalue()

def test_no_update_when_inactive(observer, mock_subject):
    """ Test if the observer does not update its state when inactive. """
    with patch('sys.stdout', new_callable=StringIO) as mock_stdout:
        observer.deactivate()
        mock_subject.change_state("new_state")
        observer.update(mock_subject)
        assert mock_stdout.getvalue() == ""
```

```
def test_error_handling(observer):
    """ Test the error handling in the observer. """
    with patch('sys.stdout', new_callable=StringIO) as mock_stdout:
        observer.activate()
        with pytest.raises(ValueError) as exc_info:
            observer.update(None)

        assert "Subject cannot be None" == str(exc_info.value)

def test_pre_update_hook(observer, mock_subject):
    """ Test the execution of the pre-update hook. """
    with patch('sys.stdout', new_callable=StringIO) as mock_stdout:
        observer.activate()
        observer.update(mock_subject)
        assert "Preparing to update AdvancedObserver." in mock_stdout.getvalue()

def test_post_update_hook(observer, mock_subject):
    """ Test the execution of the post-update hook. """
    with patch('sys.stdout', new_callable=StringIO) as mock_stdout:
        observer.activate()
        observer.update(mock_subject)
        assert "AdvancedObserver update complete." in mock_stdout.getvalue()


# import unittest
# from unittest.mock import patch
# from observer import Observer, AdvancedObserver

# class MockSubject:
#     """
#     A mock subject class to simulate state changes for testing observers.
#     """
#     def __init__(self):
#         self.state = None

#     def change_state(self, new_state):
#         self.state = new_state

# class TestObserver(unittest.TestCase):
#     """
#     Test suite for the Observer class and its functionalities.
#     """

#     def setUp(self):
#         self.subject = MockSubject()
#         self.observer = AdvancedObserver()
#     def test_activation(self):
#         """ Test if the observer correctly activates and deactivates. """
#         self.observer.deactivate()
#         self.assertFalse(self.observer.is_observer_active())

#         self.observer.activate()
#         self.assertTrue(self.observer.is_observer_active())

#     def test_update_when_active(self):
#         """ Test if the observer updates its state when active. """
#         with patch('sys.stdout') as mock_stdout:
#             self.subject.change_state("new_state")
#             self.observer.activate()
#             self.observer.update(self.subject)
#             self.assertIn("AdvancedObserver updated with new state: new_state", mock_stdout.getvalue())

#     def test_no_update_when_inactive(self):
#         """ Test if the observer does not update its state when inactive. """
#         with patch('sys.stdout') as mock_stdout:
#             self.subject.change_state("new_state")
#             self.observer.deactivate()
#             self.observer.update(self.subject)
#             self.assertEqual(mock_stdout.getvalue(), "")

#     def test_error_handling(self):
#         with patch('sys.stdout', new_callable=unittest.mock.StringIO) as mock_stdout:
#             self.observer.activate()
#             with self.assertRaises(ValueError) as context:
#                 self.observer.update(None)  # Passing None should trigger an error in the observer
#             self.assertEqual(str(context.exception), "Subject cannot be None")

#     def test_pre_update_hook(self):
#         """ Test the execution of the pre-update hook. """
#         with patch('sys.stdout') as mock_stdout:
#             self.observer.activate()
#             self.observer.update(self.subject)
#             self.assertIn("Preparing to update AdvancedObserver.", mock_stdout.getvalue())

#     def test_post_update_hook(self):
#         """ Test the execution of the post-update hook. """
#         with patch('sys.stdout') as mock_stdout:
#             self.observer.activate()
#             self.observer.update(self.subject)
#             self.assertIn("AdvancedObserver update complete.", mock_stdout.getvalue())

# def main():
```

```
# unittest.main()

# if __name__ == '__main__':
#     main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/observer/__init__.py

```
from .observer import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/memento/memento.py

```
# memento.py
from datetime import datetime
from typing import List

class Memento:
    """
    The Memento interface provides a way to retrieve the memento's metadata,
    such as creation date or name. It doesn't expose the Originator's state.
    """
    def get_name(self) -> str:
        pass

    def get_date(self) -> str:
        pass

class ConcreteMemento(Memento):
    def __init__(self, state: str) -> None:
        self._state = state
        self._date = str(datetime.now())[:19]

    def get_state(self) -> str:
        return self._state

    def get_name(self) -> str:
        return f"{self._date} / ({self._state[0:9]}...)"

    def get_date(self) -> str:
        return self._date

class Originator:
    """
    The Originator holds an important state that can change over time.
    It defines methods for saving and restoring the state from a Memento.
    """
    _state = None

    def __init__(self, state: str) -> None:
        self._state = state
        print(f"Originator: My initial state is: {self._state}")
    def do_something(self) -> None:
        print("Originator: I'm doing something important.")
        self._state = f"state_{datetime.now().timestamp()}"
        print(f"Originator: and my state has changed to: {self._state}")

    def save(self) -> Memento:
        return ConcreteMemento(self._state)

    def restore(self, memento: Memento) -> None:
        self._state = memento.get_state()
        print(f"Originator: My state has changed to: {self._state}")

class Caretaker:
    """
    The Caretaker works with Mementos via the base Memento interface.
    It can store and restore the Originator's state.
    """
    def __init__(self, originator: Originator) -> None:
        self._mementos = []
        self._originator = originator

    def backup(self) -> None:
        print("\nCaretaker: Saving Originator's state...")
        self._mementos.append(self._originator.save())

    def undo(self) -> None:
        if not self._mementos:
            return

        memento = self._mementos.pop()
        print(f"Caretaker: Restoring state to: {memento.get_name()}")
        self._originator.restore(memento)

    def show_history(self) -> None:
        print("Caretaker: Here's the list of mementos:")
        for memento in self._mementos:
            print(memento.get_name())

# Example usage
if __name__ == "__main__":
    originator = Originator("Initial State")
    caretaker = Caretaker(originator)

    caretaker.backup()
    originator.do_something()
    caretaker.backup()
    originator.do_something()

    caretaker.backup()
    originator.do_something()

    caretaker.show_history()

    print("\nClient: Now, let's rollback!\n")
    caretaker.undo()

    print("\nClient: Once more!\n")
    caretaker.undo()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/memento/test_memento.p

```
import unittest
from unittest.mock import patch
from memento import Memento, ConcreteMemento, Originator, Caretaker

class TestMementoPattern(unittest.TestCase):
    def setUp(self):
        self.originator = Originator("Initial State")
        self.caretaker = Caretaker(self.originator)

    def test_memento_creation(self):
        """Test the creation of a memento and its properties."""
        memento = self.originator.save()
        self.assertIsInstance(memento, ConcreteMemento)
        self.assertTrue(memento.get_name().startswith("20"))  # Assuming current year
        self.assertTrue(memento.get_date().startswith("20"))  # Assuming current year

    def test_state_restoration(self):
        """Test the restoration of the state in the originator from a memento."""
        self.originator._state = "New State"
        memento = self.originator.save()
        self.originator._state = "Another State"
        self.originator.restore(memento)
        self.assertEqual(self.originator._state, "New State")

    def test_caretaker_memento_management(self):
        """Test the caretaker's ability to store and retrieve mementos."""
        self.caretaker.backup()
        self.caretaker.backup()
        self.assertEqual(len(self.caretaker._mementos), 2)

    def test_caretaker_undo_functionality(self):
        """Test the caretaker's undo functionality."""
        self.originator._state = "State A"
        self.caretaker.backup()
        self.originator._state = "State B"
        self.caretaker.backup()
        self.caretaker.undo()
        self.assertEqual(self.originator._state, "State A")

def main():
    unittest.main()

if __name__ == '__main__':
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/memento/__init__.py

```
from .memento import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/adapter/adapter.py

```
# adapter.py

class Target:
    """
    The Target defines the domain-specific interface used by the client code.
    """
    def request(self) -> str:
        return "Target: The default target's behavior."


class Adaptee:
    """
    The Adaptee contains some useful behavior, but its interface is incompatible
    with the existing client code. The Adaptee needs some adaptation before the
    client code can use it.
    """
    def specific_request(self) -> str:
        return ".eetpadA eht fo roivaheb laicepS"


# Inheritance-based Adapter
class AdapterInheritance(Target, Adaptee):
    """
    The Adapter makes the Adaptee's interface compatible with the Target's
    interface via multiple inheritance.
    """
    def request(self) -> str:
        return f"Adapter (Inheritance): (TRANSLATED) {self.specific_request()[::-1]}"


# Composition-based Adapter
```

```
class AdapterComposition(Target):
    """
    The Adapter makes the Adaptee's interface compatible with the Target's
    interface via composition.
    """
    def __init__(self, adaptee: Adaptee):
        self.adaptee = adaptee

    def request(self) -> str:
        return f"Adapter (Composition): (TRANSLATED) {self.adaptee.specific_request()[::-1]}"


def client_code(target: Target):
    """
    The client code supports all classes that follow the Target interface.
    """
    print(target.request(), end="\n\n")


if __name__ == "__main__":
    print("Client: I can work just fine with the Target objects:")
    target = Target()
    client_code(target)

    adaptee = Adaptee()
    print("Client: The Adaptee class has a weird interface. See, I don't understand it:")
    print(f"Adaptee: {adaptee.specific_request()}", end="\n\n")

    print("Client: But I can work with it via the Inheritance-based Adapter:")
    adapter_inheritance = AdapterInheritance()
    client_code(adapter_inheritance)

    print("Client: And also with the Composition-based Adapter:")
    adapter_composition = AdapterComposition(adaptee)
    client_code(adapter_composition)
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/adapter/__init__.py

```
from .adapter import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/adapter/test_adapter.py

```
# test_adapter.py

from adapter import Target, Adaptee, AdapterInheritance, AdapterComposition

def test_adapter_inheritance():
    """
    Test the inheritance-based Adapter.
    """
    adaptee = Adaptee()
    adapter = AdapterInheritance()

    assert adapter.request() == f"Adapter (Inheritance): (TRANSLATED) {adaptee.specific_request()[::-1]}", \
        "AdapterInheritance does not correctly adapt Adaptee"

    print("PASS: Inheritance-based Adapter test")

def test_adapter_composition():
    """
    Test the composition-based Adapter.
    """
    adaptee = Adaptee()
    adapter = AdapterComposition(adaptee)

    assert adapter.request() == f"Adapter (Composition): (TRANSLATED) {adaptee.specific_request()[::-1]}", 
        "AdapterComposition does not correctly adapt Adaptee"

    print("PASS: Composition-based Adapter test")

def main():
    """
    Main function to run the adapter tests.
    """
    print("Testing Adapter Pattern Implementations:")
    test_adapter_inheritance()
    test_adapter_composition()

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/prototype/__init__.py

```
from .prototype import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/prototype/test_prototype.p

```
import copy
from prototype import NovaComponent, SelfReferencingEntity

def test_shallow_copy(nova_component):
    shallow_copied_component = copy.copy(nova_component)
    print("Testing Shallow Copy:")

    # Modifying the shallow copy and testing its effect on the original
    shallow_copied_component.some_list_of_objects.append("new item")
    if "new item" in nova_component.some_list_of_objects:
        print("Shallow copy modification reflected in the original object.")
    else:
        print("Shallow copy modification not reflected in the original object.")
```

```
def test_deep_copy(nova_component):
    deep_copied_component = copy.deepcopy(nova_component)
    print("\nTesting Deep Copy:")

    # Modifying the deep copy and testing its effect on the original
    deep_copied_component.some_list_of_objects.append("new deep item")
    if "new deep item" in nova_component.some_list_of_objects:
        print("Deep copy modification reflected in the original object.")
    else:
        print("Deep copy modification not reflected in the original object.")

def main():
    list_of_objects = [1, {1, 2, 3}, [1, 2, 3]]
    circular_ref = SelfReferencingEntity()
    nova_component = NovaComponent(23, list_of_objects, circular_ref)
    circular_ref.set_parent(nova_component)

    test_shallow_copy(nova_component)
    test_deep_copy(nova_component)

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/prototype/prototype.py

```
import copy

class SelfReferencingEntity:
    def __init__(self):
        self.parent = None

    def set_parent(self, parent):
        self.parent = parent

class NovaComponent:
    def __init__(self, some_int, some_list_of_objects, some_circular_ref):
        self.some_int = some_int
        self.some_list_of_objects = some_list_of_objects
        self.some_circular_ref = some_circular_ref

    def __copy__(self):
        some_list_of_objects = copy.copy(self.some_list_of_objects)
        some_circular_ref = copy.copy(self.some_circular_ref)

        new = self.__class__(
            self.some_int, some_list_of_objects, some_circular_ref
        )
        new.__dict__.update(self.__dict__)

        return new

    def __deepcopy__(self, memo=None):
        if memo is None:
            memo = {}

        some_list_of_objects = copy.deepcopy(self.some_list_of_objects, memo)
        some_circular_ref = copy.deepcopy(self.some_circular_ref, memo)

        new = self.__class__(
            self.some_int, some_list_of_objects, some_circular_ref
        )
        new.__dict__ = copy.deepcopy(self.__dict__, memo)

        return new

# Example usage
if __name__ == "__main__":
    list_of_objects = [1, {1, 2, 3}, [1, 2, 3]]
    circular_ref = SelfReferencingEntity()
    nova_component = NovaComponent(23, list_of_objects, circular_ref)
    circular_ref.set_parent(nova_component)

    shallow_copied_component = copy.copy(nova_component)
    deep_copied_component = copy.deepcopy(nova_component)

    # Test and demonstrate the differences between shallow and deep copy
    # ... (Similar to the provided example code)
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/state/__init__.py

```
from .state import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/state/test_state.py

```
import unittest
from state import Context, ConcreteStateA, ConcreteStateB, ConcreteStateC, ConcreteStateD, StateContex

class TestStatePattern(unittest.TestCase):
    def test_initial_state(self):
        """Test the initial state setup in the context."""
        context = Context(ConcreteStateA())
        self.assertIsInstance(context.state, ConcreteStateA)

    def test_state_transition(self):
        """Test state transitions based on different conditions."""
        context = Context(ConcreteStateA())
        context.set_condition(True)  # Should transition to ConcreteStateB
        context.request()
        self.assertIsInstance(context.state, ConcreteStateB)

        context.set_condition(False)  # Should transition to ConcreteStateA
        context.request()
        self.assertIsInstance(context.state, ConcreteStateA)

    def test_special_case_handling(self):
        """Test the handling of special cases."""
        context = Context(ConcreteStateC())
        context.set_special_case(True)  # Should transition to ConcreteStateD
        context.request()
        self.assertIsInstance(context.state, ConcreteStateD)

        context.set_special_case(False)  # Should transition to ConcreteStateA
        context.request()
        self.assertIsInstance(context.state, ConcreteStateA)

    # Optional: Add a test for exception handling if relevant

def main():
    unittest.main()

if __name__ == '__main__':
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/state/state.py

```
from abc import ABC, abstractmethod
from dataclasses import dataclass
from typing import Callable, Optional

@dataclass
class StateContext:
    condition: bool = False
    special_case: bool = False

class State(ABC):
    @abstractmethod
    def handle(self, context: StateContext) -> None:
        pass

class ConcreteStateA(State):
    def handle(self, context: StateContext) -> None:
        print("State A handling context.")
        next_state = ConcreteStateB() if context.condition else ConcreteStateC()
        context.change_state(next_state)
```

```
class ConcreteStateB(State):
    def handle(self, context: StateContext) -> None:
        print("State B handling context.")
        context.change_state(ConcreteStateA())

class ConcreteStateC(State):
    def handle(self, context: StateContext) -> None:
        print("State C handling context.")
        next_state = ConcreteStateD() if context.special_case else ConcreteStateA()
        context.change_state(next_state)

class ConcreteStateD(State):
    def handle(self, context: StateContext) -> None:
        print("State D handling context (Special Case).")
        context.change_state(ConcreteStateA())

class Context:
    def __init__(self, state: State):
        self.state = state
        self.context_data = StateContext()

    def change_state(self, state: State) -> None:
        self.state = state

    def request(self) -> None:
        try:
            self.state.handle(self.context_data)
        except Exception as e:
            print(f"Error occurred: {e}")

    def set_condition(self, condition: bool) -> None:
        self.context_data.condition = condition

    def set_special_case(self, special_case: bool) -> None:
        self.context_data.special_case = special_case

# Example usage
if __name__ == "__main__":
    context = Context(ConcreteStateA())
    context.request()
    context.set_condition(True)
    context.request()
    context.set_special_case(True)
    context.request()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/composite/test_composite

```
# test_composite.py
```

```
from composite import Leaf, Composite

def test_leaf_operation():
    """
    Test the operation of a leaf component.
    """
    leaf = Leaf()
    assert leaf.operation() == "Leaf", "Leaf operation did not return expected result."
    print("PASS: Leaf operation test")

def test_composite_single_child():
    """
    Test a composite with a single child.
    """
    leaf = Leaf()
    composite = Composite()
    composite.add(leaf)

    assert composite.operation() == "Branch(Leaf)", "Composite operation with one child did not return expect
    print("PASS: Composite single child test")

def test_composite_multiple_children():
    """
    Test a composite with multiple children.
    """
    composite = Composite()
    composite.add(Leaf())
    composite.add(Leaf())

    assert composite.operation() == "Branch(Leaf+Leaf)", "Composite operation with multiple children did not
    print("PASS: Composite multiple children test")

def main():
    """
    Main function to run the composite pattern tests.
    """
    print("Testing Composite Pattern:")
    test_leaf_operation()
    test_composite_single_child()
    test_composite_multiple_children()

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/composite/__init__.py

```
from .composite import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/composite/composite.py

```python
# composite.py

from __future__ import annotations
from abc import ABC, abstractmethod
from typing import List


class Component(ABC):
    """
    The base Component class declares common operations for both simple and
    complex objects of a composition.
    """

    def __init__(self) -> None:
        self._parent: Component = None

    @property
    def parent(self) -> Component:
        return self._parent

    @parent.setter
    def parent(self, parent: Component):
        self._parent = parent

    def add(self, component: Component) -> None:
        pass

    def remove(self, component: Component) -> None:
        pass

    def is_composite(self) -> bool:
        return False

    @abstractmethod
    def operation(self) -> str:
        pass


class Leaf(Component):
    """
    The Leaf class represents the end objects of a composition. A leaf can't
    have any children. Usually, it's the Leaf objects that do the actual work.
    """

    def operation(self) -> str:
        return "Leaf"
```

```
class Composite(Component):
    """
    The Composite class represents complex components that may have children.
    It delegates the actual work to their children and then 'sum-up' the result.
    """

    def __init__(self) -> None:
        super().__init__()
        self._children: List[Component] = []

    def add(self, component: Component) -> None:
        self._children.append(component)
        component.parent = self

    def remove(self, component: Component) -> None:
        self._children.remove(component)
        component.parent = None

    def is_composite(self) -> bool:
        return True

    def operation(self) -> str:
        results = [child.operation() for child in self._children]
        return f"Branch({'+'.join(results)})"


def client_code(component: Component) -> None:
    print(f"RESULT: {component.operation()}", end="")


def client_code2(component1: Component, component2: Component) -> None:
    if component1.is_composite():
        component1.add(component2)
    print(f"RESULT: {component1.operation()}", end="")


if __name__ == "__main__":
    simple = Leaf()
    print("Client: I've got a simple component:")
    client_code(simple)
    print("\n")

    tree = Composite()

    branch1 = Composite()
    branch1.add(Leaf())
    branch1.add(Leaf())

    branch2 = Composite()
    branch2.add(Leaf())

    tree.add(branch1)
    tree.add(branch2)

    print("Client: Now I've got a composite tree:")
    client_code(tree)
    print("\n")

    print("Client: I don't need to check the components classes even when managing the tree:")
    client_code2(tree, simple)
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/memoize/memoize.py

```
import functools
import time

def memoize(max_size=100, timeout=None):
    def memoize_decorator(func):
        cache = {}
        timestamps = {}
        @functools.wraps(func)
        def wrapper(*args, **kwargs):
            key = (args, frozenset(kwargs.items()))

            # Check for expired cache entries
            if timeout:
                for k in list(timestamps.keys()):
                    if time.time() - timestamps[k] > timeout:
                        del cache[k]
                        del timestamps[k]

            if key in cache:
                return cache[key]

            # If cache size limit is reached, remove the oldest item
            if len(cache) >= max_size:
                oldest_key = min(timestamps, key=timestamps.get)
                del cache[oldest_key]
                del timestamps[oldest_key]

            result = func(*args, **kwargs)
            cache[key] = result
            timestamps[key] = time.time()
            return result
        return wrapper
    return memoize_decorator

# Example usage
@memoize(max_size=50, timeout=300)  # 50 items in cache and 5 minutes timeout
def some_function(arg1, arg2, **kwargs):
    # Your function implementation
    return arg1 + arg2  # Replace with actual computation

print(some_function(3, 4, option='value'))
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/memoize/test_memoize.p

```
import unittest
from unittest.mock import patch
from memoize import memoize
import time

class TestMemoizeDecorator(unittest.TestCase):
    def setUp(self):
        @memoize(max_size=2, timeout=1)
        def test_func(a, b):
            return a + b
        self.test_func = test_func

    def test_basic_memoization(self):
        """Test basic memoization functionality."""
        result1 = self.test_func(1, 2)
        result2 = self.test_func(1, 2)
        self.assertEqual(result1, result2)

    def test_cache_size_limit(self):
        """Test cache size limit handling."""
        self.test_func(1, 2)
        self.test_func(3, 4)
        self.test_func(5, 6)  # This should remove the oldest cache (1, 2)
        with patch('time.time', return_value=time.time() + 2):
            result = self.test_func(1, 2)  # Recalculate as it should be removed from cache
            self.assertEqual(result, 3)

    def test_timeout_handling(self):
        """Test timeout handling in the cache."""
        self.test_func(7, 8)
        with patch('time.time', return_value=time.time() + 2):
            result = self.test_func(7, 8)  # Recalculate as it should be expired
            self.assertEqual(result, 15)
```

```
def main():
    unittest.main()

if __name__ == '__main__':
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/memoize/__init__.py

```
from .memoize import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/iterator/__init__.py

```
from .iterator import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/iterator/iterator.py

```
from collections.abc import Iterable, Iterator
from typing import Any, List

class AlphabeticalOrderIterator(Iterator):
    """
    Concrete Iterators implement various traversal algorithms.
    """
    _position: int = None
    _reverse: bool = False

    def __init__(self, collection: List[Any], reverse: bool = False) -> None:
        self._collection = collection
        self._reverse = reverse
        self._position = -1 if reverse else 0

    def __next__(self):
        try:
            value = self._collection[self._position]
            self._position += -1 if self._reverse else 1
        except IndexError:
            raise StopIteration()
        return value

class WordsCollection(Iterable):
    """
    Concrete Collections provide methods for retrieving fresh iterator instances.
    """
    def __init__(self, collection: List[Any] = []) -> None:
        self._collection = collection

    def __iter__(self) -> AlphabeticalOrderIterator:
        return AlphabeticalOrderIterator(self._collection)

    def get_reverse_iterator(self) -> AlphabeticalOrderIterator:
        return AlphabeticalOrderIterator(self._collection, True)
    def add_item(self, item: Any):
        self._collection.append(item)

# Example usage
if __name__ == "__main__":
    collection = WordsCollection()
    collection.add_item("First")
    collection.add_item("Second")
    collection.add_item("Third")

    print("Straight traversal:")
    for item in collection:
        print(item)

    print("\nReverse traversal:")
    for item in collection.get_reverse_iterator():
        print(item)
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/iterator/test_iterator.py

```
from collections.abc import Iterable, Iterator
from typing import Any, List

class AlphabeticalOrderIterator(Iterator):
    """
    Concrete Iterators implement various traversal algorithms.
    """
    _position: int = None
    _reverse: bool = False

    def __init__(self, collection: List[Any], reverse: bool = False) -> None:
        self._collection = collection
        self._reverse = reverse
        self._position = -1 if reverse else 0

    def __next__(self):
        try:
            value = self._collection[self._position]
            self._position += -1 if self._reverse else 1
        except IndexError:
            raise StopIteration()
        return value

class WordsCollection(Iterable):
    """
    Concrete Collections provide methods for retrieving fresh iterator instances.
    """
    def __init__(self, collection: List[Any] = []) -> None:
        self._collection = collection

    def __iter__(self) -> AlphabeticalOrderIterator:
        return AlphabeticalOrderIterator(self._collection)

    def get_reverse_iterator(self) -> AlphabeticalOrderIterator:
        return AlphabeticalOrderIterator(self._collection, True)

    def add_item(self, item: Any):
        self._collection.append(item)

def main():
    collection = WordsCollection()
    collection.add_item("First")
    collection.add_item("Second")
    collection.add_item("Third")

    print("Straight traversal:")
    for item in collection:
        print(item)

    print("\nReverse traversal:")
    for item in collection.get_reverse_iterator():
        print(item)

# Example usage
if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/chain_of_responsibility/__

```
from .chain_of_responsibility import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/chain_of_responsibility/tes

```
from chain_of_responsibility import AIModelHandler, DataPreprocessingHandler, VisualizationHandler, Abst

def test_individual_handler(handler: AbstractHandler, request: str):
    result = handler.handle(request)
    if result:
        print(f"  Handled by {handler.__class__.__name__}: {result}")
    else:
        print(f"  {handler.__class__.__name__} passed the request.")

def test_full_chain(chain_head: AbstractHandler, request: str):
    print(f"\nTesting full chain with request: {request}")
    result = chain_head.handle(request)
    if result:
        print(f"Handled by chain: {result}")
    else:
        print("Request was left unhandled by the full chain.")

def main():
    # Setting up individual handlers
    ai_model_handler = AIModelHandler()
    data_handler = DataPreprocessingHandler()
    visualization_handler = VisualizationHandler()

    # Building the chain
    ai_model_handler.set_next(data_handler).set_next(visualization_handler)

    # Testing individual handlers
    test_requests = ["Train", "Preprocess", "Visualize", "Deploy", "Unknown"]
    for request in test_requests:
        print(f"\nTesting AIModelHandler with request: {request}")
        test_individual_handler(ai_model_handler, request)
        print(f"\nTesting DataPreprocessingHandler with request: {request}")
        test_individual_handler(data_handler, request)
        print(f"\nTesting VisualizationHandler with request: {request}")
        test_individual_handler(visualization_handler, request)

    # Testing the full chain
    for request in test_requests:
        test_full_chain(ai_model_handler, request)

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/chain_of_responsibility/ch

```
from abc import ABC, abstractmethod
from typing import Any, Optional

class Handler(ABC):
    """
    The Handler interface declares methods for building the chain of handlers and executing requests.
    """
    @abstractmethod
    def set_next(self, handler: 'Handler') -> 'Handler':
        pass

    @abstractmethod
    def handle(self, request: Any) -> Optional[str]:
        pass

class AbstractHandler(Handler):
    """
    Default chaining behavior implementation.
    """
    _next_handler: Handler = None

    def set_next(self, handler: 'Handler') -> 'Handler':
        self._next_handler = handler
        return handler

    def handle(self, request: Any) -> Optional[str]:
        if self._next_handler:
            return self._next_handler.handle(request)
        return None

# Concrete Handlers
class AIModelHandler(AbstractHandler):
    def handle(self, request: Any) -> str:
        if request == "Train":
            return f"AIModelHandler: Training model with {request}"
        else:
            return super().handle(request)

class DataPreprocessingHandler(AbstractHandler):
    def handle(self, request: Any) -> str:
        if request == "Preprocess":
            return f"DataPreprocessingHandler: Preprocessing {request}"
        else:
            return super().handle(request)

class VisualizationHandler(AbstractHandler):
    def handle(self, request: Any) -> str:
        if request == "Visualize":
            return f"VisualizationHandler: Visualizing data with {request}"
        else:
            return super().handle(request)

# Client code example
def client_code(handler: Handler) -> None:
    for operation in ["Train", "Preprocess", "Visualize", "Deploy"]:
        print(f"\nClient: Requesting to {operation}")
        result = handler.handle(operation)
        if result:
            print(f"  {result}", end="")
        else:
            print(f"  {operation} was left unhandled.", end="")

# Example usage
if __name__ == "__main__":
    ai_model_handler = AIModelHandler()
    data_handler = DataPreprocessingHandler()
    visualization_handler = VisualizationHandler()

    ai_model_handler.set_next(data_handler).set_next(visualization_handler)

    print("Chain: AI Model > Data Preprocessing > Visualization")
    client_code(ai_model_handler)
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/visitor/__init__.py

```
from .visitor import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/visitor/visitor.py

```
from abc import ABC, abstractmethod

class Visitor(ABC):
    """
    The Visitor interface declares a set of visiting methods that correspond to
    element classes. The signature of a visiting method allows the visitor to
    identify the exact class of the element being visited.
    """
    @abstractmethod
    def visit_concrete_element_a(self, element):
        pass

    @abstractmethod
    def visit_concrete_element_b(self, element):
        pass

class ConcreteVisitor1(Visitor):
    def visit_concrete_element_a(self, element):
        print(f"{element.operation_a()} + ConcreteVisitor1")

    def visit_concrete_element_b(self, element):
        print(f"{element.operation_b()} + ConcreteVisitor1")

class ConcreteVisitor2(Visitor):
    def visit_concrete_element_a(self, element):
        print(f"{element.operation_a()} + ConcreteVisitor2")

    def visit_concrete_element_b(self, element):
        print(f"{element.operation_b()} + ConcreteVisitor2")

class Element(ABC):
    """
    The Element interface declares an `accept` method that should take a base
    visitor interface as an argument.
    """
    @abstractmethod
    def accept(self, visitor: Visitor):
        pass

class ConcreteElementA(Element):
    def accept(self, visitor: Visitor):
        visitor.visit_concrete_element_a(self)

    def operation_a(self):
        return "ConcreteElementA"

class ConcreteElementB(Element):
    def accept(self, visitor: Visitor):
        visitor.visit_concrete_element_b(self)

    def operation_b(self):
        return "ConcreteElementB"

# Example usage
if __name__ == "__main__":
    elements = [ConcreteElementA(), ConcreteElementB()]

    visitor1 = ConcreteVisitor1()
    for element in elements:
        element.accept(visitor1)

    visitor2 = ConcreteVisitor2()
    for element in elements:
        element.accept(visitor2)
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/visitor/test_visitor.py

```
import unittest
from unittest.mock import patch
from visitor import ConcreteVisitor1, ConcreteVisitor2, ConcreteElementA, ConcreteElementB

class TestConcreteVisitor1(unittest.TestCase):
    @patch('sys.stdout')
    def test_visit_concrete_element_a(self, mock_stdout):
        element_a = ConcreteElementA()
        visitor1 = ConcreteVisitor1()
        element_a.accept(visitor1)
        mock_stdout.write.assert_called_with("ConcreteElementA + ConcreteVisitor1\n")

    @patch('sys.stdout')
    def test_visit_concrete_element_b(self, mock_stdout):
        element_b = ConcreteElementB()
        visitor1 = ConcreteVisitor1()
        element_b.accept(visitor1)
        mock_stdout.write.assert_called_with("ConcreteElementB + ConcreteVisitor1\n")

class TestConcreteVisitor2(unittest.TestCase):
    @patch('sys.stdout')
    def test_visit_concrete_element_a(self, mock_stdout):
        element_a = ConcreteElementA()
        visitor2 = ConcreteVisitor2()
        element_a.accept(visitor2)
        mock_stdout.write.assert_called_with("ConcreteElementA + ConcreteVisitor2\n")

    @patch('sys.stdout')
    def test_visit_concrete_element_b(self, mock_stdout):
        element_b = ConcreteElementB()
        visitor2 = ConcreteVisitor2()
        element_b.accept(visitor2)
        mock_stdout.write.assert_called_with("ConcreteElementB + ConcreteVisitor2\n")

def main():
    unittest.main()

if __name__ == '__main__':
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/singleton/test_singleton.py

```
# test_singleton.py

from singleton import AIModelManager

def test_singleton_instance_creation():
    """
    Test that the Singleton instance is created only once.
    """
    print("Testing Singleton instance creation...")
    first_instance = AIModelManager()
    second_instance = AIModelManager()

    assert first_instance is second_instance, "Singleton instances are not the same"
    print("PASS: Singleton instance creation test")

def test_singleton_configuration_persistence():
    """
    Test that changes in configuration are reflected across all instances.
    """
    print("Testing Singleton configuration persistence...")
    manager = AIModelManager()
    initial_config = manager.get_config("response_length")
    # Change configuration
    manager.update_config("response_length", 512)

    # Create new instance and check if the configuration change is reflected
    new_manager = AIModelManager()
    new_config = new_manager.get_config("response_length")

    assert new_config == 512, "Configuration change is not reflected in the new instance"
    assert initial_config != new_config, "Initial and new configurations are the same"
    print("PASS: Singleton configuration persistence test")


def main():
    test_singleton_instance_creation()
    test_singleton_configuration_persistence()

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/singleton/__init__.py

```
from .singleton import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/singleton/singleton.py

```
from threading import Lock

class SingletonMeta(type):
    """
    Thread-safe implementation of Singleton for managing AI model configurations.
    """
    _instances = {}
    _lock: Lock = Lock()

    def __call__(cls, *args, **kwargs):
        with cls._lock:
            if cls not in cls._instances:
                instance = super().__call__(*args, **kwargs)
                cls._instances[cls] = instance
        return cls._instances[cls]

class AIModelManager(metaclass=SingletonMeta):
    def __init__(self):
        # Initialize with default configuration
        self.config = {
            "language_model": "GPT-3",
            "response_length": 128,
            "custom_behavior": {}
        }
    def update_config(self, key, value):
        self.config[key] = value

    def get_config(self, key):
        return self.config.get(key, None)

    def perform_ai_logic(self):
        # Method to perform AI-related operations
        pass

# Example of direct usage:
# ai_manager = AIModelManager()
# ai_manager.update_config("response_length", 256)
# print(ai_manager.get_config("response_length"))
# ai_manager.perform_ai_logic()

# Example of indirect usage:
def main():
    # Creating the Singleton instance
    ai_manager = AIModelManager()

    # Initial configuration
    print("Initial Configuration:", ai_manager.config)

    # Updating configuration in one part of the system
    ai_manager.update_config("response_length", 256)
    print("Updated Configuration after first change:", ai_manager.config)

    # Accessing the Singleton in another part of the system
    another_manager_instance = AIModelManager()
    print("Configuration accessed from a different part:", another_manager_instance.config)

    # Demonstrating that the configuration change is reflected across all instances
    another_manager_instance.update_config("language_model", "Custom AI Model")
    print("Configuration after updating from another part:", ai_manager.config)

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/command/test_command.

```
from command import SimpleCommand, ComplexCommand, Receiver, Invoker

def test_simple_command():
    print("Testing SimpleCommand:")
    simple_command = SimpleCommand("Simple Operation")
    simple_command.execute()
```

```
def test_complex_command():
    print("\nTesting ComplexCommand:")
    receiver = Receiver()
    complex_command = ComplexCommand(receiver, "Data A", "Data B")
    complex_command.execute()

def test_invoker():
    print("\nTesting Invoker:")
    invoker = Invoker()
    receiver = Receiver()

    # Setting up SimpleCommand and ComplexCommand for the invoker
    invoker.set_on_start(SimpleCommand("Initialization"))
    invoker.set_on_finish(ComplexCommand(receiver, "Finalize Operation", "Clean Up"))

    invoker.do_something_important()

def main():
    test_simple_command()
    test_complex_command()
    test_invoker()

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/command/command.py

```
from abc import ABC, abstractmethod

class Command(ABC):
    """
    The Command interface declares a method for executing a command.
    """
    @abstractmethod
    def execute(self) -> None:
        pass

class SimpleCommand(Command):
    """
    Some commands can implement simple operations on their own.
    """
    def __init__(self, payload: str) -> None:
        self._payload = payload

    def execute(self) -> None:
        print(f"SimpleCommand: Doing something simple like printing ({self._payload})")
```

```
class ComplexCommand(Command):
    """
    Complex commands delegate operations to other objects, called 'receivers.'
    """
    def __init__(self, receiver: 'Receiver', a: str, b: str) -> None:
        self._receiver = receiver
        self._a = a
        self._b = b

    def execute(self) -> None:
        print("ComplexCommand: Delegating complex tasks to a receiver object")
        self._receiver.do_something(self._a)
        self._receiver.do_something_else(self._b)

class Receiver:
    """
    The Receiver class contains important business logic.
    """
    def do_something(self, a: str) -> None:
        print(f"Receiver: Working on ({a}).")

    def do_something_else(self, b: str) -> None:
        print(f"Receiver: Also working on ({b}).")

class Invoker:
    """
    The Invoker is associated with commands and sends requests to the command.
    """
    _on_start = None
    _on_finish = None

    def set_on_start(self, command: Command):
        self._on_start = command

    def set_on_finish(self, command: Command):
        self._on_finish = command

    def do_something_important(self) -> None:
        print("Invoker: Does anybody want something done before I begin?")
        if isinstance(self._on_start, Command):
            self._on_start.execute()

        print("\nInvoker: ...doing something really important...")

        print("\nInvoker: Does anybody want something done after I finish?")
        if isinstance(self._on_finish, Command):
            self._on_finish.execute()
```

```
# Example usage
if __name__ == "__main__":
    invoker = Invoker()
    invoker.set_on_start(SimpleCommand("Start operation"))
    receiver = Receiver()
    invoker.set_on_finish(ComplexCommand(receiver, "Send email", "Save report"))

    invoker.do_something_important()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/command/__init__.py

```
from .command import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/flyweight/flyweight.py

```
import json
from typing import Dict, List

class Flyweight:
    def __init__(self, shared_state: List[str]) -> None:
        self._shared_state = shared_state

    def operation(self, unique_state: List[str]) -> None:
        shared = json.dumps(self._shared_state)
        unique = json.dumps(unique_state)
        print(f"Flyweight: Shared ({shared}) and unique ({unique}) state.")

class FlyweightFactory:
    _flyweights: Dict[str, Flyweight] = {}

    def __init__(self, initial_flyweights: List[List[str]]) -> None:
        for state in initial_flyweights:
            self._flyweights[self.get_key(state)] = Flyweight(state)

    def get_key(self, state: List[str]) -> str:
        return "_".join(sorted(state))

    def get_flyweight(self, shared_state: List[str]) -> Flyweight:
        key = self.get_key(shared_state)
        if not self._flyweights.get(key):
            print("FlyweightFactory: Creating new flyweight.")
            self._flyweights[key] = Flyweight(shared_state)
        else:
            print("FlyweightFactory: Reusing existing flyweight.")
        return self._flyweights[key]

    def list_flyweights(self) -> None:
        print(f"FlyweightFactory: I have {len(self._flyweights)} flyweights:")
        for key in self._flyweights:
            print(key)

# Client code example
def add_ai_component_to_system(factory: FlyweightFactory, data: List[str]) -> None:
    flyweight = factory.get_flyweight(data[:-1])
    flyweight.operation(data)

# Example usage
if __name__ == "__main__":
    factory = FlyweightFactory([
        ["NeuralNet", "Classifier", "Image"],
        ["NeuralNet", "Regressor", "TimeSeries"]
    ])

    factory.list_flyweights()

    add_ai_component_to_system(factory, ["NeuralNet", "Classifier", "Image", "ImageSetA"])
    add_ai_component_to_system(factory, ["NeuralNet", "Classifier", "Audio", "AudioSetB"])

    factory.list_flyweights()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/flyweight/__init__.py

```
from .flyweight import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/flyweight/test_flyweight.py

```
from flyweight import FlyweightFactory, add_ai_component_to_system

def test_flyweight_pattern():
    # Creating a Flyweight Factory with some initial shared states
    factory = FlyweightFactory([
        ["NeuralNet", "Classifier", "Image"],
        ["NeuralNet", "Regressor", "TimeSeries"]
    ])

    # Listing initial flyweights
    print("Initial flyweights in the factory:")
    factory.list_flyweights()

    # Adding components and testing if flyweights are reused or newly created
    print("\nAdding a new AI component to the system:")
    add_ai_component_to_system(factory, ["NeuralNet", "Classifier", "Image", "DatasetX"])

    print("\nAdding another AI component to the system (should reuse flyweight):")
    add_ai_component_to_system(factory, ["NeuralNet", "Classifier", "Image", "DatasetY"])

    print("\nAdding a different AI component (should create new flyweight):")
    add_ai_component_to_system(factory, ["NeuralNet", "Classifier", "Audio", "DatasetZ"])
    # Listing final flyweights to verify the correct creation and reuse
    print("\nFinal flyweights in the factory:")
    factory.list_flyweights()

def main():
    test_flyweight_pattern()

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/builder/__init__.py

```
from .builder import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/builder/builder.py

```
from abc import ABC, abstractmethod

# Builder Interface
class AIComponentBuilder(ABC):

    @property
    @abstractmethod
    def product(self):
        """Property to get the product."""
        pass

    @abstractmethod
    def add_ai_module(self):
        """Method to add an AI module to the product."""
        pass

    @abstractmethod
    def add_learning_capability(self):
        """Method to add learning capability to the product."""
        pass

    @abstractmethod
    def add_interaction_interface(self):
        """Method to add an interaction interface to the product."""
        pass

# Concrete Builder
class ConcreteAIComponentBuilder(AIComponentBuilder):
    def __init__(self):
        self.reset()

    def reset(self):
        """Reset the builder to start with a fresh product."""
        self._product = AIComponent()
    @property
    def product(self):
        """Retrieve the built product and reset the builder."""
        product = self._product
        self.reset()
        return product

    def add_ai_module(self):
        """Add an AI module to the product."""
        self._product.add("AI Module")

    def add_learning_capability(self):
        """Add learning capability to the product."""
        self._product.add("Learning Capability")

    def add_interaction_interface(self):
        """Add an interaction interface to the product."""
        self._product.add("Interaction Interface")

# Product Class
class AIComponent:
    def __init__(self):
        self.parts = []

    def add(self, part):
        """Add a part to the component."""
        self.parts.append(part)

    def list_parts(self):
        """List all parts of the component."""
        print(f"AI Component Parts: {', '.join(self.parts)}", end="")

# Director Class
class Director:
    def __init__(self):
        self._builder = None

    @property
    def builder(self):
        """Property to get and set the builder."""
        return self._builder

    @builder.setter
    def builder(self, builder):
        self._builder = builder
    def build_minimal_ai_component(self):
        """Build a minimal AI component."""
        self.builder.add_ai_module()

    def build_full_featured_ai_component(self):
        """Build a full-featured AI component."""
        self.builder.add_ai_module()
        self.builder.add_learning_capability()
        self.builder.add_interaction_interface()

# Client Code (optional here, might be in a separate test file)
if __name__ == "__main__":
    director = Director()
    builder = ConcreteAIComponentBuilder()
    director.builder = builder

    print("Building a minimal AI component:")
    director.build_minimal_ai_component()
    builder.product.list_parts()

    print("\n\nBuilding a full-featured AI component:")
    director.build_full_featured_ai_component()
    builder.product.list_parts()

    print("\n\nBuilding a custom AI component:")
    builder.add_ai_module()
    builder.add_interaction_interface()
    builder.product.list_parts()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/builder/test_builder.py

```
from builder import Director, ConcreteAIComponentBuilder

def test_minimal_ai_component(director, builder):
    print("Testing minimal AI component construction:")
    director.builder = builder
    director.build_minimal_ai_component()
    builder.product.list_parts()

def test_full_featured_ai_component(director, builder):
    print("\nTesting full-featured AI component construction:")
    director.builder = builder
    director.build_full_featured_ai_component()
    builder.product.list_parts()

def test_custom_ai_component(builder):
    print("\nTesting custom AI component construction:")
    builder.add_ai_module()
    builder.add_learning_capability()  # Adding only specific parts
    builder.product.list_parts()

def main():
    director = Director()
    builder = ConcreteAIComponentBuilder()

    test_minimal_ai_component(director, builder)
    test_full_featured_ai_component(director, builder)
    test_custom_ai_component(builder)

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/mediator/test_mediator.py

```
import unittest
from unittest.mock import patch
from mediator import Mediator, ConcreteMediator, BaseComponent, Component1, Component2

class TestMediatorPattern(unittest.TestCase):
    def setUp(self):
        self.component1 = Component1()
        self.component2 = Component2()
        self.mediator = ConcreteMediator(self.component1, self.component2)

    def test_mediator_initialization(self):
        """Test if the mediator is correctly set in the components."""
        self.assertEqual(self.component1.mediator, self.mediator)
        self.assertEqual(self.component2.mediator, self.mediator)

    def test_component_communication(self):
        """Test the communication between components via the mediator."""
        with patch('sys.stdout') as mock_stdout:
            self.component1.do_a()
            self.assertIn("Component 1 does A.", mock_stdout.getvalue())
            self.assertIn("Mediator reacts on A and triggers:", mock_stdout.getvalue())
            self.assertIn("Component 2 does C.", mock_stdout.getvalue())

            mock_stdout.reset()
            self.component2.do_d()
            self.assertIn("Component 2 does D.", mock_stdout.getvalue())
            self.assertIn("Mediator reacts on D and triggers:", mock_stdout.getvalue())
            self.assertIn("Component 1 does B.", mock_stdout.getvalue())
            self.assertIn("Component 2 does C.", mock_stdout.getvalue())

    def test_mediator_reactions(self):
        """Test mediator's reactions to different events."""
        with patch('sys.stdout') as mock_stdout:
            self.mediator.notify(self.component1, "A")
            self.assertIn("Mediator reacts on A and triggers:", mock_stdout.getvalue())
            self.assertIn("Component 2 does C.", mock_stdout.getvalue())

            mock_stdout.reset()
            self.mediator.notify(self.component2, "D")
            self.assertIn("Mediator reacts on D and triggers:", mock_stdout.getvalue())
            self.assertIn("Component 1 does B.", mock_stdout.getvalue())
            self.assertIn("Component 2 does C.", mock_stdout.getvalue())

def main():
    unittest.main()

if __name__ == '__main__':
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/mediator/mediator.py

```
from abc import ABC, abstractmethod

class Mediator(ABC):
    """
    The Mediator interface declares a method for components to notify the mediator about events.
    """
    @abstractmethod
    def notify(self, sender: object, event: str) -> None:
        pass

class ConcreteMediator(Mediator):
    def __init__(self, component1: 'Component1', component2: 'Component2') -> None:
        self._component1 = component1
        self._component1.mediator = self
        self._component2 = component2
        self._component2.mediator = self

    def notify(self, sender: object, event: str) -> None:
        if event == "A":
            print("Mediator reacts on A and triggers:")
            self._component2.do_c()
        elif event == "D":
            print("Mediator reacts on D and triggers:")
            self._component1.do_b()
            self._component2.do_c()

class BaseComponent:
    """
    Base Component class with a mediator.
    """
    def __init__(self, mediator: Mediator = None) -> None:
        self._mediator = mediator

    @property
    def mediator(self) -> Mediator:
        return self._mediator

    @mediator.setter
    def mediator(self, mediator: Mediator) -> None:
        self._mediator = mediator

class Component1(BaseComponent):
    def do_a(self) -> None:
        print("Component 1 does A.")
        self.mediator.notify(self, "A")

    def do_b(self) -> None:
        print("Component 1 does B.")
        self.mediator.notify(self, "B")

class Component2(BaseComponent):
    def do_c(self) -> None:
        print("Component 2 does C.")
        self.mediator.notify(self, "C")

    def do_d(self) -> None:
        print("Component 2 does D.")
        self.mediator.notify(self, "D")

# Example usage
if __name__ == "__main__":
    c1 = Component1()
    c2 = Component2()
    mediator = ConcreteMediator(c1, c2)

    print("Client triggers operation A.")
    c1.do_a()

    print("\nClient triggers operation D.")
    c2.do_d()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/mediator/__init__.py

```
from .mediator import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/factory/test_factory.py

```
from factory import BotFactory, ChatBot, DataAnalysisBot, ConcurrentBot
```

```
def test_bot_creation(factory):
    # Testing the creation of ChatBot
    print("Testing ChatBot Creation:")
    chat_bot = factory.create_bot("chat")
    print(chat_bot.perform_task())

    # Testing the creation of DataAnalysisBot
    print("\nTesting DataAnalysisBot Creation:")
    data_bot = factory.create_bot("data")
    print(data_bot.perform_task())

def test_dynamic_bot_registration(factory):
    # Dynamically registering and testing a new bot type
    class ResearchBot:
        def perform_task(self):
            return "ResearchBot performing research."

    factory.register_new_bot_type("research", ResearchBot)
    research_bot = factory.create_bot("research")
    print("\nTesting Dynamically Registered ResearchBot:")
    print(research_bot.perform_task())

def main():
    bot_factory = BotFactory()
    bot_factory.register_new_bot_type("chat", ChatBot)
    bot_factory.register_new_bot_type("data", DataAnalysisBot)
    bot_factory.register_new_bot_type("concurrent", lambda: ConcurrentBot(ChatBot()))

    test_bot_creation(bot_factory)
    test_dynamic_bot_registration(bot_factory)

if __name__ == "__main__":
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/factory/__init__.py

```
from .factory import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/factory/factory.py

```
from abc import ABC, abstractmethod
import threading

# Step 1: Dynamic Bot Factory Interface
class Factory(ABC):
    @abstractmethod
    def create_bot(self, bot_type):
        pass

    @abstractmethod
    def register_new_bot_type(self, bot_type, bot_creator):
        pass

# Step 2: Polymorphic Concrete Bot Factories
class BotFactory(Factory):
    def __init__(self):
        self.bot_creators = {}

    def create_bot(self, bot_type):
        return self.bot_creators[bot_type]()

    def register_new_bot_type(self, bot_type, bot_creator):
        self.bot_creators[bot_type] = bot_creator

# Step 3: Abstract Bot Interface
class AbstractBot(ABC):
    @abstractmethod
    def perform_task(self):
        pass

    @abstractmethod
    def learn_new_skill(self, skill):
        pass

# Step 4: Various AI Bots
class ChatBot(AbstractBot):
    def perform_task(self):
        return "ChatBot engaging in conversation."

    def learn_new_skill(self, skill):
        return f"ChatBot learning {skill}."

class DataAnalysisBot(AbstractBot):
    def perform_task(self):
        return "DataAnalysisBot analyzing data."

    def learn_new_skill(self, skill):
        return f"DataAnalysisBot learning {skill}."

# Step 5: Concurrency in Bots
class ConcurrentBot(AbstractBot):
    def __init__(self, bot):
        self.bot = bot
        self.lock = threading.Lock()

    def perform_task(self):
        with self.lock:
            return self.bot.perform_task()

    def learn_new_skill(self, skill):
        with self.lock:
            return self.bot.learn_new_skill(skill)

# Step 6: Client Code Demonstration
def client_code(factory: BotFactory):
    factory.register_new_bot_type("chat", ChatBot)
    factory.register_new_bot_type("data", DataAnalysisBot)

    chat_bot = factory.create_bot("chat")
    print(chat_bot.perform_task())

    # Dynamically registering a new bot type
    factory.register_new_bot_type("concurrent", lambda: ConcurrentBot(ChatBot()))
    concurrent_bot = factory.create_bot("concurrent")
    print(concurrent_bot.perform_task())

# Demonstration
if __name__ == "__main__":
    bot_factory = BotFactory()
    client_code(bot_factory)
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/strategy/test_strategy.py

```
import unittest
from unittest.mock import patch
from strategy import Context, ConcreteStrategyA, ConcreteStrategyB, Strategy, reverse_alphabetical

class TestStrategyPattern(unittest.TestCase):
    def test_concrete_strategy_a(self):
        """Test ConcreteStrategyA."""
        strategy = ConcreteStrategyA()
        data = ["e", "b", "d", "a", "c"]
        result = strategy.do_algorithm(data)
        self.assertEqual(result, ["a", "b", "c", "d", "e"])

    def test_concrete_strategy_b(self):
        """Test ConcreteStrategyB."""
        strategy = ConcreteStrategyB()
        data = ["e", "b", "d", "a", "c"]
        result = list(strategy.do_algorithm(data))
        self.assertEqual(result, ["e", "d", "c", "b", "a"])

    def test_context_with_different_strategies(self):
        """Test the context with different strategies."""
        context = Context(ConcreteStrategyA())
        data = ["e", "b", "d", "a", "c"]

        with patch('sys.stdout') as mock_stdout:
            context.do_some_business_logic()
            self.assertIn(','.join(sorted(data)), mock_stdout.getvalue())

        context.strategy = ConcreteStrategyB()
        with patch('sys.stdout') as mock_stdout:
            context.do_some_business_logic()
            self.assertIn(','.join(reversed(sorted(data))), mock_stdout.getvalue())

        context.strategy = Strategy(lambda data: reverse_alphabetical(data))
        with patch('sys.stdout') as mock_stdout:
            context.do_some_business_logic()
            self.assertIn(','.join(reversed(sorted(data))), mock_stdout.getvalue())

def main():
    unittest.main()

if __name__ == '__main__':
    main()
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/strategy/__init__.py

```
from .strategy import *
```

/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/strategy/strategy.py

```
from abc import ABC, abstractmethod
from typing import List, Callable, Any
from dataclasses import dataclass

class Strategy(ABC):
    @abstractmethod
    def do_algorithm(self, data: List[str]) -> List[str]:
        pass

class ConcreteStrategyA(Strategy):
    def do_algorithm(self, data: List[str]) -> List[str]:
        return sorted(data)

class ConcreteStrategyB(Strategy):
    def do_algorithm(self, data: List[str]) -> List[str]:
        return reversed(sorted(data))

# Example of a functional strategy using a lambda function
reverse_alphabetical = lambda data: reversed(sorted(data))

@dataclass
class Context:
    strategy: Strategy

    def do_some_business_logic(self) -> None:
        result = self.strategy.do_algorithm(["a", "b", "c", "d", "e"])
        print(",".join(result))

# Example usage
if __name__ == "__main__":
    context = Context(ConcreteStrategyA())
    print("Client: Strategy is set to normal sorting.")
    context.do_some_business_logic()

    print("\nClient: Strategy is set to reverse sorting.")
    context.strategy = ConcreteStrategyB()
    context.do_some_business_logic()

    # Using a functional strategy
    print("\nClient: Strategy is set to functional reverse sorting.")
    context.strategy = Strategy(lambda data: reverse_alphabetical(data))
    context.do_some_business_logic()
```

```
# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/conftest.py

import pytest

@pytest.fixture
def director():
    return Director()  # Replace with the actual object creation logic


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/map_file_structure.py

import os
import time
import argparse
import pwd
import grp
import hashlib
import mimetypes
import subprocess
import stat
from pathlib import Path

from stream_to_console import stc



def get_file_details(file_path):
    """
    Retrieves comprehensive details about a file.

    Args:
    file_path (str): Path to the file.

    Returns:
    dict: A dictionary containing various file details.
    """
    try:
        # Basic file stats
        stats = os.stat(file_path)
        file_info = {
            "size": stats.st_size,
            "last_modified": time.ctime(stats.st_mtime),
            "last_accessed": time.ctime(stats.st_atime),
            "created": time.ctime(stats.st_ctime)
        }

        # Owner and permissions
        file_info["owner"] = stat.filemode(stats.st_mode)
        file_info["uid"] = stats.st_uid
        file_info["gid"] = stats.st_gid
        # File hash for integrity
        hasher = hashlib.sha256()
        with open(file_path, 'rb') as file:
            buf = file.read()
            hasher.update(buf)
        file_info["hash"] = hasher.hexdigest()

        # Additional details based on file type
        if file_path.endswith('.py'):  # Example for Python files
            with open(file_path, 'r') as file:
                lines = file.readlines()
                file_info["line_count"] = len(lines)
                # Additional Python-specific analysis can be done here

        return file_info

    except Exception as e:
        return {"error": str(e)}


def parse_arguments():
    parser = argparse.ArgumentParser(description='Generate file structure with optional verbosity and deep scan')
    parser.add_argument('-v', '--verbose', action='store_true', help='Enable verbose output')
    parser.add_argument('-d', '--deep_scan', action='store_true', help='Enable deep scanning for additional file details')
    return parser.parse_args()

def print_verbose(message, status):
    if status == "info":
        # Green for file details, blue for summary headers
        parts = message.split(":")
        colored_message = "\033[32m" + parts[0] + "\033[0m" if len(parts) == 1 else "\033[34m" + parts[0] + ":\033[32m"
+ ":".join(parts[1:]) + "\033[0m"
        print(colored_message)
    else:
        # Default colors for other statuses
        color_code = "32" if status == "success" else "31"
        print(f"\033[{color_code}m{message}\033[0m")

def print_verbose_info(message):
    # Blue color for variable content
    print(f"\033[34m{message}\033[0m", end="")

def print_verbose_label(message):
    # Grey color for non-variable text
    print(f"\033[90m{message}\033[0m", end="")

def color_text(text, color_code):
    return f"\033[{color_code}m{text}\033[0m"

def format_file_size(size):
    for unit in ['B', 'KB', 'MB', 'GB', 'TB']:
        if size < 1024:
            return f"{size:.2f}{unit}"
        size /= 1024
```

```
def get_file_permissions(file_path):
    permissions = oct(os.stat(file_path).st_mode)[-3:]
    return permissions

def get_file_owner(file_path):
    uid = os.stat(file_path).st_uid
    gid = os.stat(file_path).st_gid
    user = pwd.getpwuid(uid).pw_name
    group = grp.getgrgid(gid).gr_name
    return user, group

def get_file_hash(file_path):
    sha256_hash = hashlib.sha256()
    with open(file_path, "rb") as f:
        for byte_block in iter(lambda: f.read(4096), b""):
            sha256_hash.update(byte_block)
    return sha256_hash.hexdigest()

def get_file_type(file_path):
    if os.path.islink(file_path):
        return "Symbolic Link"
    elif os.path.isdir(file_path):
        return "Directory"
    elif os.path.isfile(file_path):
        return "File"
    else:
        return "Other"

def get_mime_type(file_path):
    mime_type, _ = mimetypes.guess_type(file_path)
    return mime_type if mime_type else "Unknown"

def get_git_commit_history(file_path, script_dir):
    try:
        cmd = f"git log -n 3 --pretty=format:'%h - %s (%cr)' -- {file_path}"
        process = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, cwd=script_dir)
        stdout, stderr = process.communicate()
        return stdout.decode().strip() if stdout else "Not available"
    except Exception as e:
        return str(e)

def print_summary(total_files, file_types, mime_types):
    print_verbose("File summary:", "info")

    # Print the summary details directly from the variables, not from the file
    print_verbose(f"Total files processed: \033[34m{total_files}\033[32m", "info")
    print_verbose("File types distribution:", "info")
    for f_type, count in file_types.items():
        print_verbose(f"  \033[32m{f_type}: \033[34m{count}\033[32m", "info")
    print_verbose("MIME types distribution:", "info")
    for m_type, count in mime_types.items():
        print_verbose(f"  \033[32m{m_type}: \033[34m{count}\033[32m", "info")
    print('')
```

```
def print_deep_scan_summary(deep_scan_details):
    if deep_scan_details:  # Check if there are any deep scan details
        print_verbose("Deep scan details:", "info")
        for file_path, details in deep_scan_details.items():
            detail_parts = [color_text(f"File: {os.path.basename(file_path)}", 34)]
            for key, value in details.items():
                detail_parts.append(color_text(f"{key.capitalize()}: {value}", 34))
            print_verbose(", ".join(detail_parts), "info")
        print('')



def generate_file_structure(script_dir, run_name, base_output_dir='file_tree/runs',
                            skip_dirs=None, include_hidden=False, deep_scan=False, verbose=False):
    if skip_dirs is None:
        skip_dirs = ['bin', 'lib', 'include', 'your_lib_folder', 'archive', '.git', '__pycache__']

    output_dir = os.path.join(base_output_dir, run_name)
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    output_file = os.path.join(output_dir, 'file_structure.txt')
    summary_file = os.path.join(output_dir, 'summary.txt')
    error_log_file = os.path.join(output_dir, 'error_log.txt')

    total_files = 0
    file_types = {}
    mime_types = {}
    deep_scan_details = {}

    def update_distributions(file_type, mime_type):
        nonlocal total_files, file_types, mime_types
        total_files += 1
        file_types[file_type] = file_types.get(file_type, 0) + 1
        mime_types[mime_type] = mime_types.get(mime_type, 0) + 1

    with open(output_file, 'w') as file_out, open(summary_file, 'w') as summary_out, open(error_log_file, 'w') as
error_log:
        for root, dirs, files in os.walk(script_dir):
            if not include_hidden:
                dirs[:] = [d for d in dirs if not d.startswith('.')]
                files = [f for f in files if not f.startswith('.')]
            dirs[:] = [d for d in dirs if d not in skip_dirs]

            for f in files:
                file_path = os.path.join(root, f)
                try:
                    file_stat = os.stat(file_path)
                    file_size = format_file_size(file_stat.st_size)
                    mod_time = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(file_stat.st_mtime))
                    file_type = get_file_type(file_path)
                    mime_type = get_mime_type(file_path)
                    update_distributions(file_type, mime_type)
                if deep_scan:
                    file_hash = get_file_hash(file_path) if deep_scan else "N/A"
                    git_history = get_git_commit_history(file_path, script_dir) if deep_scan else "N/A"
                    deep_scan_details[file_path] = {'hash': file_hash, 'git_history': git_history}

                file_detail = f'{os.path.join(root.replace(script_dir, ""), f)} - Type: {file_type} MIME: {mime_type} Size:
{file_size}'
                file_out.write(f'{file_detail} | {deep_scan_details} Modified: {mod_time}\n')

                if verbose:
                    print_verbose_label("Name: ")
                    print_verbose_info(f"{os.path.basename(file_path)}, ")
                    print_verbose_label("Type: ")
                    print_verbose_info(f"{file_type} ")
                    print_verbose_label("MIME: ")
                    print_verbose_info(f"{mime_type} ")
                    print_verbose_label("Size: ")
                    print_verbose_info(f"{file_size} ")
                    print_verbose_label("Last Modified: ")
                    print_verbose_info(f"{mod_time} ")
                    if deep_scan:
                        for file_path, details in deep_scan_details.items():
                            detail_parts = [color_text(f"File: {os.path.basename(file_path)}", 34)]
                            for key, value in details.items():
                                detail_parts.append(color_text(f"{key.capitalize()}: {value}", 34))
                            print_verbose(", ".join(detail_parts), "info")
                    print('')

            except Exception as e:
                error_log.write(f"Error processing file {file_path}: {e}\n")
                if verbose:
                    print_verbose(f"\rError processing file {file_path}: {e}", "error")

        summary_out.write(f'Total files processed: {total_files}\n')
        summary_out.write('File types distribution:\n')
        for f_type, count in file_types.items():
            summary_out.write(f'  {f_type}: {count}\n')
        summary_out.write('MIME types distribution:\n')
        for m_type, count in mime_types.items():
            summary_out.write(f'  {m_type}: {count}\n')



        if deep_scan:
            print_verbose("Deep scan details:", "info")
            for file, details in deep_scan_details.items():
                print_verbose(f"\033[32mFile: \033[34m{file}\033[32m Hash: \033[34m{details['hash']}\033[32m Git
History: \033[34m{details['git_history']}\033[32m", "info")

        if verbose:
            print_verbose(f"\rFile structure generation complete. Total files processed: {total_files}", "success")
            print_summary(total_files, file_types, mime_types)

        if verbose and deep_scan:
            print_deep_scan_summary(deep_scan_details)
        # At the end, just print the total files processed
        print(f"File map complete. Total files processed: {total_files}")

if __name__ == '__main__':
    args = parse_arguments()
    verbose = args.verbose
    deep_scan = args.deep_scan
    script_directory = os.getcwd()
    current_time = time.strftime("%Y%m%d_%H%M%S")
    run_name = f'run_{current_time}'
    generate_file_structure(script_directory, run_name, deep_scan=deep_scan, verbose=verbose)


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/test_design_patterns.py

# # test_design_patterns.py
# New way of testing

import importlib
import sys

def run_all_tests():
    patterns = [
        "factory", "builder", "prototype", "singleton", "adapter",
        "bridge", "composite", "decorator", "facade", "flyweight",
        "proxy", "chain_of_responsibility", "command", "iterator",
        "observer", "memento", "mediator", "memoize", "state",
        "strategy", "template", "visitor"
        # Add more patterns as needed
    ]

    for pattern in patterns:
        try:
            test_module = importlib.import_module(f"{pattern}.test_{pattern}")
            print(f"\nTesting {pattern.capitalize()} Pattern:")
            test_module.run_tests()
        except ModuleNotFoundError:
            print(f"Test module for {pattern} not found.", file=sys.stderr)
        except AttributeError:
            print(f"No run_tests() function in test_{pattern}.", file=sys.stderr)

if __name__ == "__main__":
    run_all_tests()



# Old way of testing with unittest
# # Test standard design patterns
# from factory.test_factory import main as test_factory_main
# from builder.test_builder import main as test_builder_main
# from prototype.test_prototype import main as test_prototype_main
# from singleton.test_singleton import main as test_singleton_main
# from adapter.test_adapter import main as test_adapter_main
# from bridge.test_bridge import main as test_bridge_main
# from composite.test_composite import main as test_composite_main
# from decorator.test_decorator import main as test_decorator_main
# from facade.test_facade import main as test_facade_main
# from flyweight.test_flyweight import main as test_flyweight_main
# from proxy.test_proxy import main as test_proxy_main
# from chain_of_responsibility.test_chain_of_responsibility import main as test_chain_of_responsibility_main
# from command.test_command import main as test_command_main
# from iterator.test_iterator import main as test_iterator_main
# from observer.test_observer import main as test_observer_main
# from memento.test_memento import main as test_memento_main
# from mediator.test_mediator import main as test_mediator_main
# from memoize.test_memoize import main as test_memoize_main
# from state.test_state import main as test_state_main
# from strategy.test_strategy import main as test_strategy_main
# from template.test_template import main as test_template_main
# from visitor.test_visitor import main as test_visitor_main

# # Test composite design patterns
# # from event_queue.test_queue import main as test_event_queue_main
# # from thread_pool.test_thread_pool import main as test_thread_pool_main
# # from web_scraper.test_scraper import main as test_scraper_main
# # from zip_file.test_zip_file import main as test_zip_file_main

# def run_all_tests():
#     print("Testing Factory Pattern:")
#     test_factory_main()

#     print("\n\nTesting Builder Pattern:")
#     test_builder_main()

#     print("\n\nTesting Prototype Pattern:")
#     test_prototype_main()

#     print("\n\nTesting Singleton Pattern:")
#     test_singleton_main()

#     print("\n\nTesting Adapter Pattern:")
#     test_adapter_main()

#     print("\n\nTesting Bridge Pattern:")
#     test_bridge_main()

#     print("\n\nTesting Composite Pattern:")
#     test_composite_main()

#     print("\n\nTesting Decorator Pattern:")
#     test_decorator_main()

#     print("\n\nTesting Facade Pattern:")
#     test_facade_main()

#     print("\n\nTesting Flyweight Pattern:")
#     test_flyweight_main()
#     print("\n\nTesting Proxy Pattern:")
#     test_proxy_main()

#     print("\n\nTesting Chain of Responsibility Pattern:")
#     test_chain_of_responsibility_main()

#     print("\n\nTesting Command Pattern:")
#     test_command_main()

#     print("\n\nTesting Iterator Pattern:")
#     test_iterator_main()

#     print("\n\nTesting Observer Pattern:")
#     test_observer_main()

#     print("\n\nTesting State Pattern:")
#     test_state_main()

#     print("\n\nTesting Strategy Pattern:")
#     test_strategy_main()

#     print("\n\nTesting Template Method Pattern:")
#     test_template_main()

#     print("\n\nTesting Visitor Pattern:")
#     test_visitor_main()

#     print("\n\nTesting Memento Pattern:")
#     test_memento_main()

#     print("\n\nTesting Mediator Pattern:")
#     test_mediator_main()

#     print("\n\nTesting Memoize Pattern:")
#     test_memoize_main()

#     # print("\n\nTesting Event Queue Pattern:")
#     # test_event_queue_main()

#     # print("\n\nTesting Thread Pool Pattern:")
#     # test_thread_pool_main()

#     # print("\n\nTesting Web Scraper Pattern:")
#     # test_scraper_main()

#     # print("\n\nTesting Zip File Pattern:")
#     # test_zip_file_main()

#     #=============================================================================#
#     print()

# if __name__ == "__main__":
#     run_all_tests()
# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/copy_codebase_to_file.py

import os
import argparse
import logging
from datetime import datetime
import zipfile
import json
import xml.etree.ElementTree as ET
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

def write_to_pdf(file_data, output_file):
    c = canvas.Canvas(output_file, pagesize=letter)
    width, height = letter
    c.drawString(30, height - 30, "Extracted Code Report")

    y_position = height - 50
    for file_path, content in file_data.items():
        c.drawString(30, y_position, file_path)
        y_position -= 20
        for line in content.split('\n'):
            c.drawString(40, y_position, line)
            y_position -= 15
            if y_position < 40:
                c.showPage()
                y_position = height - 50

    c.save()

def ensure_directory_exists(directory):
    if not os.path.exists(directory):
        os.makedirs(directory)

def write_to_json(file_data, output_file):
    ensure_directory_exists(os.path.dirname(output_file))
    with open(output_file, 'w') as json_file:
        json.dump(file_data, json_file, indent=4)

def write_to_xml(file_data, output_file):
    ensure_directory_exists(os.path.dirname(output_file))
    root = ET.Element("files")
    for file_path, content in file_data.items():
        file_elem = ET.SubElement(root, "file", path=file_path)
        content_elem = ET.SubElement(file_elem, "content")
        content_elem.text = content

    tree = ET.ElementTree(root)
    tree.write(output_file)

def setup_logging(log_level, log_file=None):
    log_format = '%(asctime)s - %(levelname)s - %(message)s'
    logging.basicConfig(filename=log_file if log_file else None,
                        level=log_level, format=log_format)

def parse_arguments():
    parser = argparse.ArgumentParser(description='Extract Python code from a directory into separate files in an output
folder.')
    parser.add_argument('--directory', type=str, help='Directory to scan for Python files (relative or absolute).')
    parser.add_argument('--output_folder', type=str, help='Folder to write extracted code into separate files (relative or
absolute).')
    parser.add_argument('--log', type=str, help='Optional log file')
    parser.add_argument('--log_level', type=str, default='INFO', choices=['DEBUG', 'INFO', 'WARNING', 'ERROR',
'CRITICAL'],
                        help='Set the logging level (default: INFO)')
    parser.add_argument('--min_size', type=int, default=0, help='Minimum file size in bytes')
    parser.add_argument('--max_size', type=int, default=None, help='Maximum file size in bytes')
    parser.add_argument('--before_date', type=str, default=None, help='Filter files modified before this date (YYYY-
MM-DD)')
    parser.add_argument('--format', type=str, default='txt', choices=['txt', 'json', 'xml', 'pdf'],
                        help='Output format: txt, json, xml, or pdf (default: txt)')
    return parser.parse_args()

def is_python_file(file_path):
    return file_path.endswith('.py')

def filter_files(file_path, min_size, max_size, before_date):
    try:
        file_stat = os.stat(file_path)
        file_size = file_stat.st_size
        file_mod_time = datetime.fromtimestamp(file_stat.st_mtime)

        if (min_size is not None and file_size < min_size) or \
           (max_size is not None and file_size > max_size) or \
           (before_date is not None and file_mod_time > before_date):
            return False
        return True
    except Exception as e:
        logging.error(f"Error filtering file {file_path}: {e}")
        return False

def recursive_traverse_directory(directory, min_size, max_size, before_date):
    for root, dirs, files in os.walk(directory):
        for file in files:
            file_path = os.path.join(root, file)
            if is_python_file(file_path) and filter_files(file_path, min_size, max_size, before_date):
                yield file_path

def read_file(file_path):
    try:
        with open(file_path, 'r') as infile:
            content = infile.read()
            return content
    except IOError as e:
        logging.error(f"Error reading file {file_path}: {e}")
        return None
```

```
def write_to_single_file(file_path, content, outfile):
    outfile.write(f"\n\n# File: {file_path}\n\n")
    outfile.write(content)

def extract_python_code(directory, output_file, min_size, max_size, before_date):
    try:
        with open(output_file, 'w') as outfile:
            for file_path in recursive_traverse_directory(directory, min_size, max_size, before_date):
                content = read_file(file_path)
                if content:
                    write_to_single_file(file_path, content, outfile)
    except Exception as e:
        logging.error(f"Error while writing to file {output_file}: {e}")

def convert_date_string(date_str):
    return datetime.strptime(date_str, '%Y-%m-%d') if date_str else None

def zip_folder(output_folder, zip_file_name):
    try:
        with zipfile.ZipFile(zip_file_name, 'w', zipfile.ZIP_DEFLATED) as zipf:
            for root, dirs, files in os.walk(output_folder):
                for file in files:
                    file_path = os.path.join(root, file)
                    zipf.write(file_path, os.path.relpath(file_path, output_folder))
        logging.info(f"Folder zipped into: {zip_file_name}")
    except Exception as e:
        logging.error(f"Error zipping folder {output_folder}: {e}")

if __name__ == '__main__':
    print("Running script...")
    args = parse_arguments()
    current_dir = os.getcwd()

    # Set default for directory
    args.directory = args.directory or current_dir

    # Set default for output folder
    cwd_name = os.path.basename(current_dir)
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    args.output_folder = args.output_folder or os.path.join(current_dir, f"{cwd_name}_codebase_copies",
f"codebase_copy_{timestamp}")

    # Ensure the output folder exists
    ensure_directory_exists(args.output_folder)

    # Generate output filename based on chosen format
    args.format = args.format or 'txt'
    output_file = os.path.join(args.output_folder, f"extracted_code_{timestamp}.{args.format}")

    setup_logging(args.log_level, args.log)

    # Convert date string to datetime object
    before_date = convert_date_string(args.before_date) if args.before_date else None
    # Process files based on the chosen format
    if args.format in ['json', 'xml', 'pdf']:
        file_data = {}
        for file_path in recursive_traverse_directory(args.directory, args.min_size, args.max_size, before_date):
            content = read_file(file_path)
            if content:
                file_data[file_path] = content

        if args.format == 'json':
            write_to_json(file_data, output_file)
        elif args.format == 'xml':
            write_to_xml(file_data, output_file)
        elif args.format == 'pdf':
            write_to_pdf(file_data, output_file)
    else:
        # Default to text format
        extract_python_code(args.directory, output_file, args.min_size, args.max_size, before_date)

    print("Script execution completed.")


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/stream_to_console.py

# NovaSystem/src/utils/stream_to_console.py

import traceback
import sys
import time
import art
import random
from colorama import Fore, Back, Style, init
# from ..utils.border_maker import border_maker

# ANSI Escape Codes for additional styles
ANSI_STYLES = {
    "underline": "\033[4m",
    "double_underline": "\033[21m",
    "invert_colors": "\033[7m",
    "italic": "\033[3m",
    "strikethrough": "\033[9m",
    "reset": "\033[0m"
}

# Initialize colorama
init(autoreset=True)

def apply_color(text, foreground_color=None, background_color=None, style=None):
    """Applies color and style to text."""
    colored_text = text
    if foreground_color:
        if hasattr(Fore, foreground_color.upper()):
            colored_text = getattr(Fore, foreground_color.upper()) + colored_text
        else:
            raise ValueError(f"Invalid foreground color: {foreground_color}")

    if background_color:
        if hasattr(Back, background_color.upper()):
            colored_text = getattr(Back, background_color.upper()) + colored_text
        else:
            raise ValueError(f"Invalid background color: {background_color}")

    if style:
        colored_text = style + colored_text
    return colored_text

font_options = [
    "block", "caligraphy","graffiti", "colossal",
    "sub-zero", "slant", "fancy1", "fancy2", "fancy3",
    "fancy4", "fancy5", "fancy6", "fancy7", "fancy8", "fancy9",
    "fancy10", "fancy11", "fancy12", "fancy13", "fancy14",
    "fancy15", "fancy16", "fancy17", "fancy18", "fancy19",
    "fancy20", "banner", "big", "bubble", "digital", "ivrit",
    "mirror", "script", "shadow", "speed", "stampatello",
    "term", "avatar", "barbwire", "bear", "bell", "benjamin",
    "bigchief", "binary", "broadway", "bubblebath", "bulbhead",
    "chunky", "coinstak", "contessa", "contrast", "cosmic",
    "cosmike", "cricket", "cyberlarge", "cybermedium", "cybersmall",
    "decimal", "diamond", "dietcola", "digital", "doh",
    "doom", "dotmatrix", "double", "drpepper", "eftichess",
    "eftifont", "eftipiti", "eftirobot", "eftitalic", "eftiwall",
    "eftiwater", "epic", "fender", "fourtops", "fraktur",
    "goofy", "gothic", "graceful", "gradient", "helv",
    "hollywood", "invita", "isometric1", "isometric2", "isometric3",
    "isometric4", "italic", "jazmine", "jerusalem", "katakana",
    "kban", "keyboard", "knob", "larry3d", "lcd",
    "lean", "letters", "linux", "lockergnome", "madrid",
    "marquee", "maxfour", "mike", "mini", "mirror",
    "mnemonic", "morse", "moscow", "mshebrew210", "nancyj",
    "nancyj-fancy", "nancyj-underlined", "nipples", "ntgreek", "nvscript",
    "o8", "ogre", "pawp", "peaks", "pebbles",
    "pepper", "poison", "puffy", "pyramid", "rectangles",
    "relief", "relief2", "rev", "roman", "rot13",
    "rounded", "rowancap", "rozzo", "runic", "runyc",
    "sblood", "script", "serifcap", "shadow", "short",
    "slscript", "small", "smisome1", "smkeyboard", "smscript",
    "smshadow", "smslant", "smtengwar", "speed", "stampatello",
    "standard", "starwars", "stellar", "stop", "straight",
    "tanja", "tengwar", "term", "thick", "thin",
    "threepoint", "ticks", "ticksslant", "tinker-toy", "tombstone",
    "trek", "tsalagi", "twopoint", "univers", "usaflag",
    "wavy", "weird"
]

def apply_colorama_style(bold=False, underline=False, invert_colors=False, double_underline=False, hidden=False,
italic=False, strikethrough=False, style=None, fg_style=None, bg_style=None):
    """Returns the combined style string based on flags."""
    style_str = ''
    if bold:
        style_str += Style.BRIGHT
    if hidden:
        style_str += Style.DIM
    if underline:
        style_str += ANSI_STYLES["underline"]
    if double_underline:
        style_str += ANSI_STYLES["double_underline"]
    if invert_colors:
        style_str += ANSI_STYLES["invert_colors"]
    if italic:
        style_str += ANSI_STYLES["italic"]
    if strikethrough:
        style_str += ANSI_STYLES["strikethrough"]
    if fg_style:
        if fg_style == "DIM":
            style_str += Style.DIM
        if fg_style == "BRIGHT":
            style_str += Style.BRIGHT
        if fg_style == "NORMAL":
            style_str += Style.NORMAL
        if fg_style == "RESET_ALL":
            style_str += Style.RESET_ALL
    if bg_style:
        if bg_style == "DIM":
            style_str += Style.DIM
        if bg_style == "BRIGHT":
            style_str += Style.BRIGHT
        if bg_style == "NORMAL":
            style_str += Style.NORMAL
        if bg_style == "RESET_ALL":
            style_str += Style.RESET_ALL
    if style:
        if style == "DIM":
            style_str += Style.DIM
        if style == "BRIGHT":
            style_str += Style.BRIGHT
        if style == "NORMAL":
            style_str += Style.NORMAL
        if style == "RESET_ALL":
            style_str += Style.RESET_ALL
    return style_str

def stream_to_console(message, delay=0.0035, foreground_color=None, background_color=None,
rainbow_effect=False, **style_flags):
    """
    Streams a message to the console character by character with optional delay, colors, and effects.
    """
    # Validate input types
    if not isinstance(message, str):
        raise TypeError("Message must be a string.")
    if not isinstance(delay, (float, int)):
        raise TypeError("Delay must be a number.")
    # Stream function
    try:
        # Validate delay
        delay = max(0.0001, min(delay, 1.0))  # Clamp delay

        # Style string
        style_str = apply_colorama_style(**style_flags)

        # Stream each character
        for char in message:
            if rainbow_effect:
                fg_color = random.choice(["RED", "GREEN", "YELLOW", "BLUE", "MAGENTA", "CYAN"])
                char = apply_color(char, foreground_color=fg_color, background_color=background_color, style=style_str)
            else:
                char = apply_color(char, foreground_color, background_color, style_str)

            sys.stdout.write(char)
            sys.stdout.flush()
            time.sleep(delay)

        # Reset color at the end
        sys.stdout.write(Style.RESET_ALL)
        sys.stdout.flush()
    except Exception as e:
        exc_type, exc_value, exc_traceback = sys.exc_info()
        traceback_details = {
            'filename': exc_traceback.tb_frame.f_code.co_filename,
            'lineno': exc_traceback.tb_lineno,
            'name': exc_traceback.tb_frame.f_code.co_name,
            'type': exc_type.__name__,
            'message': str(exc_value),
        }
        error_message = "Error in stream_to_console: [{}] {}".format(traceback_details['type'],
traceback_details['message'])
        error_details = "File: {}, Line: {}, In: {}".format(traceback_details['filename'], traceback_details['lineno'],
traceback_details['name'])
        sys.stderr.write(error_message + "\n" + error_details + "\n")
        sys.stderr.flush()
        raise

    print()  # Newline at the end
# Example usage and test cases remain the same

# Example usage
# stream_to_console("Hello, NovaSystem AI!", rainbow_effect=True)

# Test cases as a list of dictionaries
test_cases = [
    {"message": "Simple message with default settings."},
    {"message": "Slower text...", "delay": 0.05},
    {"message": "Red text.", "foreground_color": "red"},
    {"message": "Green text.", "foreground_color": "green"},
    {"message": "Green on blue.", "foreground_color": "green", "background_color": "blue"},
    {"message": "Rainbow effect!", "rainbow_effect": True},
    {"message": "Slower rainbow text...", "delay": 0.07, "rainbow_effect": True},
    {"message": "Green on red, slowly.", "delay": 0.05, "foreground_color": "green", "background_color": "red"},
    {"message": "Bold text.", "bold": True},
    {"message": "Underlined text.", "underline": True},
    {"message": "Inverted colors.", "invert_colors": True},
    {"message": "Blue background.", "background_color": "blue"},
    {"message": "Cyan text on yellow.", "foreground_color": "cyan", "background_color": "yellow"},
    {"message": "Double underline.", "double_underline": True},
    {"message": "Hidden text.", "hidden": True},
    {"message": "Slower inverted rainbow text...", "delay": 0.07, "rainbow_effect": True, "invert_colors": True},
    {"message": "Italicized text.", "italic": True},
    {"message": "Strikethrough text.", "strikethrough": True},
]

def test():
    # Generate ASCII art with a random font
  random_font = random.choice(font_options)
  random_ascii_art = art.text2art("NovaSystem", font=random_font)

  # Stream the ASCII art first
  stream_to_console(random_ascii_art, delay=0.0004)

  # Stream each test case
  for case in test_cases:
      stream_to_console(**case)

stc = stream_to_console

if __name__ == "__main__":
    test()

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/proxy/proxy.py

from abc import ABC, abstractmethod
import datetime
now = datetime.datetime.now()

class Subject(ABC):
    """
    The Subject interface declares common operations for both RealSubject and the Proxy.
    """
    @abstractmethod
    def request(self) -> None:
        pass

class RealSubject(Subject):
    """
    The RealSubject contains core business logic.
    """
    def request(self) -> None:
        print("RealSubject: Handling request.")

class Proxy(Subject):
    """
    The Proxy has an interface identical to the RealSubject.
    """
    def __init__(self, real_subject: RealSubject) -> None:
        self._real_subject = real_subject

    def request(self) -> None:
        if self.check_access():
            self._real_subject.request()
            self.log_access()

    def check_access(self) -> bool:
        print("Proxy: Checking access prior to firing a real request.")
        return True

    def log_access(self) -> None:
        print("Proxy: Logging the time of request.", end="")
        print(f"Time: {now.time()}")

# Client code example
def client_code(subject: Subject) -> None:
    subject.request()

# Example usage
if __name__ == "__main__":
    real_subject = RealSubject()
    proxy = Proxy(real_subject)

    print("Client: Executing with RealSubject:")
    client_code(real_subject)

    print("\nClient: Executing with Proxy:")
    client_code(proxy)


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/proxy/__init__.py

from .proxy import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/proxy/test_proxy.py

from proxy import RealSubject, Proxy, client_code

def test_real_subject():
    print("Testing RealSubject:")
    real_subject = RealSubject()
    client_code(real_subject)

def test_proxy():
    print("\nTesting Proxy:")
    real_subject = RealSubject()
    proxy = Proxy(real_subject)
    client_code(proxy)

def main():
    test_real_subject()
    test_proxy()

if __name__ == "__main__":
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/decorator/decorator.py

class AIComponent:
    """
    Base AIComponent interface defines operations that can be altered by decorators.
    """
    def operation(self) -> str:
        pass

class ConcreteAIComponent(AIComponent):
    """
    Concrete AIComponents provide default implementations of the operations.
    """
    def operation(self) -> str:
        return "ConcreteAIComponent"

class Decorator(AIComponent):
    """
    Base Decorator class follows the same interface as other components.
    """
    _component: AIComponent = None

    def __init__(self, component: AIComponent) -> None:
        self._component = component

    def operation(self) -> str:
        return self._component.operation()

class LoggingDecorator(Decorator):
    """
    Concrete Decorator that adds logging functionality.
    """
    def operation(self) -> str:
        # Additional behavior before calling the wrapped object
        result = self._component.operation()
        # Additional behavior after calling the wrapped object
        return f"LoggingDecorator({result})"

class PerformanceDecorator(Decorator):
    """
    Concrete Decorator that adds performance tracking functionality.
    """
    def operation(self) -> str:
        # Performance tracking behavior
        result = self._component.operation()
        # Additional behavior
        return f"PerformanceDecorator({result})"
```

```
# Client code
def client_code(component: AIComponent) -> None:
    print(f"RESULT: {component.operation()}", end="")

# Example usage
if __name__ == "__main__":
    simple_component = ConcreteAIComponent()
    decorated_component = PerformanceDecorator(LoggingDecorator(simple_component))

    print("Client: I've got a component with additional behaviors:")
    client_code(decorated_component)


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/decorator/__init__.py

from .decorator import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/decorator/test_decorator.py

from .decorator import ConcreteAIComponent, LoggingDecorator, PerformanceDecorator

def test_decorator_pattern():
    # Testing with the basic AI component
    basic_component = ConcreteAIComponent()
    print("Basic AI Component:", basic_component.operation())

    # Adding Logging functionality
    logged_component = LoggingDecorator(basic_component)
    print("Logged AI Component:", logged_component.operation())

    # Adding Performance tracking on top of Logging
    perf_logged_component = PerformanceDecorator(logged_component)
    print("Performance Tracked and Logged AI Component:", perf_logged_component.operation())

def main():
    test_decorator_pattern()

if __name__ == "__main__":
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/bridge/bridge.py

# bridge.py

from __future__ import annotations
from abc import ABC, abstractmethod

class Abstraction:
    """
    The Abstraction defines the interface for the 'control' part of the two
    class hierarchies. It maintains a reference to an object of the
    Implementation hierarchy and delegates all of the real work to this object.
    """
    def __init__(self, implementation: Implementation) -> None:
        self.implementation = implementation

    def operation(self) -> str:
        return (f"Abstraction: Base operation with:\n"
                f"{self.implementation.operation_implementation()}")

class ExtendedAbstraction(Abstraction):
    """
    You can extend the Abstraction without changing the Implementation classes.
    """
    def operation(self) -> str:
        return (f"ExtendedAbstraction: Extended operation with:\n"
                f"{self.implementation.operation_implementation()}")

class Implementation(ABC):
    """
    The Implementation defines the interface for all implementation classes. It
    doesn't have to match the Abstraction's interface. In fact, the two
    interfaces can be entirely different. Typically the Implementation interface
    provides only primitive operations, while the Abstraction defines higher-
    level operations based on those primitives.
    """
    @abstractmethod
    def operation_implementation(self) -> str:
        pass

class ConcreteImplementationA(Implementation):
    def operation_implementation(self) -> str:
        return "ConcreteImplementationA: Here's the result on the platform A."

class ConcreteImplementationB(Implementation):
    def operation_implementation(self) -> str:
        return "ConcreteImplementationB: Here's the result on the platform B."

def client_code(abstraction: Abstraction) -> None:
    """
    Except for the initialization phase, where an Abstraction object gets linked
    with a specific Implementation object, the client code should only depend on
    the Abstraction class. This way the client code can support any abstraction-
    implementation combination.
    """
    print(abstraction.operation(), end="")

if __name__ == "__main__":
    """
    The client code should be able to work with any pre-configured abstraction-
    implementation combination.
    """
    implementation = ConcreteImplementationA()
    abstraction = Abstraction(implementation)
    client_code(abstraction)
    print("\n")

    implementation = ConcreteImplementationB()
    abstraction = ExtendedAbstraction(implementation)
    client_code(abstraction)


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/bridge/__init__.py

from .bridge import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/bridge/test_bridge.py

# test_bridge.py

from bridge import Abstraction, ExtendedAbstraction, ConcreteImplementationA, ConcreteImplementationB

def test_abstraction_with_concrete_implementation_a():
    """
    Test Abstraction with ConcreteImplementationA.
    """
    implementation = ConcreteImplementationA()
    abstraction = Abstraction(implementation)
    result = abstraction.operation()

    assert result == "Abstraction: Base operation with:\nConcreteImplementationA: Here's the result on the platform A.",
\
        "Abstraction with ConcreteImplementationA failed"

    print("PASS: Abstraction with ConcreteImplementationA")

def test_abstraction_with_concrete_implementation_b():
    """
    Test Abstraction with ConcreteImplementationB.
    """
    implementation = ConcreteImplementationB()
    abstraction = Abstraction(implementation)
    result = abstraction.operation()

    assert result == "Abstraction: Base operation with:\nConcreteImplementationB: Here's the result on the platform B.",
\
        "Abstraction with ConcreteImplementationB failed"

    print("PASS: Abstraction with ConcreteImplementationB")

def test_extended_abstraction_with_concrete_implementation_a():
    """
    Test ExtendedAbstraction with ConcreteImplementationA.
    """
    implementation = ConcreteImplementationA()
    abstraction = ExtendedAbstraction(implementation)
    result = abstraction.operation()

    assert result == "ExtendedAbstraction: Extended operation with:\nConcreteImplementationA: Here's the result on the
```

```
platform A.", \
        "ExtendedAbstraction with ConcreteImplementationA failed"

    print("PASS: ExtendedAbstraction with ConcreteImplementationA")

def test_extended_abstraction_with_concrete_implementation_b():
    """
    Test ExtendedAbstraction with ConcreteImplementationB.
    """
    implementation = ConcreteImplementationB()
    abstraction = ExtendedAbstraction(implementation)
    result = abstraction.operation()

    assert result == "ExtendedAbstraction: Extended operation with:\nConcreteImplementationB: Here's the result on the
platform B.", \
        "ExtendedAbstraction with ConcreteImplementationB failed"

    print("PASS: ExtendedAbstraction with ConcreteImplementationB")

def main():
    """
    Main function to run the Bridge pattern tests.
    """
    print("Testing Bridge Pattern Implementations:")
    test_abstraction_with_concrete_implementation_a()
    test_abstraction_with_concrete_implementation_b()
    test_extended_abstraction_with_concrete_implementation_a()
    test_extended_abstraction_with_concrete_implementation_b()

if __name__ == "__main__":
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/template/__init__.py

from .template import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/template/template.py

from abc import ABC, abstractmethod

class AbstractClass(ABC):
    """
    The Abstract Class defines a template method that contains a skeleton of
    some algorithm, composed of calls to (usually) abstract primitive operations.
    Concrete subclasses should implement these operations, but leave the
    template method itself intact.
    """
    def template_method(self) -> None:
        self.base_operation1()
        self.required_operations1()
        self.base_operation2()
        self.hook1()
        self.required_operations2()
        self.base_operation3()
        self.hook2()

    # These operations already have implementations.
    def base_operation1(self):
        print("AbstractClass says: I am doing the bulk of the work")

    def base_operation2(self):
        print("AbstractClass says: But I let subclasses override some operations")

    def base_operation3(self):
        print("AbstractClass says: But I am doing the majority of the work anyway")

    # These operations have to be implemented in subclasses.
    @abstractmethod
    def required_operations1(self):
        pass

    @abstractmethod
    def required_operations2(self):
        pass

    # These are "hooks." Subclasses may override them, but it's not mandatory
    # since the hooks already have default (but empty) implementation.
    def hook1(self):
        pass

    def hook2(self):
        pass

class ConcreteClass1(AbstractClass):
    def required_operations1(self):
        print("ConcreteClass1 says: Implemented Operation1")

    def required_operations2(self):
        print("ConcreteClass1 says: Implemented Operation2")

class ConcreteClass2(AbstractClass):
    def required_operations1(self):
        print("ConcreteClass2 says: Implemented Operation1")

    def required_operations2(self):
        print("ConcreteClass2 says: Implemented Operation2")

    def hook1(self):
        print("ConcreteClass2 says: Overridden Hook1")

# Example usage
if __name__ == "__main__":
    print("Same client code can work with different subclasses:")
    concrete_class1 = ConcreteClass1()
    concrete_class1.template_method()

    print("\nSame client code can work with different subclasses:")
    concrete_class2 = ConcreteClass2()
    concrete_class2.template_method()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/template/test_template.py

import unittest
from unittest.mock import patch
from template import ConcreteClass1, ConcreteClass2

class TestConcreteClass1(unittest.TestCase):
    @patch('sys.stdout')
    def test_required_operations1(self, mock_stdout):
        concrete_class1 = ConcreteClass1()
        concrete_class1.required_operations1()
        mock_stdout.write.assert_called_with("ConcreteClass1 says: Implemented Operation1\n")

    @patch('sys.stdout')
    def test_required_operations2(self, mock_stdout):
        concrete_class1 = ConcreteClass1()
        concrete_class1.required_operations2()
        mock_stdout.write.assert_called_with("ConcreteClass1 says: Implemented Operation2\n")

class TestConcreteClass2(unittest.TestCase):
    @patch('sys.stdout')
    def test_required_operations1(self, mock_stdout):
        concrete_class2 = ConcreteClass2()
        concrete_class2.required_operations1()
        mock_stdout.write.assert_called_with("ConcreteClass2 says: Implemented Operation1\n")

    @patch('sys.stdout')
    def test_required_operations2(self, mock_stdout):
        concrete_class2 = ConcreteClass2()
        concrete_class2.required_operations2()
        mock_stdout.write.assert_called_with("ConcreteClass2 says: Implemented Operation2\n")

    @patch('sys.stdout')
    def test_hook1(self, mock_stdout):
        concrete_class2 = ConcreteClass2()
        concrete_class2.hook1()
        mock_stdout.write.assert_called_with("ConcreteClass2 says: Overridden Hook1\n")

def main():
    unittest.main()

if __name__ == '__main__':
    main()



# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/facade/facade.py

class AIProcessingSubsystem:
    """
    A subsystem that might perform AI-related tasks.
    """
    def initialize(self) -> str:
        return "AIProcessingSubsystem: Initialized and ready to process."

    def process_data(self, data) -> str:
        return f"AIProcessingSubsystem: Processing data - {data}"

class DataAnalysisSubsystem:
    """
    A subsystem for data analysis.
    """
    def analyze(self, data) -> str:
        return f"DataAnalysisSubsystem: Analyzing data - {data}"

class NovaSystemFacade:
    """
    The Facade class provides a simple interface to the complex logic of NovaSystem's subsystems.
    """
    def __init__(self) -> None:
        self._ai_processor = AIProcessingSubsystem()
        self._data_analyzer = DataAnalysisSubsystem()

    def process_and_analyze_data(self, data) -> str:
        results = []
        results.append(self._ai_processor.initialize())
        results.append(self._ai_processor.process_data(data))
        results.append(self._data_analyzer.analyze(data))
        return "\n".join(results)

# Client Code
def client_code(facade: NovaSystemFacade) -> None:
    print(facade.process_and_analyze_data("Sample Data"), end="")

# Example usage
if __name__ == "__main__":
    facade = NovaSystemFacade()
    client_code(facade)



# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/facade/__init__.py

from .facade import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/facade/test_facade.py

from facade import NovaSystemFacade, client_code

def test_novasystem_facade():
    # Creating the Facade instance
    novasystem_facade = NovaSystemFacade()

    # Simulating client interaction with the facade
    print("Testing NovaSystem Facade:")
    client_code(novasystem_facade)
```

```
def main():
    test_novasystem_facade()

if __name__ == "__main__":
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/observer/observer.py

from abc import ABC, abstractmethod

class Observer(ABC):
    def __init__(self):
        self.is_active = True

    @abstractmethod
    def update(self, subject) -> None:
        pass

    def activate(self):
        self.is_active = True

    def deactivate(self):
        self.is_active = False

    def is_observer_active(self) -> bool:
        return self.is_active

    def handle_error(self, error: Exception):
        print(f"Observer error: {error}")

    def pre_update(self):
        pass

    def post_update(self):
        pass


# Example of a concrete observer class with expanded functionality
class AdvancedObserver(Observer):
    def __init__(self):
        super().__init__()

    def update(self, subject) -> None:
        if not self.is_active:
            return

        if subject is None:
            raise ValueError("Subject cannot be None")  # Directly raise the exception

        try:
            self.pre_update()
            # Ensure 'subject' has attribute 'state' before trying to access it
            state = getattr(subject, 'state', 'No state')  # Default value if 'state' is not present
            print(f"AdvancedObserver updated with new state: {state}")
            self.post_update()
        except Exception as e:
            self.handle_error(e)

    def pre_update(self):
        print("Preparing to update AdvancedObserver.")

    def post_update(self):
        print("AdvancedObserver update complete.")

    def handle_error(self, error: Exception):
        print(f"Error in AdvancedObserver: {error}")
        # Optionally, you can re-raise the exception if needed for tests
        raise error


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/observer/test_observer.py

# # DesignPatterns/observer/test_observer.py

import pytest
from io import StringIO
from unittest.mock import patch
from observer import AdvancedObserver

class MockSubject:
    """ A mock subject class for testing the observer. """
    def __init__(self):
        self.state = None

    def change_state(self, new_state):
        self.state = new_state

@pytest.fixture
def observer():
    """ Fixture to create an AdvancedObserver instance. """
    return AdvancedObserver()

@pytest.fixture
def mock_subject():
    """ Fixture to create a MockSubject instance. """
    return MockSubject()

def test_activation(observer):
    """ Test if the observer activates and deactivates correctly. """
    observer.deactivate()
    assert not observer.is_observer_active()

    observer.activate()
    assert observer.is_observer_active()

def test_update_when_active(observer, mock_subject):
    """ Test if the observer updates its state when active. """
    with patch('sys.stdout', new_callable=StringIO) as mock_stdout:
        observer.activate()
        mock_subject.change_state("new_state")
        observer.update(mock_subject)
        assert "AdvancedObserver updated with new state: new_state" in mock_stdout.getvalue()

def test_no_update_when_inactive(observer, mock_subject):
    """ Test if the observer does not update its state when inactive. """
    with patch('sys.stdout', new_callable=StringIO) as mock_stdout:
        observer.deactivate()
        mock_subject.change_state("new_state")
        observer.update(mock_subject)
        assert mock_stdout.getvalue() == ""

def test_error_handling(observer):
    """ Test the error handling in the observer. """
    with patch('sys.stdout', new_callable=StringIO) as mock_stdout:
        observer.activate()
        with pytest.raises(ValueError) as exc_info:
            observer.update(None)

        assert "Subject cannot be None" == str(exc_info.value)

def test_pre_update_hook(observer, mock_subject):
    """ Test the execution of the pre-update hook. """
    with patch('sys.stdout', new_callable=StringIO) as mock_stdout:
        observer.activate()
        observer.update(mock_subject)
        assert "Preparing to update AdvancedObserver." in mock_stdout.getvalue()

def test_post_update_hook(observer, mock_subject):
    """ Test the execution of the post-update hook. """
    with patch('sys.stdout', new_callable=StringIO) as mock_stdout:
        observer.activate()
        observer.update(mock_subject)
        assert "AdvancedObserver update complete." in mock_stdout.getvalue()



# import unittest
# from unittest.mock import patch
# from observer import Observer, AdvancedObserver

# class MockSubject:
#     """
#     A mock subject class to simulate state changes for testing observers.
#     """
#     def __init__(self):
#         self.state = None

#     def change_state(self, new_state):
#         self.state = new_state

# class TestObserver(unittest.TestCase):
```

```
#     """
#     Test suite for the Observer class and its functionalities.
#     """

#     def setUp(self):
#         self.subject = MockSubject()
#         self.observer = AdvancedObserver()

#     def test_activation(self):
#         """ Test if the observer correctly activates and deactivates. """
#         self.observer.deactivate()
#         self.assertFalse(self.observer.is_observer_active())

#         self.observer.activate()
#         self.assertTrue(self.observer.is_observer_active())

#     def test_update_when_active(self):
#         """ Test if the observer updates its state when active. """
#         with patch('sys.stdout') as mock_stdout:
#             self.subject.change_state("new_state")
#             self.observer.activate()
#             self.observer.update(self.subject)
#             self.assertIn("AdvancedObserver updated with new state: new_state", mock_stdout.getvalue())

#     def test_no_update_when_inactive(self):
#         """ Test if the observer does not update its state when inactive. """
#         with patch('sys.stdout') as mock_stdout:
#             self.subject.change_state("new_state")
#             self.observer.deactivate()
#             self.observer.update(self.subject)
#             self.assertEqual(mock_stdout.getvalue(), "")

#     def test_error_handling(self):
#         with patch('sys.stdout', new_callable=unittest.mock.StringIO) as mock_stdout:
#             self.observer.activate()
#             with self.assertRaises(ValueError) as context:
#                 self.observer.update(None)  # Passing None should trigger an error in the observer
#             self.assertEqual(str(context.exception), "Subject cannot be None")

#     def test_pre_update_hook(self):
#         """ Test the execution of the pre-update hook. """
#         with patch('sys.stdout') as mock_stdout:
#             self.observer.activate()
#             self.observer.update(self.subject)
#             self.assertIn("Preparing to update AdvancedObserver.", mock_stdout.getvalue())

#     def test_post_update_hook(self):
#         """ Test the execution of the post-update hook. """
#         with patch('sys.stdout') as mock_stdout:
#             self.observer.activate()
#             self.observer.update(self.subject)
#             self.assertIn("AdvancedObserver update complete.", mock_stdout.getvalue())

# def main():
```

```
#     unittest.main()

# if __name__ == '__main__':
#     main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/observer/__init__.py

from .observer import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/memento/memento.py

# memento.py
from datetime import datetime
from typing import List

class Memento:
    """
    The Memento interface provides a way to retrieve the memento's metadata,
    such as creation date or name. It doesn't expose the Originator's state.
    """
    def get_name(self) -> str:
        pass

    def get_date(self) -> str:
        pass

class ConcreteMemento(Memento):
    def __init__(self, state: str) -> None:
        self._state = state
        self._date = str(datetime.now())[:19]

    def get_state(self) -> str:
        return self._state

    def get_name(self) -> str:
        return f"{self._date} / ({self._state[0:9]}...)"

    def get_date(self) -> str:
        return self._date

class Originator:
    """
    The Originator holds an important state that can change over time.
    It defines methods for saving and restoring the state from a Memento.
    """
    _state = None

    def __init__(self, state: str) -> None:
        self._state = state
        print(f"Originator: My initial state is: {self._state}")

    def do_something(self) -> None:
        print("Originator: I'm doing something important.")
        self._state = f"state_{datetime.now().timestamp()}"
        print(f"Originator: and my state has changed to: {self._state}")

    def save(self) -> Memento:
        return ConcreteMemento(self._state)

    def restore(self, memento: Memento) -> None:
        self._state = memento.get_state()
        print(f"Originator: My state has changed to: {self._state}")

class Caretaker:
    """
    The Caretaker works with Mementos via the base Memento interface.
    It can store and restore the Originator's state.
    """
    def __init__(self, originator: Originator) -> None:
        self._mementos = []
        self._originator = originator

    def backup(self) -> None:
        print("\nCaretaker: Saving Originator's state...")
        self._mementos.append(self._originator.save())

    def undo(self) -> None:
        if not self._mementos:
            return

        memento = self._mementos.pop()
        print(f"Caretaker: Restoring state to: {memento.get_name()}")
        self._originator.restore(memento)

    def show_history(self) -> None:
        print("Caretaker: Here's the list of mementos:")
        for memento in self._mementos:
            print(memento.get_name())

# Example usage
if __name__ == "__main__":
    originator = Originator("Initial State")
    caretaker = Caretaker(originator)

    caretaker.backup()
    originator.do_something()

    caretaker.backup()
    originator.do_something()

    caretaker.backup()
    originator.do_something()

    caretaker.show_history()

    print("\nClient: Now, let's rollback!\n")
    caretaker.undo()
    print("\nClient: Once more!\n")
    caretaker.undo()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/memento/test_memento.py

import unittest
from unittest.mock import patch
from memento import Memento, ConcreteMemento, Originator, Caretaker

class TestMementoPattern(unittest.TestCase):
    def setUp(self):
        self.originator = Originator("Initial State")
        self.caretaker = Caretaker(self.originator)

    def test_memento_creation(self):
        """Test the creation of a memento and its properties."""
        memento = self.originator.save()
        self.assertIsInstance(memento, ConcreteMemento)
        self.assertTrue(memento.get_name().startswith("20"))  # Assuming current year
        self.assertTrue(memento.get_date().startswith("20"))  # Assuming current year

    def test_state_restoration(self):
        """Test the restoration of the state in the originator from a memento."""
        self.originator._state = "New State"
        memento = self.originator.save()
        self.originator._state = "Another State"
        self.originator.restore(memento)
        self.assertEqual(self.originator._state, "New State")

    def test_caretaker_memento_management(self):
        """Test the caretaker's ability to store and retrieve mementos."""
        self.caretaker.backup()
        self.caretaker.backup()
        self.assertEqual(len(self.caretaker._mementos), 2)

    def test_caretaker_undo_functionality(self):
        """Test the caretaker's undo functionality."""
        self.originator._state = "State A"
        self.caretaker.backup()
        self.originator._state = "State B"
        self.caretaker.backup()
        self.caretaker.undo()
        self.assertEqual(self.originator._state, "State A")

def main():
    unittest.main()

if __name__ == '__main__':
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/memento/__init__.py
```

```
from .memento import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/adapter/adapter.py

# adapter.py

class Target:
    """
    The Target defines the domain-specific interface used by the client code.
    """
    def request(self) -> str:
        return "Target: The default target's behavior."


class Adaptee:
    """
    The Adaptee contains some useful behavior, but its interface is incompatible
    with the existing client code. The Adaptee needs some adaptation before the
    client code can use it.
    """
    def specific_request(self) -> str:
        return ".eetpadA eht fo roivaheb laicepS"


# Inheritance-based Adapter
class AdapterInheritance(Target, Adaptee):
    """
    The Adapter makes the Adaptee's interface compatible with the Target's
    interface via multiple inheritance.
    """
    def request(self) -> str:
        return f"Adapter (Inheritance): (TRANSLATED) {self.specific_request()[::-1]}"


# Composition-based Adapter
class AdapterComposition(Target):
    """
    The Adapter makes the Adaptee's interface compatible with the Target's
    interface via composition.
    """
    def __init__(self, adaptee: Adaptee):
        self.adaptee = adaptee

    def request(self) -> str:
        return f"Adapter (Composition): (TRANSLATED) {self.adaptee.specific_request()[::-1]}"


def client_code(target: Target):
    """
    The client code supports all classes that follow the Target interface.
    """
    print(target.request(), end="\n\n")
```

```
if __name__ == "__main__":
    print("Client: I can work just fine with the Target objects:")
    target = Target()
    client_code(target)

    adaptee = Adaptee()
    print("Client: The Adaptee class has a weird interface. See, I don't understand it:")
    print(f"Adaptee: {adaptee.specific_request()}", end="\n\n")

    print("Client: But I can work with it via the Inheritance-based Adapter:")
    adapter_inheritance = AdapterInheritance()
    client_code(adapter_inheritance)

    print("Client: And also with the Composition-based Adapter:")
    adapter_composition = AdapterComposition(adaptee)
    client_code(adapter_composition)


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/adapter/__init__.py

from .adapter import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/adapter/test_adapter.py

# test_adapter.py

from adapter import Target, Adaptee, AdapterInheritance, AdapterComposition

def test_adapter_inheritance():
    """
    Test the inheritance-based Adapter.
    """
    adaptee = Adaptee()
    adapter = AdapterInheritance()

    assert adapter.request() == f"Adapter (Inheritance): (TRANSLATED) {adaptee.specific_request()[::-1]}", \
        "AdapterInheritance does not correctly adapt Adaptee"

    print("PASS: Inheritance-based Adapter test")

def test_adapter_composition():
    """
    Test the composition-based Adapter.
    """
    adaptee = Adaptee()
    adapter = AdapterComposition(adaptee)

    assert adapter.request() == f"Adapter (Composition): (TRANSLATED) {adaptee.specific_request()[::-1]}", \
        "AdapterComposition does not correctly adapt Adaptee"

    print("PASS: Composition-based Adapter test")

def main():
    """
    Main function to run the adapter tests.
    """
    print("Testing Adapter Pattern Implementations:")
    test_adapter_inheritance()
    test_adapter_composition()

if __name__ == "__main__":
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/prototype/__init__.py

from .prototype import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/prototype/test_prototype.py

import copy
from prototype import NovaComponent, SelfReferencingEntity

def test_shallow_copy(nova_component):
    shallow_copied_component = copy.copy(nova_component)
    print("Testing Shallow Copy:")

    # Modifying the shallow copy and testing its effect on the original
    shallow_copied_component.some_list_of_objects.append("new item")
    if "new item" in nova_component.some_list_of_objects:
        print("Shallow copy modification reflected in the original object.")
    else:
        print("Shallow copy modification not reflected in the original object.")

def test_deep_copy(nova_component):
    deep_copied_component = copy.deepcopy(nova_component)
    print("\nTesting Deep Copy:")

    # Modifying the deep copy and testing its effect on the original
    deep_copied_component.some_list_of_objects.append("new deep item")
    if "new deep item" in nova_component.some_list_of_objects:
        print("Deep copy modification reflected in the original object.")
    else:
        print("Deep copy modification not reflected in the original object.")

def main():
    list_of_objects = [1, {1, 2, 3}, [1, 2, 3]]
    circular_ref = SelfReferencingEntity()
    nova_component = NovaComponent(23, list_of_objects, circular_ref)
    circular_ref.set_parent(nova_component)

    test_shallow_copy(nova_component)
    test_deep_copy(nova_component)

if __name__ == "__main__":
    main()
```

```python
# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/prototype/prototype.py

import copy

class SelfReferencingEntity:
    def __init__(self):
        self.parent = None

    def set_parent(self, parent):
        self.parent = parent

class NovaComponent:
    def __init__(self, some_int, some_list_of_objects, some_circular_ref):
        self.some_int = some_int
        self.some_list_of_objects = some_list_of_objects
        self.some_circular_ref = some_circular_ref

    def __copy__(self):
        some_list_of_objects = copy.copy(self.some_list_of_objects)
        some_circular_ref = copy.copy(self.some_circular_ref)

        new = self.__class__(
            self.some_int, some_list_of_objects, some_circular_ref
        )
        new.__dict__.update(self.__dict__)

        return new

    def __deepcopy__(self, memo=None):
        if memo is None:
            memo = {}

        some_list_of_objects = copy.deepcopy(self.some_list_of_objects, memo)
        some_circular_ref = copy.deepcopy(self.some_circular_ref, memo)

        new = self.__class__(
            self.some_int, some_list_of_objects, some_circular_ref
        )
        new.__dict__ = copy.deepcopy(self.__dict__, memo)

        return new

# Example usage
if __name__ == "__main__":
    list_of_objects = [1, {1, 2, 3}, [1, 2, 3]]
    circular_ref = SelfReferencingEntity()
    nova_component = NovaComponent(23, list_of_objects, circular_ref)
    circular_ref.set_parent(nova_component)

    shallow_copied_component = copy.copy(nova_component)
    deep_copied_component = copy.deepcopy(nova_component)

    # Test and demonstrate the differences between shallow and deep copy
    # ... (Similar to the provided example code)


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/state/__init__.py

from .state import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/state/test_state.py

import unittest
from state import Context, ConcreteStateA, ConcreteStateB, ConcreteStateC, ConcreteStateD, StateContext

class TestStatePattern(unittest.TestCase):
    def test_initial_state(self):
        """Test the initial state setup in the context."""
        context = Context(ConcreteStateA())
        self.assertIsInstance(context.state, ConcreteStateA)

    def test_state_transition(self):
        """Test state transitions based on different conditions."""
        context = Context(ConcreteStateA())
        context.set_condition(True)  # Should transition to ConcreteStateB
        context.request()
        self.assertIsInstance(context.state, ConcreteStateB)

        context.set_condition(False)  # Should transition to ConcreteStateA
        context.request()
        self.assertIsInstance(context.state, ConcreteStateA)

    def test_special_case_handling(self):
        """Test the handling of special cases."""
        context = Context(ConcreteStateC())
        context.set_special_case(True)  # Should transition to ConcreteStateD
        context.request()
        self.assertIsInstance(context.state, ConcreteStateD)

        context.set_special_case(False)  # Should transition to ConcreteStateA
        context.request()
        self.assertIsInstance(context.state, ConcreteStateA)

    # Optional: Add a test for exception handling if relevant

def main():
    unittest.main()

if __name__ == '__main__':
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/state/state.py

from abc import ABC, abstractmethod
from dataclasses import dataclass
from typing import Callable, Optional
```

```
@dataclass
class StateContext:
    condition: bool = False
    special_case: bool = False

class State(ABC):
    @abstractmethod
    def handle(self, context: StateContext) -> None:
        pass

class ConcreteStateA(State):
    def handle(self, context: StateContext) -> None:
        print("State A handling context.")
        next_state = ConcreteStateB() if context.condition else ConcreteStateC()
        context.change_state(next_state)

class ConcreteStateB(State):
    def handle(self, context: StateContext) -> None:
        print("State B handling context.")
        context.change_state(ConcreteStateA())

class ConcreteStateC(State):
    def handle(self, context: StateContext) -> None:
        print("State C handling context.")
        next_state = ConcreteStateD() if context.special_case else ConcreteStateA()
        context.change_state(next_state)

class ConcreteStateD(State):
    def handle(self, context: StateContext) -> None:
        print("State D handling context (Special Case).")
        context.change_state(ConcreteStateA())

class Context:
    def __init__(self, state: State):
        self.state = state
        self.context_data = StateContext()

    def change_state(self, state: State) -> None:
        self.state = state

    def request(self) -> None:
        try:
            self.state.handle(self.context_data)
        except Exception as e:
            print(f"Error occurred: {e}")

    def set_condition(self, condition: bool) -> None:
        self.context_data.condition = condition

    def set_special_case(self, special_case: bool) -> None:
        self.context_data.special_case = special_case

# Example usage
```

```
if __name__ == "__main__":
    context = Context(ConcreteStateA())
    context.request()
    context.set_condition(True)
    context.request()
    context.set_special_case(True)
    context.request()


# File:
/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/composite/test_composite.py

# test_composite.py

from composite import Leaf, Composite

def test_leaf_operation():
    """
    Test the operation of a leaf component.
    """
    leaf = Leaf()
    assert leaf.operation() == "Leaf", "Leaf operation did not return expected result."
    print("PASS: Leaf operation test")

def test_composite_single_child():
    """
    Test a composite with a single child.
    """
    leaf = Leaf()
    composite = Composite()
    composite.add(leaf)

    assert composite.operation() == "Branch(Leaf)", "Composite operation with one child did not return expected result."
    print("PASS: Composite single child test")

def test_composite_multiple_children():
    """
    Test a composite with multiple children.
    """
    composite = Composite()
    composite.add(Leaf())
    composite.add(Leaf())

    assert composite.operation() == "Branch(Leaf+Leaf)", "Composite operation with multiple children did not return
expected result."
    print("PASS: Composite multiple children test")

def main():
    """
    Main function to run the composite pattern tests.
    """
    print("Testing Composite Pattern:")
    test_leaf_operation()
    test_composite_single_child()
    test_composite_multiple_children()

if __name__ == "__main__":
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/composite/__init__.py

from .composite import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/composite/composite.py

# composite.py

from __future__ import annotations
from abc import ABC, abstractmethod
from typing import List


class Component(ABC):
    """
    The base Component class declares common operations for both simple and
    complex objects of a composition.
    """

    def __init__(self) -> None:
        self._parent: Component = None

    @property
    def parent(self) -> Component:
        return self._parent

    @parent.setter
    def parent(self, parent: Component):
        self._parent = parent

    def add(self, component: Component) -> None:
        pass

    def remove(self, component: Component) -> None:
        pass

    def is_composite(self) -> bool:
        return False

    @abstractmethod
    def operation(self) -> str:
        pass


class Leaf(Component):
    """
    The Leaf class represents the end objects of a composition. A leaf can't
    have any children. Usually, it's the Leaf objects that do the actual work.
    """

    def operation(self) -> str:
        return "Leaf"


class Composite(Component):
    """
    The Composite class represents complex components that may have children.
    It delegates the actual work to their children and then 'sum-up' the result.
    """

    def __init__(self) -> None:
        super().__init__()
        self._children: List[Component] = []

    def add(self, component: Component) -> None:
        self._children.append(component)
        component.parent = self

    def remove(self, component: Component) -> None:
        self._children.remove(component)
        component.parent = None

    def is_composite(self) -> bool:
        return True

    def operation(self) -> str:
        results = [child.operation() for child in self._children]
        return f"Branch({'+'.join(results)})"


def client_code(component: Component) -> None:
    print(f"RESULT: {component.operation()}", end="")


def client_code2(component1: Component, component2: Component) -> None:
    if component1.is_composite():
        component1.add(component2)
    print(f"RESULT: {component1.operation()}", end="")


if __name__ == "__main__":
    simple = Leaf()
    print("Client: I've got a simple component:")
    client_code(simple)
    print("\n")

    tree = Composite()

    branch1 = Composite()
    branch1.add(Leaf())
    branch1.add(Leaf())
    branch2 = Composite()
    branch2.add(Leaf())

    tree.add(branch1)
    tree.add(branch2)

    print("Client: Now I've got a composite tree:")
    client_code(tree)
    print("\n")

    print("Client: I don't need to check the components classes even when managing the tree:")
    client_code2(tree, simple)


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/memoize/memoize.py

import functools
import time

def memoize(max_size=100, timeout=None):
    def memoize_decorator(func):
        cache = {}
        timestamps = {}
        @functools.wraps(func)
        def wrapper(*args, **kwargs):
            key = (args, frozenset(kwargs.items()))

            # Check for expired cache entries
            if timeout:
                for k in list(timestamps.keys()):
                    if time.time() - timestamps[k] > timeout:
                        del cache[k]
                        del timestamps[k]

            if key in cache:
                return cache[key]

            # If cache size limit is reached, remove the oldest item
            if len(cache) >= max_size:
                oldest_key = min(timestamps, key=timestamps.get)
                del cache[oldest_key]
                del timestamps[oldest_key]

            result = func(*args, **kwargs)
            cache[key] = result
            timestamps[key] = time.time()
            return result
        return wrapper
    return memoize_decorator

# Example usage
@memoize(max_size=50, timeout=300)  # 50 items in cache and 5 minutes timeout
def some_function(arg1, arg2, **kwargs):
    # Your function implementation
    return arg1 + arg2  # Replace with actual computation

print(some_function(3, 4, option='value'))



# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/memoize/test_memoize.py

import unittest
from unittest.mock import patch
from memoize import memoize
import time

class TestMemoizeDecorator(unittest.TestCase):
    def setUp(self):
        @memoize(max_size=2, timeout=1)
        def test_func(a, b):
            return a + b
        self.test_func = test_func

    def test_basic_memoization(self):
        """Test basic memoization functionality."""
        result1 = self.test_func(1, 2)
        result2 = self.test_func(1, 2)
        self.assertEqual(result1, result2)

    def test_cache_size_limit(self):
        """Test cache size limit handling."""
        self.test_func(1, 2)
        self.test_func(3, 4)
        self.test_func(5, 6)  # This should remove the oldest cache (1, 2)
        with patch('time.time', return_value=time.time() + 2):
            result = self.test_func(1, 2)  # Recalculate as it should be removed from cache
            self.assertEqual(result, 3)

    def test_timeout_handling(self):
        """Test timeout handling in the cache."""
        self.test_func(7, 8)
        with patch('time.time', return_value=time.time() + 2):
            result = self.test_func(7, 8)  # Recalculate as it should be expired
            self.assertEqual(result, 15)

def main():
    unittest.main()

if __name__ == '__main__':
    main()



# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/memoize/__init__.py

from .memoize import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/iterator/__init__.py
```

```
from .iterator import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/iterator/iterator.py

from collections.abc import Iterable, Iterator
from typing import Any, List

class AlphabeticalOrderIterator(Iterator):
    """
    Concrete Iterators implement various traversal algorithms.
    """
    _position: int = None
    _reverse: bool = False

    def __init__(self, collection: List[Any], reverse: bool = False) -> None:
        self._collection = collection
        self._reverse = reverse
        self._position = -1 if reverse else 0

    def __next__(self):
        try:
            value = self._collection[self._position]
            self._position += -1 if self._reverse else 1
        except IndexError:
            raise StopIteration()
        return value

class WordsCollection(Iterable):
    """
    Concrete Collections provide methods for retrieving fresh iterator instances.
    """
    def __init__(self, collection: List[Any] = []) -> None:
        self._collection = collection

    def __iter__(self) -> AlphabeticalOrderIterator:
        return AlphabeticalOrderIterator(self._collection)

    def get_reverse_iterator(self) -> AlphabeticalOrderIterator:
        return AlphabeticalOrderIterator(self._collection, True)

    def add_item(self, item: Any):
        self._collection.append(item)

# Example usage
if __name__ == "__main__":
    collection = WordsCollection()
    collection.add_item("First")
    collection.add_item("Second")
    collection.add_item("Third")

    print("Straight traversal:")
    for item in collection:
        print(item)
    print("\nReverse traversal:")
    for item in collection.get_reverse_iterator():
        print(item)


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/iterator/test_iterator.py

from collections.abc import Iterable, Iterator
from typing import Any, List

class AlphabeticalOrderIterator(Iterator):
    """
    Concrete Iterators implement various traversal algorithms.
    """
    _position: int = None
    _reverse: bool = False

    def __init__(self, collection: List[Any], reverse: bool = False) -> None:
        self._collection = collection
        self._reverse = reverse
        self._position = -1 if reverse else 0

    def __next__(self):
        try:
            value = self._collection[self._position]
            self._position += -1 if self._reverse else 1
        except IndexError:
            raise StopIteration()
        return value

class WordsCollection(Iterable):
    """
    Concrete Collections provide methods for retrieving fresh iterator instances.
    """
    def __init__(self, collection: List[Any] = []) -> None:
        self._collection = collection

    def __iter__(self) -> AlphabeticalOrderIterator:
        return AlphabeticalOrderIterator(self._collection)

    def get_reverse_iterator(self) -> AlphabeticalOrderIterator:
        return AlphabeticalOrderIterator(self._collection, True)

    def add_item(self, item: Any):
        self._collection.append(item)

def main():
    collection = WordsCollection()
    collection.add_item("First")
    collection.add_item("Second")
    collection.add_item("Third")

    print("Straight traversal:")
    for item in collection:
        print(item)

    print("\nReverse traversal:")
    for item in collection.get_reverse_iterator():
        print(item)

# Example usage
if __name__ == "__main__":
    main()



# File:
/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/chain_of_responsibility/__init__.py

from .chain_of_responsibility import *

# File:
/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/chain_of_responsibility/test_chain_
of_responsibility.py

from chain_of_responsibility import AIModelHandler, DataPreprocessingHandler, VisualizationHandler,
AbstractHandler

def test_individual_handler(handler: AbstractHandler, request: str):
    result = handler.handle(request)
    if result:
        print(f"  Handled by {handler.__class__.__name__}: {result}")
    else:
        print(f"  {handler.__class__.__name__} passed the request.")

def test_full_chain(chain_head: AbstractHandler, request: str):
    print(f"\nTesting full chain with request: {request}")
    result = chain_head.handle(request)
    if result:
        print(f"Handled by chain: {result}")
    else:
        print("Request was left unhandled by the full chain.")

def main():
    # Setting up individual handlers
    ai_model_handler = AIModelHandler()
    data_handler = DataPreprocessingHandler()
    visualization_handler = VisualizationHandler()

    # Building the chain
    ai_model_handler.set_next(data_handler).set_next(visualization_handler)

    # Testing individual handlers
    test_requests = ["Train", "Preprocess", "Visualize", "Deploy", "Unknown"]
    for request in test_requests:
        print(f"\nTesting AIModelHandler with request: {request}")
        test_individual_handler(ai_model_handler, request)
        print(f"\nTesting DataPreprocessingHandler with request: {request}")
        test_individual_handler(data_handler, request)
        print(f"\nTesting VisualizationHandler with request: {request}")
        test_individual_handler(visualization_handler, request)

    # Testing the full chain
    for request in test_requests:
        test_full_chain(ai_model_handler, request)

if __name__ == "__main__":
    main()



# File:
/Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/chain_of_responsibility/chain_of_r
esponsibility.py

from abc import ABC, abstractmethod
from typing import Any, Optional

class Handler(ABC):
    """
    The Handler interface declares methods for building the chain of handlers and executing requests.
    """
    @abstractmethod
    def set_next(self, handler: 'Handler') -> 'Handler':
        pass

    @abstractmethod
    def handle(self, request: Any) -> Optional[str]:
        pass

class AbstractHandler(Handler):
    """
    Default chaining behavior implementation.
    """
    _next_handler: Handler = None

    def set_next(self, handler: 'Handler') -> 'Handler':
        self._next_handler = handler
        return handler

    def handle(self, request: Any) -> Optional[str]:
        if self._next_handler:
            return self._next_handler.handle(request)
        return None

# Concrete Handlers
class AIModelHandler(AbstractHandler):
    def handle(self, request: Any) -> str:
        if request == "Train":
            return f"AIModelHandler: Training model with {request}"
        else:
            return super().handle(request)

class DataPreprocessingHandler(AbstractHandler):
    def handle(self, request: Any) -> str:
        if request == "Preprocess":
            return f"DataPreprocessingHandler: Preprocessing {request}"
        else:
            return super().handle(request)

class VisualizationHandler(AbstractHandler):
    def handle(self, request: Any) -> str:
        if request == "Visualize":
            return f"VisualizationHandler: Visualizing data with {request}"
        else:
            return super().handle(request)

# Client code example
def client_code(handler: Handler) -> None:
    for operation in ["Train", "Preprocess", "Visualize", "Deploy"]:
        print(f"\nClient: Requesting to {operation}")
        result = handler.handle(operation)
        if result:
            print(f"  {result}", end="")
        else:
            print(f"  {operation} was left unhandled.", end="")

# Example usage
if __name__ == "__main__":
    ai_model_handler = AIModelHandler()
    data_handler = DataPreprocessingHandler()
    visualization_handler = VisualizationHandler()

    ai_model_handler.set_next(data_handler).set_next(visualization_handler)

    print("Chain: AI Model > Data Preprocessing > Visualization")
    client_code(ai_model_handler)


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/visitor/__init__.py

from .visitor import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/visitor/visitor.py

from abc import ABC, abstractmethod

class Visitor(ABC):
    """
    The Visitor interface declares a set of visiting methods that correspond to
    element classes. The signature of a visiting method allows the visitor to
    identify the exact class of the element being visited.
    """
    @abstractmethod
    def visit_concrete_element_a(self, element):
        pass

    @abstractmethod
    def visit_concrete_element_b(self, element):
        pass

class ConcreteVisitor1(Visitor):
    def visit_concrete_element_a(self, element):
        print(f"{element.operation_a()} + ConcreteVisitor1")

    def visit_concrete_element_b(self, element):
        print(f"{element.operation_b()} + ConcreteVisitor1")

class ConcreteVisitor2(Visitor):
    def visit_concrete_element_a(self, element):
        print(f"{element.operation_a()} + ConcreteVisitor2")

    def visit_concrete_element_b(self, element):
        print(f"{element.operation_b()} + ConcreteVisitor2")

class Element(ABC):
    """
    The Element interface declares an `accept` method that should take a base
    visitor interface as an argument.
    """
    @abstractmethod
    def accept(self, visitor: Visitor):
        pass

class ConcreteElementA(Element):
    def accept(self, visitor: Visitor):
        visitor.visit_concrete_element_a(self)

    def operation_a(self):
        return "ConcreteElementA"

class ConcreteElementB(Element):
    def accept(self, visitor: Visitor):
        visitor.visit_concrete_element_b(self)

    def operation_b(self):
        return "ConcreteElementB"

# Example usage
if __name__ == "__main__":
    elements = [ConcreteElementA(), ConcreteElementB()]

    visitor1 = ConcreteVisitor1()
    for element in elements:
        element.accept(visitor1)

    visitor2 = ConcreteVisitor2()
    for element in elements:
        element.accept(visitor2)


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/visitor/test_visitor.py
```

```
import unittest
from unittest.mock import patch
from visitor import ConcreteVisitor1, ConcreteVisitor2, ConcreteElementA, ConcreteElementB

class TestConcreteVisitor1(unittest.TestCase):
    @patch('sys.stdout')
    def test_visit_concrete_element_a(self, mock_stdout):
        element_a = ConcreteElementA()
        visitor1 = ConcreteVisitor1()
        element_a.accept(visitor1)
        mock_stdout.write.assert_called_with("ConcreteElementA + ConcreteVisitor1\n")

    @patch('sys.stdout')
    def test_visit_concrete_element_b(self, mock_stdout):
        element_b = ConcreteElementB()
        visitor1 = ConcreteVisitor1()
        element_b.accept(visitor1)
        mock_stdout.write.assert_called_with("ConcreteElementB + ConcreteVisitor1\n")

class TestConcreteVisitor2(unittest.TestCase):
    @patch('sys.stdout')
    def test_visit_concrete_element_a(self, mock_stdout):
        element_a = ConcreteElementA()
        visitor2 = ConcreteVisitor2()
        element_a.accept(visitor2)
        mock_stdout.write.assert_called_with("ConcreteElementA + ConcreteVisitor2\n")

    @patch('sys.stdout')
    def test_visit_concrete_element_b(self, mock_stdout):
        element_b = ConcreteElementB()
        visitor2 = ConcreteVisitor2()
        element_b.accept(visitor2)
        mock_stdout.write.assert_called_with("ConcreteElementB + ConcreteVisitor2\n")

def main():
    unittest.main()

if __name__ == '__main__':
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/singleton/test_singleton.py

# test_singleton.py

from singleton import AIModelManager

def test_singleton_instance_creation():
    """
    Test that the Singleton instance is created only once.
    """
    print("Testing Singleton instance creation...")
    first_instance = AIModelManager()
    second_instance = AIModelManager()

    assert first_instance is second_instance, "Singleton instances are not the same"
    print("PASS: Singleton instance creation test")

def test_singleton_configuration_persistence():
    """
    Test that changes in configuration are reflected across all instances.
    """
    print("Testing Singleton configuration persistence...")
    manager = AIModelManager()
    initial_config = manager.get_config("response_length")

    # Change configuration
    manager.update_config("response_length", 512)

    # Create new instance and check if the configuration change is reflected
    new_manager = AIModelManager()
    new_config = new_manager.get_config("response_length")

    assert new_config == 512, "Configuration change is not reflected in the new instance"
    assert initial_config != new_config, "Initial and new configurations are the same"
    print("PASS: Singleton configuration persistence test")


def main():
    test_singleton_instance_creation()
    test_singleton_configuration_persistence()

if __name__ == "__main__":
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/singleton/__init__.py

from .singleton import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/singleton/singleton.py

from threading import Lock

class SingletonMeta(type):
    """
    Thread-safe implementation of Singleton for managing AI model configurations.
    """
    _instances = {}
    _lock: Lock = Lock()

    def __call__(cls, *args, **kwargs):
        with cls._lock:
            if cls not in cls._instances:
                instance = super().__call__(*args, **kwargs)
                cls._instances[cls] = instance
        return cls._instances[cls]
```

```
class AIModelManager(metaclass=SingletonMeta):
    def __init__(self):
        # Initialize with default configuration
        self.config = {
            "language_model": "GPT-3",
            "response_length": 128,
            "custom_behavior": {}
        }

    def update_config(self, key, value):
        self.config[key] = value

    def get_config(self, key):
        return self.config.get(key, None)

    def perform_ai_logic(self):
        # Method to perform AI-related operations
        pass

# Example of direct usage:
# ai_manager = AIModelManager()
# ai_manager.update_config("response_length", 256)
# print(ai_manager.get_config("response_length"))
# ai_manager.perform_ai_logic()

# Example of indirect usage:
def main():
    # Creating the Singleton instance
    ai_manager = AIModelManager()

    # Initial configuration
    print("Initial Configuration:", ai_manager.config)

    # Updating configuration in one part of the system
    ai_manager.update_config("response_length", 256)
    print("Updated Configuration after first change:", ai_manager.config)

    # Accessing the Singleton in another part of the system
    another_manager_instance = AIModelManager()
    print("Configuration accessed from a different part:", another_manager_instance.config)

    # Demonstrating that the configuration change is reflected across all instances
    another_manager_instance.update_config("language_model", "Custom AI Model")
    print("Configuration after updating from another part:", ai_manager.config)

if __name__ == "__main__":
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/command/test_command.py

from command import SimpleCommand, ComplexCommand, Receiver, Invoker
```

```
def test_simple_command():
    print("Testing SimpleCommand:")
    simple_command = SimpleCommand("Simple Operation")
    simple_command.execute()

def test_complex_command():
    print("\nTesting ComplexCommand:")
    receiver = Receiver()
    complex_command = ComplexCommand(receiver, "Data A", "Data B")
    complex_command.execute()

def test_invoker():
    print("\nTesting Invoker:")
    invoker = Invoker()
    receiver = Receiver()

    # Setting up SimpleCommand and ComplexCommand for the invoker
    invoker.set_on_start(SimpleCommand("Initialization"))
    invoker.set_on_finish(ComplexCommand(receiver, "Finalize Operation", "Clean Up"))

    invoker.do_something_important()

def main():
    test_simple_command()
    test_complex_command()
    test_invoker()

if __name__ == "__main__":
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/command/command.py

from abc import ABC, abstractmethod

class Command(ABC):
    """
    The Command interface declares a method for executing a command.
    """
    @abstractmethod
    def execute(self) -> None:
        pass

class SimpleCommand(Command):
    """
    Some commands can implement simple operations on their own.
    """
    def __init__(self, payload: str) -> None:
        self._payload = payload

    def execute(self) -> None:
        print(f"SimpleCommand: Doing something simple like printing ({self._payload})")

class ComplexCommand(Command):
    """
    Complex commands delegate operations to other objects, called 'receivers.'
    """
    def __init__(self, receiver: 'Receiver', a: str, b: str) -> None:
        self._receiver = receiver
        self._a = a
        self._b = b

    def execute(self) -> None:
        print("ComplexCommand: Delegating complex tasks to a receiver object")
        self._receiver.do_something(self._a)
        self._receiver.do_something_else(self._b)

class Receiver:
    """
    The Receiver class contains important business logic.
    """
    def do_something(self, a: str) -> None:
        print(f"Receiver: Working on ({a}).")

    def do_something_else(self, b: str) -> None:
        print(f"Receiver: Also working on ({b}).")

class Invoker:
    """
    The Invoker is associated with commands and sends requests to the command.
    """
    _on_start = None
    _on_finish = None

    def set_on_start(self, command: Command):
        self._on_start = command

    def set_on_finish(self, command: Command):
        self._on_finish = command

    def do_something_important(self) -> None:
        print("Invoker: Does anybody want something done before I begin?")
        if isinstance(self._on_start, Command):
            self._on_start.execute()

        print("\nInvoker: ...doing something really important...")

        print("\nInvoker: Does anybody want something done after I finish?")
        if isinstance(self._on_finish, Command):
            self._on_finish.execute()

# Example usage
if __name__ == "__main__":
    invoker = Invoker()
    invoker.set_on_start(SimpleCommand("Start operation"))
    receiver = Receiver()
    invoker.set_on_finish(ComplexCommand(receiver, "Send email", "Save report"))
    invoker.do_something_important()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/command/__init__.py

from .command import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/flyweight/flyweight.py

import json
from typing import Dict, List

class Flyweight:
    def __init__(self, shared_state: List[str]) -> None:
        self._shared_state = shared_state

    def operation(self, unique_state: List[str]) -> None:
        shared = json.dumps(self._shared_state)
        unique = json.dumps(unique_state)
        print(f"Flyweight: Shared ({shared}) and unique ({unique}) state.")

class FlyweightFactory:
    _flyweights: Dict[str, Flyweight] = {}

    def __init__(self, initial_flyweights: List[List[str]]) -> None:
        for state in initial_flyweights:
            self._flyweights[self.get_key(state)] = Flyweight(state)

    def get_key(self, state: List[str]) -> str:
        return "_".join(sorted(state))

    def get_flyweight(self, shared_state: List[str]) -> Flyweight:
        key = self.get_key(shared_state)
        if not self._flyweights.get(key):
            print("FlyweightFactory: Creating new flyweight.")
            self._flyweights[key] = Flyweight(shared_state)
        else:
            print("FlyweightFactory: Reusing existing flyweight.")
        return self._flyweights[key]

    def list_flyweights(self) -> None:
        print(f"FlyweightFactory: I have {len(self._flyweights)} flyweights:")
        for key in self._flyweights:
            print(key)

# Client code example
def add_ai_component_to_system(factory: FlyweightFactory, data: List[str]) -> None:
    flyweight = factory.get_flyweight(data[:-1])
    flyweight.operation(data)

# Example usage
if __name__ == "__main__":
    factory = FlyweightFactory([
        ["NeuralNet", "Classifier", "Image"],
        ["NeuralNet", "Regressor", "TimeSeries"]
    ])

    factory.list_flyweights()

    add_ai_component_to_system(factory, ["NeuralNet", "Classifier", "Image", "ImageSetA"])
    add_ai_component_to_system(factory, ["NeuralNet", "Classifier", "Audio", "AudioSetB"])

    factory.list_flyweights()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/flyweight/__init__.py

from .flyweight import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/flyweight/test_flyweight.py

from flyweight import FlyweightFactory, add_ai_component_to_system

def test_flyweight_pattern():
    # Creating a Flyweight Factory with some initial shared states
    factory = FlyweightFactory([
        ["NeuralNet", "Classifier", "Image"],
        ["NeuralNet", "Regressor", "TimeSeries"]
    ])

    # Listing initial flyweights
    print("Initial flyweights in the factory:")
    factory.list_flyweights()

    # Adding components and testing if flyweights are reused or newly created
    print("\nAdding a new AI component to the system:")
    add_ai_component_to_system(factory, ["NeuralNet", "Classifier", "Image", "DatasetX"])

    print("\nAdding another AI component to the system (should reuse flyweight):")
    add_ai_component_to_system(factory, ["NeuralNet", "Classifier", "Image", "DatasetY"])

    print("\nAdding a different AI component (should create new flyweight):")
    add_ai_component_to_system(factory, ["NeuralNet", "Classifier", "Audio", "DatasetZ"])

    # Listing final flyweights to verify the correct creation and reuse
    print("\nFinal flyweights in the factory:")
    factory.list_flyweights()

def main():
    test_flyweight_pattern()

if __name__ == "__main__":
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/builder/__init__.py

from .builder import *
```

```
# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/builder/builder.py

from abc import ABC, abstractmethod

# Builder Interface
class AIComponentBuilder(ABC):

    @property
    @abstractmethod
    def product(self):
        """Property to get the product."""
        pass

    @abstractmethod
    def add_ai_module(self):
        """Method to add an AI module to the product."""
        pass

    @abstractmethod
    def add_learning_capability(self):
        """Method to add learning capability to the product."""
        pass

    @abstractmethod
    def add_interaction_interface(self):
        """Method to add an interaction interface to the product."""
        pass

# Concrete Builder
class ConcreteAIComponentBuilder(AIComponentBuilder):
    def __init__(self):
        self.reset()

    def reset(self):
        """Reset the builder to start with a fresh product."""
        self._product = AIComponent()

    @property
    def product(self):
        """Retrieve the built product and reset the builder."""
        product = self._product
        self.reset()
        return product

    def add_ai_module(self):
        """Add an AI module to the product."""
        self._product.add("AI Module")

    def add_learning_capability(self):
        """Add learning capability to the product."""
        self._product.add("Learning Capability")

    def add_interaction_interface(self):
        """Add an interaction interface to the product."""
        self._product.add("Interaction Interface")

# Product Class
class AIComponent:
    def __init__(self):
        self.parts = []

    def add(self, part):
        """Add a part to the component."""
        self.parts.append(part)

    def list_parts(self):
        """List all parts of the component."""
        print(f"AI Component Parts: {', '.join(self.parts)}", end="")

# Director Class
class Director:
    def __init__(self):
        self._builder = None

    @property
    def builder(self):
        """Property to get and set the builder."""
        return self._builder

    @builder.setter
    def builder(self, builder):
        self._builder = builder

    def build_minimal_ai_component(self):
        """Build a minimal AI component."""
        self.builder.add_ai_module()

    def build_full_featured_ai_component(self):
        """Build a full-featured AI component."""
        self.builder.add_ai_module()
        self.builder.add_learning_capability()
        self.builder.add_interaction_interface()

# Client Code (optional here, might be in a separate test file)
if __name__ == "__main__":
    director = Director()
    builder = ConcreteAIComponentBuilder()
    director.builder = builder

    print("Building a minimal AI component:")
    director.build_minimal_ai_component()
    builder.product.list_parts()

    print("\n\nBuilding a full-featured AI component:")
    director.build_full_featured_ai_component()
    builder.product.list_parts()
    print("\n\nBuilding a custom AI component:")
    builder.add_ai_module()
    builder.add_interaction_interface()
    builder.product.list_parts()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/builder/test_builder.py

from builder import Director, ConcreteAIComponentBuilder

def test_minimal_ai_component(director, builder):
    print("Testing minimal AI component construction:")
    director.builder = builder
    director.build_minimal_ai_component()
    builder.product.list_parts()

def test_full_featured_ai_component(director, builder):
    print("\nTesting full-featured AI component construction:")
    director.builder = builder
    director.build_full_featured_ai_component()
    builder.product.list_parts()

def test_custom_ai_component(builder):
    print("\nTesting custom AI component construction:")
    builder.add_ai_module()
    builder.add_learning_capability()  # Adding only specific parts
    builder.product.list_parts()

def main():
    director = Director()
    builder = ConcreteAIComponentBuilder()

    test_minimal_ai_component(director, builder)
    test_full_featured_ai_component(director, builder)
    test_custom_ai_component(builder)

if __name__ == "__main__":
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/mediator/test_mediator.py

import unittest
from unittest.mock import patch
from mediator import Mediator, ConcreteMediator, BaseComponent, Component1, Component2

class TestMediatorPattern(unittest.TestCase):
    def setUp(self):
        self.component1 = Component1()
        self.component2 = Component2()
        self.mediator = ConcreteMediator(self.component1, self.component2)

    def test_mediator_initialization(self):
        """Test if the mediator is correctly set in the components."""
        self.assertEqual(self.component1.mediator, self.mediator)
        self.assertEqual(self.component2.mediator, self.mediator)

    def test_component_communication(self):
        """Test the communication between components via the mediator."""
        with patch('sys.stdout') as mock_stdout:
            self.component1.do_a()
            self.assertIn("Component 1 does A.", mock_stdout.getvalue())
            self.assertIn("Mediator reacts on A and triggers:", mock_stdout.getvalue())
            self.assertIn("Component 2 does C.", mock_stdout.getvalue())

            mock_stdout.reset()
            self.component2.do_d()
            self.assertIn("Component 2 does D.", mock_stdout.getvalue())
            self.assertIn("Mediator reacts on D and triggers:", mock_stdout.getvalue())
            self.assertIn("Component 1 does B.", mock_stdout.getvalue())
            self.assertIn("Component 2 does C.", mock_stdout.getvalue())

    def test_mediator_reactions(self):
        """Test mediator's reactions to different events."""
        with patch('sys.stdout') as mock_stdout:
            self.mediator.notify(self.component1, "A")
            self.assertIn("Mediator reacts on A and triggers:", mock_stdout.getvalue())
            self.assertIn("Component 2 does C.", mock_stdout.getvalue())

            mock_stdout.reset()
            self.mediator.notify(self.component2, "D")
            self.assertIn("Mediator reacts on D and triggers:", mock_stdout.getvalue())
            self.assertIn("Component 1 does B.", mock_stdout.getvalue())
            self.assertIn("Component 2 does C.", mock_stdout.getvalue())

def main():
    unittest.main()

if __name__ == '__main__':
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/mediator/mediator.py

from abc import ABC, abstractmethod

class Mediator(ABC):
    """
    The Mediator interface declares a method for components to notify the mediator about events.
    """
    @abstractmethod
    def notify(self, sender: object, event: str) -> None:
        pass

class ConcreteMediator(Mediator):
    def __init__(self, component1: 'Component1', component2: 'Component2') -> None:
        self._component1 = component1
        self._component1.mediator = self
        self._component2 = component2
        self._component2.mediator = self

    def notify(self, sender: object, event: str) -> None:
        if event == "A":
            print("Mediator reacts on A and triggers:")
            self._component2.do_c()
        elif event == "D":
            print("Mediator reacts on D and triggers:")
            self._component1.do_b()
            self._component2.do_c()

class BaseComponent:
    """
    Base Component class with a mediator.
    """
    def __init__(self, mediator: Mediator = None) -> None:
        self._mediator = mediator

    @property
    def mediator(self) -> Mediator:
        return self._mediator

    @mediator.setter
    def mediator(self, mediator: Mediator) -> None:
        self._mediator = mediator

class Component1(BaseComponent):
    def do_a(self) -> None:
        print("Component 1 does A.")
        self.mediator.notify(self, "A")

    def do_b(self) -> None:
        print("Component 1 does B.")
        self.mediator.notify(self, "B")

class Component2(BaseComponent):
    def do_c(self) -> None:
        print("Component 2 does C.")
        self.mediator.notify(self, "C")

    def do_d(self) -> None:
        print("Component 2 does D.")
        self.mediator.notify(self, "D")

# Example usage
if __name__ == "__main__":
    c1 = Component1()
    c2 = Component2()
    mediator = ConcreteMediator(c1, c2)

    print("Client triggers operation A.")
    c1.do_a()
    print("\nClient triggers operation D.")
    c2.do_d()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/mediator/__init__.py

from .mediator import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/factory/test_factory.py

from factory import BotFactory, ChatBot, DataAnalysisBot, ConcurrentBot

def test_bot_creation(factory):
    # Testing the creation of ChatBot
    print("Testing ChatBot Creation:")
    chat_bot = factory.create_bot("chat")
    print(chat_bot.perform_task())

    # Testing the creation of DataAnalysisBot
    print("\nTesting DataAnalysisBot Creation:")
    data_bot = factory.create_bot("data")
    print(data_bot.perform_task())

def test_dynamic_bot_registration(factory):
    # Dynamically registering and testing a new bot type
    class ResearchBot:
        def perform_task(self):
            return "ResearchBot performing research."

    factory.register_new_bot_type("research", ResearchBot)
    research_bot = factory.create_bot("research")
    print("\nTesting Dynamically Registered ResearchBot:")
    print(research_bot.perform_task())

def main():
    bot_factory = BotFactory()
    bot_factory.register_new_bot_type("chat", ChatBot)
    bot_factory.register_new_bot_type("data", DataAnalysisBot)
    bot_factory.register_new_bot_type("concurrent", lambda: ConcurrentBot(ChatBot()))

    test_bot_creation(bot_factory)
    test_dynamic_bot_registration(bot_factory)

if __name__ == "__main__":
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/factory/__init__.py

from .factory import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/factory/factory.py

from abc import ABC, abstractmethod
```

```
import threading

# Step 1: Dynamic Bot Factory Interface
class Factory(ABC):
    @abstractmethod
    def create_bot(self, bot_type):
        pass

    @abstractmethod
    def register_new_bot_type(self, bot_type, bot_creator):
        pass

# Step 2: Polymorphic Concrete Bot Factories
class BotFactory(Factory):
    def __init__(self):
        self.bot_creators = {}

    def create_bot(self, bot_type):
        return self.bot_creators[bot_type]()

    def register_new_bot_type(self, bot_type, bot_creator):
        self.bot_creators[bot_type] = bot_creator

# Step 3: Abstract Bot Interface
class AbstractBot(ABC):
    @abstractmethod
    def perform_task(self):
        pass

    @abstractmethod
    def learn_new_skill(self, skill):
        pass

# Step 4: Various AI Bots
class ChatBot(AbstractBot):
    def perform_task(self):
        return "ChatBot engaging in conversation."

    def learn_new_skill(self, skill):
        return f"ChatBot learning {skill}."

class DataAnalysisBot(AbstractBot):
    def perform_task(self):
        return "DataAnalysisBot analyzing data."

    def learn_new_skill(self, skill):
        return f"DataAnalysisBot learning {skill}."

# Step 5: Concurrency in Bots
class ConcurrentBot(AbstractBot):
    def __init__(self, bot):
        self.bot = bot
        self.lock = threading.Lock()
    def perform_task(self):
        with self.lock:
            return self.bot.perform_task()

    def learn_new_skill(self, skill):
        with self.lock:
            return self.bot.learn_new_skill(skill)

# Step 6: Client Code Demonstration
def client_code(factory: BotFactory):
    factory.register_new_bot_type("chat", ChatBot)
    factory.register_new_bot_type("data", DataAnalysisBot)

    chat_bot = factory.create_bot("chat")
    print(chat_bot.perform_task())

    # Dynamically registering a new bot type
    factory.register_new_bot_type("concurrent", lambda: ConcurrentBot(ChatBot()))
    concurrent_bot = factory.create_bot("concurrent")
    print(concurrent_bot.perform_task())

# Demonstration
if __name__ == "__main__":
    bot_factory = BotFactory()
    client_code(bot_factory)


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/strategy/test_strategy.py

import unittest
from unittest.mock import patch
from strategy import Context, ConcreteStrategyA, ConcreteStrategyB, Strategy, reverse_alphabetical

class TestStrategyPattern(unittest.TestCase):
    def test_concrete_strategy_a(self):
        """Test ConcreteStrategyA."""
        strategy = ConcreteStrategyA()
        data = ["e", "b", "d", "a", "c"]
        result = strategy.do_algorithm(data)
        self.assertEqual(result, ["a", "b", "c", "d", "e"])

    def test_concrete_strategy_b(self):
        """Test ConcreteStrategyB."""
        strategy = ConcreteStrategyB()
        data = ["e", "b", "d", "a", "c"]
        result = list(strategy.do_algorithm(data))
        self.assertEqual(result, ["e", "d", "c", "b", "a"])

    def test_context_with_different_strategies(self):
        """Test the context with different strategies."""
        context = Context(ConcreteStrategyA())
        data = ["e", "b", "d", "a", "c"]

        with patch('sys.stdout') as mock_stdout:
            context.do_some_business_logic()
            self.assertIn(','.join(sorted(data)), mock_stdout.getvalue())

        context.strategy = ConcreteStrategyB()
        with patch('sys.stdout') as mock_stdout:
            context.do_some_business_logic()
            self.assertIn(','.join(reversed(sorted(data))), mock_stdout.getvalue())

        context.strategy = Strategy(lambda data: reverse_alphabetical(data))
        with patch('sys.stdout') as mock_stdout:
            context.do_some_business_logic()
            self.assertIn(','.join(reversed(sorted(data))), mock_stdout.getvalue())

def main():
    unittest.main()

if __name__ == '__main__':
    main()


# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/strategy/__init__.py

from .strategy import *

# File: /Users/ctavolazzi/Code/WinfoNova/Nova_System_Git/NovaSystem/DesignPatterns/strategy/strategy.py

from abc import ABC, abstractmethod
from typing import List, Callable, Any
from dataclasses import dataclass

class Strategy(ABC):
    @abstractmethod
    def do_algorithm(self, data: List[str]) -> List[str]:
        pass

class ConcreteStrategyA(Strategy):
    def do_algorithm(self, data: List[str]) -> List[str]:
        return sorted(data)

class ConcreteStrategyB(Strategy):
    def do_algorithm(self, data: List[str]) -> List[str]:
        return reversed(sorted(data))

# Example of a functional strategy using a lambda function
reverse_alphabetical = lambda data: reversed(sorted(data))

@dataclass
class Context:
    strategy: Strategy

    def do_some_business_logic(self) -> None:
        result = self.strategy.do_algorithm(["a", "b", "c", "d", "e"])
        print(",".join(result))
```

```
# Example usage
if __name__ == "__main__":
    context = Context(ConcreteStrategyA())
    print("Client: Strategy is set to normal sorting.")
    context.do_some_business_logic()

    print("\nClient: Strategy is set to reverse sorting.")
    context.strategy = ConcreteStrategyB()
    context.do_some_business_logic()

    # Using a functional strategy
    print("\nClient: Strategy is set to functional reverse sorting.")
    context.strategy = Strategy(lambda data: reverse_alphabetical(data))
    context.do_some_business_logic()
```

```
=== Summary ===
Total Files Processed: 5
File Type 'File': 5 files
MIME Type 'text/x-python': 5 files
```

```
conftest.py - Type: File MIME: text/x-python Size: 118.00B | {} Modified: 2023-12-05 21:55:58
map_file_structure.py - Type: File MIME: text/x-python Size: 10.45KB | {} Modified: 2023-12-05 21:54:26
test_design_patterns.py - Type: File MIME: text/x-python Size: 4.76KB | {} Modified: 2023-12-05 21:40:52
README.md - Type: File MIME: Unknown Size: 2.75KB | {} Modified: 2023-12-05 06:18:25
copy_code_to_text_file.py - Type: File MIME: text/x-python Size: 11.64KB | {} Modified: 2023-12-05 22:35:20
stream_to_console.py - Type: File MIME: text/x-python Size: 8.57KB | {} Modified: 2023-12-05 21:54:24
/proxy/proxy.py - Type: File MIME: text/x-python Size: 1.36KB | {} Modified: 2023-12-05 18:13:15
/proxy/__init__.py - Type: File MIME: text/x-python Size: 20.00B | {} Modified: 2023-12-05 18:13:15
/proxy/test_proxy.py - Type: File MIME: text/x-python Size: 403.00B | {} Modified: 2023-12-05 18:13:15
/decorator/decorator.py - Type: File MIME: text/x-python Size: 1.73KB | {} Modified: 2023-12-05 18:13:15
/decorator/__init__.py - Type: File MIME: text/x-python Size: 24.00B | {} Modified: 2023-12-05 18:13:15
/decorator/test_decorator.py - Type: File MIME: text/x-python Size: 713.00B | {} Modified: 2023-12-05 18:13:15
/bridge/bridge.py - Type: File MIME: text/x-python Size: 2.50KB | {} Modified: 2023-12-05 07:06:16
/bridge/__init__.py - Type: File MIME: text/x-python Size: 21.00B | {} Modified: 2023-12-05 07:05:42
/bridge/test_bridge.py - Type: File MIME: text/x-python Size: 2.55KB | {} Modified: 2023-12-05 07:07:05
/design_patterns_code/extracted_code_20231205_223522.txt - Type: File MIME: text/plain Size: 45.63KB | {}
Modified: 2023-12-05 22:35:22
/template/__init__.py - Type: File MIME: text/x-python Size: 23.00B | {} Modified: 2023-12-05 18:13:15
/template/template.py - Type: File MIME: text/x-python Size: 2.22KB | {} Modified: 2023-12-05 18:13:15
/template/test_template.py - Type: File MIME: text/x-python Size: 1.51KB | {} Modified: 2023-12-05 18:13:15
/facade/facade.py - Type: File MIME: text/x-python Size: 1.26KB | {} Modified: 2023-12-05 18:13:15
/facade/__init__.py - Type: File MIME: text/x-python Size: 21.00B | {} Modified: 2023-12-05 18:13:15
/facade/test_facade.py - Type: File MIME: text/x-python Size: 367.00B | {} Modified: 2023-12-05 18:13:15
/observer/observer.py - Type: File MIME: text/x-python Size: 1.61KB | {} Modified: 2023-12-05 21:35:41
/observer/test_observer.py - Type: File MIME: text/x-python Size: 5.27KB | {} Modified: 2023-12-05 21:33:03
/observer/__init__.py - Type: File MIME: text/x-python Size: 23.00B | {} Modified: 2023-12-05 18:13:15
/memento/memento.py - Type: File MIME: text/x-python Size: 2.71KB | {} Modified: 2023-12-05 18:13:15
/memento/test_memento.py - Type: File MIME: text/x-python Size: 1.65KB | {} Modified: 2023-12-05 18:19:47
/memento/__init__.py - Type: File MIME: text/x-python Size: 22.00B | {} Modified: 2023-12-05 18:20:08
/adapter/adapter.py - Type: File MIME: text/x-python Size: 1.98KB | {} Modified: 2023-12-05 07:02:39
/adapter/__init__.py - Type: File MIME: text/x-python Size: 22.00B | {} Modified: 2023-12-05 07:02:28
/adapter/test_adapter.py - Type: File MIME: text/x-python Size: 1.05KB | {} Modified: 2023-12-05 07:03:34
/prototype/__init__.py - Type: File MIME: text/x-python Size: 24.00B | {} Modified: 2023-12-05 06:40:14
/prototype/test_prototype.py - Type: File MIME: text/x-python Size: 1.36KB | {} Modified: 2023-12-05 06:42:01
/prototype/prototype.py - Type: File MIME: text/x-python Size: 1.59KB | {} Modified: 2023-12-05 06:40:49
/state/__init__.py - Type: File MIME: text/x-python Size: 20.00B | {} Modified: 2023-12-05 18:13:15
/state/test_state.py - Type: File MIME: text/x-python Size: 1.41KB | {} Modified: 2023-12-05 18:22:41
/state/state.py - Type: File MIME: text/x-python Size: 1.92KB | {} Modified: 2023-12-05 18:13:15
/composite/test_composite.py - Type: File MIME: text/x-python Size: 1.23KB | {} Modified: 2023-12-05 07:10:33
/composite/__init__.py - Type: File MIME: text/x-python Size: 24.00B | {} Modified: 2023-12-05 07:08:41
/composite/composite.py - Type: File MIME: text/x-python Size: 2.56KB | {} Modified: 2023-12-05 07:09:33
/memoize/memoize.py - Type: File MIME: text/x-python Size: 1.29KB | {} Modified: 2023-12-05 18:13:15
/memoize/test_memoize.py - Type: File MIME: text/x-python Size: 1.25KB | {} Modified: 2023-12-05 18:22:28
/memoize/__init__.py - Type: File MIME: text/x-python Size: 22.00B | {} Modified: 2023-12-05 18:13:15
/iterator/__init__.py - Type: File MIME: text/x-python Size: 23.00B | {} Modified: 2023-12-05 18:13:15
/iterator/iterator.py - Type: File MIME: text/x-python Size: 1.55KB | {} Modified: 2023-12-05 18:13:15
/iterator/test_iterator.py - Type: File MIME: text/x-python Size: 1.58KB | {} Modified: 2023-12-05 18:13:15
/chain_of_responsibility/__init__.py - Type: File MIME: text/x-python Size: 38.00B | {} Modified: 2023-12-05
18:13:15
/chain_of_responsibility/test_chain_of_responsibility.py - Type: File MIME: text/x-python Size: 1.63KB | {} Modified:
2023-12-05 18:13:15
/chain_of_responsibility/chain_of_responsibility.py - Type: File MIME: text/x-python Size: 2.24KB | {} Modified:
2023-12-05 18:13:15
```

```
/visitor/__init__.py - Type: File MIME: text/x-python Size: 22.00B | {} Modified: 2023-12-05 18:23:10
/visitor/visitor.py - Type: File MIME: text/x-python Size: 1.81KB | {} Modified: 2023-12-05 18:13:15
/visitor/test_visitor.py - Type: File MIME: text/x-python Size: 1.40KB | {} Modified: 2023-12-05 18:13:15
/singleton/test_singleton.py - Type: File MIME: text/x-python Size: 1.33KB | {} Modified: 2023-12-05 06:54:17
/singleton/__init__.py - Type: File MIME: text/x-python Size: 24.00B | {} Modified: 2023-12-05 06:49:32
/singleton/singleton.py - Type: File MIME: text/x-python Size: 1.97KB | {} Modified: 2023-12-05 06:47:16
/file_tree/runs/run_20231205_223724/error_log.txt - Type: File MIME: text/plain Size: 0.00B | {} Modified: 2023-12-05 22:37:24
/file_tree/runs/run_20231205_223724/summary.txt - Type: File MIME: text/plain Size: 0.00B | {} Modified: 2023-12-05 22:37:24
/file_tree/runs/run_20231205_223724/file_structure.txt - Type: File MIME: text/plain Size: 0.00B | {} Modified: 2023-12-05 22:37:24
/file_tree/runs/run_20231205_215429/error_log.txt - Type: File MIME: text/plain Size: 0.00B | {} Modified: 2023-12-05 21:54:29
/file_tree/runs/run_20231205_215429/summary.txt - Type: File MIME: text/plain Size: 136.00B | {} Modified: 2023-12-05 21:54:29
/file_tree/runs/run_20231205_215429/file_structure.txt - Type: File MIME: text/plain Size: 7.62KB | {} Modified: 2023-12-05 21:54:29
/command/test_command.py - Type: File MIME: text/x-python Size: 884.00B | {} Modified: 2023-12-05 18:13:15
/command/command.py - Type: File MIME: text/x-python Size: 2.25KB | {} Modified: 2023-12-05 18:13:15
/command/__init__.py - Type: File MIME: text/x-python Size: 22.00B | {} Modified: 2023-12-05 18:13:15
/flyweight/flyweight.py - Type: File MIME: text/x-python Size: 1.86KB | {} Modified: 2023-12-05 18:13:15
/flyweight/__init__.py - Type: File MIME: text/x-python Size: 24.00B | {} Modified: 2023-12-05 18:13:15
/flyweight/test_flyweight.py - Type: File MIME: text/x-python Size: 1.18KB | {} Modified: 2023-12-05 18:13:15
/builder/__init__.py - Type: File MIME: text/x-python Size: 22.00B | {} Modified: 2023-12-05 06:38:45
/builder/builder.py - Type: File MIME: text/x-python Size: 2.94KB | {} Modified: 2023-12-05 06:38:16
/builder/test_builder.py - Type: File MIME: text/x-python Size: 993.00B | {} Modified: 2023-12-05 06:34:05
/mediator/test_mediator.py - Type: File MIME: text/x-python Size: 2.12KB | {} Modified: 2023-12-05 18:20:23
/mediator/mediator.py - Type: File MIME: text/x-python Size: 1.92KB | {} Modified: 2023-12-05 18:13:15
/mediator/__init__.py - Type: File MIME: text/x-python Size: 23.00B | {} Modified: 2023-12-05 18:13:15
/factory/test_factory.py - Type: File MIME: text/x-python Size: 1.19KB | {} Modified: 2023-12-05 06:35:02
/factory/__init__.py - Type: File MIME: text/x-python Size: 22.00B | {} Modified: 2023-12-05 06:28:09
/factory/factory.py - Type: File MIME: text/x-python Size: 2.11KB | {} Modified: 2023-12-05 06:25:57
/strategy/test_strategy.py - Type: File MIME: text/x-python Size: 1.61KB | {} Modified: 2023-12-05 18:22:55
/strategy/__init__.py - Type: File MIME: text/x-python Size: 23.00B | {} Modified: 2023-12-05 18:13:15
/strategy/strategy.py - Type: File MIME: text/x-python Size: 1.30KB | {} Modified: 2023-12-05 18:13:15
```

{% for message in messages %}

- **User:** {{ message.user_message }}
- **AI:** {{ message.ai_response }}

{% endfor %}

Type your message here... Send

```
NovaSystem/
    novagpt-env/
        pyvenv.cfg
    docs/
    src/
        NovaSystem_file_structure.txt
        main.py
        tests/
            nova-test.py
        utils/
            generate_file_structure.py
            your_full_file_structure.txt
```

```
NovaSystem/
│
├── __init__.py
|
├── src/
│   ├── api/
│   │   ├── controllers/
│   │   ├── middlewares/
│   │   ├── models/
│   │   ├── schemas/
│   │   ├── routers/
│   │   └── utils/
│   │
│   ├── cli/
│   │   ├── commands/
│   │   └── utils/
│   │
│   ├── core/
│   │   ├── __init__.py      # Core module entry point
│   │   ├── setup.py         # Setup script for core module
│   │   ├── logic.py         # Core application logic
│   │   ├── config.py        # Enhanced Configuration settings
│   │   ├── models.py        # Core data models
│   │   ├── security.py      # Security management
│   │   └── ai_utils.py      # AI and Machine Learning utilities
│   │
│   ├── gui/                 # Tkinter GUI related modules
│   │   └── main_gui.py      # GUI entry point
│   │
│   └── utils/               # General utilities and helpers
│      ├── logger.py         # Custom logging module
│      ├── db_connector.py  # Database connection utilities
│      └── performance.py   # Performance monitoring tools
│
├── tests/
│   ├── api/
│   ├── cli/
│   ├── gui/
│   └── core/
│
├── logs/                    # Log files directory
│   └── application.log      # Main log file
│
├── .env                     # Environment variables
├── .gitignore               # Git ignore rules
├── README.md                # Basic documentation
├── requirements.txt         # Dependency list
└── setup.py                 # Setup script for project
```

It will need to become:

```
novasystem/
│
```

```
├── __init__.py
├── main.py  # Entry point of the application
├── cli.py  # CLI related functions and commands
├── workspace.py  # Workspace management functions
├── settings.py  # Settings management functions
├── security.py  # Security-related functions
├── utils.py  # Utility functions
└── config.py  # Configuration related functions
```

```
╔═══════════════════════════════════════════════════════════════════════════════════════════════
════════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════╗
          _____  ___        ______      ___       ___        __        ________   ___  ___    ________
____________ _________  ____    ____
         (\"   \|"  \      /    " \   |"  \    /"  |     /""\      /"       ) |"  \/"  |  /"       ) ("    _   ")  /"     "| |"  \    /"  |
         |.\\   \    |   // ____  \   \   \  //  /     /    \    (:   \___/   \   \  /  (:   \___/   )__/  \\__/  (: ______)  \
\  //   |
         |: \.   \\  |  /  /    ) :)   \\  \/. ./     /' /\  \    \___  \      \\  \/    \___  \        \\_ /      \/    |    /\\  \/.
|
         |.  \    \. | (: (____/ //     \.    //     //  __'  \    __/  \\      /   /      __/  \\       |.  |      // ___)_  |: \.
|
         |    \    \ |  \        /       \\   /     /   /  \\  \  /" \   :)   /   /      /" \   :)      \:  |     (:      "| |.  \    /:  |
          \___|\____\)   \"_____/         \__/     (___/    \___)(_______/    |___/      (_______/         \__|
\________) |___|\__/|___|

╚═══════════════════════════════════════════════════════════════════════════════════════════════
════════════════════════════════════════════════════════════════════════════════════════════════
═══════════════════════════╝
```

```
src/
    conftest.py
    NovaSystem_file_structure.txt
    main.py
    tests/
        nova-test.py
        test_stc.py
    Nova/
        nova.py
    utils/
        __init__.py
        ascii_art_generator.py
        border_maker.py
        ascii_art_utils.py
        generate_file_structure.py
        stream_to_console.py
        your_full_file_structure.txt
    AI/
        AIJournal.py
        openai_guy.py
        huggingface_guy.py
        AIForum/
            __init__.py
            aiforum.py
        npc/
            npc_manager.py
            npc.py
            __init__.py
            character.py
            dialogue.py
    gui/
        main_window.py
        __init__.py
        npc_window.py
    demos/
        __init__.py
        demo_stc.py
    media/
        __init__.py
        ASCII_art/
            art.txt
            NovaSystem_Titles/
                title0.txt
```

```
accelerate==0.24.1
annotated-types==0.5.0
anyio==3.7.1
art==6.1
bitsandbytes==0.41.2.post2
cachetools==5.3.2
certifi==2023.11.17
charset-normalizer==3.3.2
click==8.1.7
colorama==0.4.6
distro==1.8.0
fastapi==0.104.1
filelock==3.13.1
fsspec==2023.10.0
google-api-core==2.14.0
google-api-python-client==2.108.0
google-auth==2.23.4
google-auth-httplib2==0.1.1
googleapis-common-protos==1.61.0
h11==0.14.0
httpcore==0.17.3
httplib2==0.22.0
httpx==0.24.1
huggingface-hub==0.19.4
idna==3.6
iniconfig==2.0.0
Jinja2==3.1.2
MarkupSafe==2.1.3
mpmath==1.3.0
networkx==3.2.1
numpy==1.26.2
openai==1.3.5
packaging==23.2
pluggy==1.3.0
protobuf==4.24.4
psutil==5.9.6
pyasn1==0.5.1
pyasn1-modules==0.3.0
pydantic==2.5.2
pydantic_core==2.14.5
pyparsing==3.1.1
pytest==7.4.3
python-dotenv==1.0.0
PyYAML==6.0.1
regex==2023.10.3
requests==2.31.0
rsa==4.9
safetensors==0.4.1
scipy==1.11.4
sniffio==1.3.0
starlette==0.27.0
sympy==1.12
tokenizers==0.15.0
```

```
torch==2.1.1
tqdm==4.66.1
transformers==4.35.2
typer==0.9.0
typing_extensions==4.8.0
uritemplate==4.1.1
urllib3==2.0.7
```

```
openai
python-dotenv
```

```
openai
python-dotenv
```

```
absl-py==2.0.0 ; python_full_version == "3.10.13" \
    --hash=sha256:9a28abb62774ae4e8edbe2dd4c49ffcd45a6a848952a5eccc6a49f3f0fc1e2f3 \
    --hash=sha256:d9690211c5fcfefcdd1a45470ac2b5c5acd45241c3af71eed96bc5441746c0d5
alembic==1.12.1 ; python_full_version == "3.10.13" \
    --hash=sha256:47d52e3dfb03666ed945becb723d6482e52190917fdb47071440cfdba05d92cb \
    --hash=sha256:bca5877e9678b454706347bc10b97cb7d67f300320fa5c3a94423e8266e2823f
amqp==5.2.0 ; python_full_version == "3.10.13" \
    --hash=sha256:827cb12fb0baa892aad844fd95258143bce4027fdac4fccddbc43330fd281637 \
    --hash=sha256:a1ecff425ad063ad42a486c902807d1482311481c8ad95a72694b2975e75f7fd
aniso8601==9.0.1 ; python_full_version == "3.10.13" \
    --hash=sha256:1d2b7ef82963909e93c4f24ce48d4de9e66009a21bf1c1e1c85bdd0812fe412f \
    --hash=sha256:72e3117667eedf66951bb2d93f4296a56b94b078a8a95905a052611fb3f1b973
annotated-types==0.6.0 ; python_full_version == "3.10.13" \
    --hash=sha256:0641064de18ba7a25dee8f96403ebc39113d0cb953a01429249d5c7564666a43 \
    --hash=sha256:563339e807e53ffd9c267e99fc6d9ea23eb8443c08f112651963e24e22f84a5d
anyio==3.7.1 ; python_full_version == "3.10.13" \
    --hash=sha256:44a3c9aba0f5defa43261a8b3efb97891f2bd7d804e0e1f56419befa1adfc780 \
    --hash=sha256:91dee416e570e92c64041bd18b900d1d6fa78dff7048769ce5ac5ddad004fbb5
art==6.1 ; python_full_version == "3.10.13" \
    --hash=sha256:159819c418001467f8d79616fa0814277deac97c8a363d1eb3e7c0a31526bfc3 \
    --hash=sha256:6ab3031e3b7710039e73497b0e750cadfe04d4c1279ce3a123500dbafb9e1b64
astunparse==1.6.3 ; python_full_version == "3.10.13" \
    --hash=sha256:5ad93a8456f0d084c3456d059fd9a92cce667963232cbf763eac3bc5b7940872 \
    --hash=sha256:c2652417f2c8b5bb325c885ae329bdf3f86424075c4fd1a128674bc6fba4b8e8
babel==2.13.1 ; python_full_version == "3.10.13" \
    --hash=sha256:33e0952d7dd6374af8dbf6768cc4ddf3ccfefc244f9986d4074704f2fbd18900 \
    --hash=sha256:7077a4984b02b6727ac10f1f7294484f737443d7e2e66c5e4380e41a3ae0b4ed
billiard==4.2.0 ; python_full_version == "3.10.13" \
    --hash=sha256:07aa978b308f334ff8282bd4a746e681b3513db5c9a514cbdd810cbbdc19714d \
    --hash=sha256:9a3c3184cb275aa17a732f93f65b20c525d3d9f253722d26a82194803ade5a2c
blinker==1.7.0 ; python_full_version == "3.10.13" \
    --hash=sha256:c3f865d4d54db7abc53758a01601cf343fe55b84c1de4e3fa910e420b438d5b9 \
    --hash=sha256:e6820ff6fa4e4d1d8e2747c2283749c3f547e4fee112b98555cdcdae32996182
cachetools==5.3.2 ; python_full_version == "3.10.13" \
    --hash=sha256:086ee420196f7b2ab9ca2db2520aca326318b68fe5ba8bc4d49cca91add450f2 \
    --hash=sha256:861f35a13a451f94e301ce2bec7cac63e881232ccce7ed67fab9b5df4d3beaa1
celery==5.3.6 ; python_full_version == "3.10.13" \
    --hash=sha256:870cc71d737c0200c397290d730344cc991d13a057534353d124c9380267aab9 \
    --hash=sha256:9da4ea0118d232ce97dff5ed4974587fb1c0ff5c10042eb15278487cdd27d1af
certifi==2023.11.17 ; python_full_version == "3.10.13" \
    --hash=sha256:9b469f3a900bf28dc19b8cfbf8019bf47f7fdd1a65a1d4ffb98fc14166beb4d1 \
    --hash=sha256:e036ab49d5b79556f99cfc2d9320b34cfbe5be05c5871b51de9329f0603b0474
charset-normalizer==3.3.2 ; python_full_version == "3.10.13" \
    --hash=sha256:06435b539f889b1f6f4ac1758871aae42dc3a8c0e24ac9e60c2384973ad73027 \
    --hash=sha256:06a81e93cd441c56a9b65d8e1d043daeb97a3d0856d177d5c90ba85acb3db087 \
    --hash=sha256:0a55554a2fa0d408816b3b5cedf0045f4b8e1a6065aec45849de2d6f3f8e9786 \
    --hash=sha256:0b2b64d2bb6d3fb9112bafa732def486049e63de9618b5843bcdd081d8144cd8 \
    --hash=sha256:10955842570876604d404661fbccbc9c7e684caf432c09c715ec38fbae45ae09 \
    --hash=sha256:122c7fa62b130ed55f8f285bfd56d5f4b4a5b503609d181f9ad85e55c89f4185 \
    --hash=sha256:1ceae2f17a9c33cb48e3263960dc5fc8005351ee19db217e9b1bb15d28c02574 \
    --hash=sha256:1d3193f4a680c64b4b6a9115943538edb896edc190f0b222e73761716519268e \
    --hash=sha256:1f79682fbe303db92bc2b1136016a38a42e835d932bab5b3b1bfcfbf0640e519 \
    --hash=sha256:2127566c664442652f024c837091890cb1942c30937add288223dc895793f898 \
    --hash=sha256:22afcb9f253dac0696b5a4be4a1c0f8762f8239e21b99680099abd9b2b1b2269 \
    --hash=sha256:25baf083bf6f6b341f4121c2f3c548875ee6f5339300e08be3f2b2ba1721cdd3 \
    --hash=sha256:2e81c7b9c8979ce92ed306c249d46894776a909505d8f5a4ba55b14206e3222f \
    --hash=sha256:3287761bc4ee9e33561a7e058c72ac0938c4f57fe49a09eae428fd88aafe7bb6 \
    --hash=sha256:34d1c8da1e78d2e001f363791c98a272bb734000fcef47a491c1e3b0505657a8 \
    --hash=sha256:37e55c8e51c236f95b033f6fb391d7d7970ba5fe7ff453dad675e88cf303377a \
    --hash=sha256:3d47fa203a7bd9c5b6cee4736ee84ca03b8ef23193c0d1ca99b5089f72645c73 \
    --hash=sha256:3e4d1f6587322d2788836a99c69062fbb091331ec940e02d12d179c1d53e25fc \
    --hash=sha256:42cb296636fcc8b0644486d15c12376cb9fa75443e00fb25de0b8602e64c1714 \
    --hash=sha256:45485e01ff4d3630ec0d9617310448a8702f70e9c01906b0d0118bdf9d124cf2 \
    --hash=sha256:4a78b2b446bd7c934f5dcedc588903fb2f5eec172f3d29e52a9096a43722adfc \
    --hash=sha256:4ab2fe47fae9e0f9dee8c04187ce5d09f48eabe611be8259444906793ab7cbce \
    --hash=sha256:4d0d1650369165a14e14e1e47b372cfcb31d6ab44e6e33cb2d4e57265290044d \
    --hash=sha256:549a3a73da901d5bc3ce8d24e0600d1fa85524c10287f6004fbab87672bf3e1e \
    --hash=sha256:55086ee1064215781fff39a1af09518bc9255b50d6333f2e4c74ca09fac6a8f6 \
    --hash=sha256:572c3763a264ba47b3cf708a44ce965d98555f618ca42c926a9c1616d8f34269 \
    --hash=sha256:573f6eac48f4769d667c4442081b1794f52919e7edada77495aaed9236d13a96 \
    --hash=sha256:5b4c145409bef602a690e7cfad0a15a55c13320ff7a3ad7ca59c13bb8ba4d45d \
    --hash=sha256:6463effa3186ea09411d50efc7d85360b38d5f09b870c48e4600f63af490e56a \
    --hash=sha256:65f6f63034100ead094b8744b3b97965785388f308a64cf8d7c34f2f2e5be0c4 \
    --hash=sha256:663946639d296df6a2bb2aa51b60a2454ca1cb29835324c640dafb5ff2131a77 \
    --hash=sha256:6897af51655e3691ff853668779c7bad41579facacf5fd7253b0133308cf000d \
    --hash=sha256:68d1f8a9e9e37c1223b656399be5d6b448dea850bed7d0f87a8311f1ff3dabb0 \
    --hash=sha256:6ac7ffc7ad6d040517be39eb591cac5ff87416c2537df6ba3cba3bae290c0fed \
    --hash=sha256:6b3251890fff30ee142c44144871185dbe13b11bab478a88887a639655be1068 \
    --hash=sha256:6c4caeef8fa63d06bd437cd4bdcf3ffefe6738fb1b25951440d80dc7df8c03ac \
    --hash=sha256:6ef1d82a3af9d3eecdba2321dc1b3c238245d890843e040e41e470ffa64c3e25 \
    --hash=sha256:753f10e867343b4511128c6ed8c82f7bec3bd026875576dfd88483c5c73b2fd8 \
    --hash=sha256:7cd13a2e3ddeed6913a65e66e94b51d80a041145a026c27e6bb76c31a853c6ab \
    --hash=sha256:7ed9e526742851e8d5cc9e6cf41427dfc6068d4f5a3bb03659444b4cabf6bc26 \
    --hash=sha256:7f04c839ed0b6b98b1a7501a002144b76c18fb1c1850c8b98d458ac269e26ed2 \
    --hash=sha256:802fe99cca7457642125a8a88a084cef28ff0cf9407060f7b93dca5aa25480db \
    --hash=sha256:80402cd6ee291dcb72644d6eac93785fe2c8b9cb30893c1af5b8fdd753b9d40f \
    --hash=sha256:8465322196c8b4d7ab6d1e049e4c5cb460d0394da4a27d23cc242fbf0034b6b5 \
    --hash=sha256:86216b5cee4b06df986d214f664305142d9c76df9b6512be2738aa72a2048f99 \
    --hash=sha256:87d1351268731db79e0f8e745d92493ee2841c974128ef629dc518b937d9194c \
    --hash=sha256:8bdb58ff7ba23002a4c5808d608e4e6c687175724f54a5dade5fa8c67b604e4d \
    --hash=sha256:8c622a5fe39a48f78944a87d4fb8a53ee07344641b0562c540d840748571b811 \
    --hash=sha256:8d756e44e94489e49571086ef83b2bb8ce311e730092d2c34ca8f7d925cb20aa \
    --hash=sha256:8f4a014bc36d3c57402e2977dada34f9c12300af536839dc38c0beab8878f38a \
    --hash=sha256:9063e24fdb1e498ab71cb7419e24622516c4a04476b17a2dab57e8baa30d6e03 \
    --hash=sha256:90d558489962fd4918143277a773316e56c72da56ec7aa3dc3dbbe20fdfed15b \
    --hash=sha256:923c0c831b7cfcb071580d3f46c4baf50f174be571576556269530f4bbd79d04 \
    --hash=sha256:95f2a5796329323b8f0512e09dbb7a1860c46a39da62ecb2324f116fa8fdc85c \
    --hash=sha256:96b02a3dc4381e5494fad39be677abcb5e6634bf7b4fa83a6dd3112607547001 \
    --hash=sha256:9f96df6923e21816da7e0ad3fd47dd8f94b2a5ce594e00677c0013018b813458 \
    --hash=sha256:a10af20b82360ab00827f916a6058451b723b4e65030c5a18577c8b2de5b3389 \
    --hash=sha256:a50aebfa173e157099939b17f18600f72f84eed3049e743b68ad15bd69b6bf99 \
    --hash=sha256:a981a536974bbc7a512cf44ed14938cf01030a99e9b3a06dd59578882f06f985 \
    --hash=sha256:a9a8e9031d613fd2009c182b69c7b2c1ef8239a0efb1df3f7c8da66d5dd3d537 \
    --hash=sha256:ae5f4161f18c61806f411a13b0310bea87f987c7d2ecdbdaad0e94eb2e404238 \
    --hash=sha256:aed38f6e4fb3f5d6bf81bfa990a07806be9d83cf7bacef998ab1a9bd660a581f \
    --hash=sha256:b01b88d45a6fcb69667cd6d2f7a9aeb4bf53760d7fc536bf679ec94fe9f3ff3d \
    --hash=sha256:b261ccdec7821281dade748d088bb6e9b69e6d15b30652b74cbbac25e280b796 \
    --hash=sha256:b2b0a0c0517616b6869869f8c581d4eb2dd83a4d79e0ebcb7d373ef9956aeb0a \
    --hash=sha256:b4a23f61ce87adf89be746c8a8974fe1c823c891d8f86eb218bb957c924bb143 \
    --hash=sha256:bd8f7df7d12c2db9fab40bdd87a7c09b1530128315d047a086fa3ae3435cb3a8 \
    --hash=sha256:beb58fe5cdb101e3a055192ac291b7a21e3b7ef4f67fa1d74e331a7f2124341c \
    --hash=sha256:c002b4ffc0be611f0d9da932eb0f704fe2602a9a949d1f738e4c34c75b0863d5 \
    --hash=sha256:c083af607d2515612056a31f0a8d9e0fcb5876b7bfc0abad3ecd275bc4ebc2d5 \
    --hash=sha256:c180f51afb394e165eafe4ac2936a14bee3eb10debc9d9e4db8958fe36afe711 \
    --hash=sha256:c235ebd9baae02f1b77bcea61bce332cb4331dc3617d254df3323aa01ab47bd4 \
    --hash=sha256:cd70574b12bb8a4d2aaa0094515df2463cb429d8536cfb6c7ce983246983e5a6 \
    --hash=sha256:d0eccceffcb53201b5bfebb52600a5fb483a20b61da9dbc885f8b103cbe7598c \
    --hash=sha256:d965bba47ddeec8cd560687584e88cf699fd28f192ceb452d1d7ee807c5597b7 \
    --hash=sha256:db364eca23f876da6f9e16c9da0df51aa4f104a972735574842618b8c6d999d4 \
    --hash=sha256:ddbb2551d7e0102e7252db79ba445cdab71b26640817ab1e3e3648dad515003b \
    --hash=sha256:deb6be0ac38ece9ba87dea880e438f25ca3eddfac8b002a2ec3d9183a454e8ae \
    --hash=sha256:e06ed3eb3218bc64786f7db41917d4e686cc4856944f53d5bdf83a6884432e12 \
    --hash=sha256:e27ad930a842b4c5eb8ac0016b0a54f5aebbe679340c26101df33424142c143c \
    --hash=sha256:e537484df0d8f426ce2afb2d0f8e1c3d0b114b83f8850e5f2fbea0e797bd82ae \
    --hash=sha256:eb00ed941194665c332bf8e078baf037d6c35d7c4f3102ea2d4f16ca94a26dc8 \
    --hash=sha256:eb6904c354526e758fda7167b33005998fb68c46fbc10e013ca97f21ca5c8887 \
    --hash=sha256:eb8821e09e916165e160797a6c17edda0679379a4be5c716c260e836e122f54b \
    --hash=sha256:efcb3f6676480691518c177e3b465bcddf57cea040302f9f4e6e191af91174d4 \
    --hash=sha256:f27273b60488abe721a075bcca6d7f3964f9f6f067c8c4c605743023d7d3944f \
    --hash=sha256:f30c3cb33b24454a82faecaf01b19c18562b1e89558fb6c56de4d9118a032fd5 \
    --hash=sha256:fb69256e180cb6c8a894fee62b3afebae785babc1ee98b81cdf68bbca1987f33 \
    --hash=sha256:fd1abc0d89e30cc4e02e4064dc67fcc51bd941eb395c502aac3ec19fab46b519 \
    --hash=sha256:ff8fa367d09b717b2a17a052544193ad76cd49979c805768879cb63d9ca50561
click-didyoumean==0.3.0 ; python_full_version == "3.10.13" \
    --hash=sha256:a0713dc7a1de3f06bc0df5a9567ad19ead2d3d5689b434768a6145bff77c0667 \
    --hash=sha256:f184f0d851d96b6d29297354ed981b7dd71df7ff500d82fa6d11f0856bee8035
click-plugins==1.1.1 ; python_full_version == "3.10.13" \
    --hash=sha256:46ab999744a9d831159c3411bb0c79346d94a444df9a3a3742e9ed63645f264b \
    --hash=sha256:5d262006d3222f5057fd81e1623d4443e41dcda5dc815c06b442aa3c02889fc8
click-repl==0.3.0 ; python_full_version == "3.10.13" \
    --hash=sha256:17849c23dba3d667247dc4defe1757fff98694e90fe37474f3feebb69ced26a9 \
    --hash=sha256:fb7e06deb8da8de86180a33a9da97ac316751c094c6899382da7feeeeb51b812
click==8.1.7 ; python_full_version == "3.10.13" \
    --hash=sha256:ae74fb96c20a0277a1d615f1e4d73c8414f5a98db8b799a7931d1582f3390c28 \
    --hash=sha256:ca9853ad459e787e2192211578cc907e7594e294c7ccc834310722b41b9ca6de
colorama==0.4.6 ; python_full_version == "3.10.13" \
    --hash=sha256:08695f5cb7ed6e0531a20572697297273c47b8cae5a63ffc6d6ed5c201be6e44 \
    --hash=sha256:4f1d9991f5acc0ca119f9d443620b77f9d6b33703e51011c16baf57afb285fc6
distro==1.8.0 ; python_full_version == "3.10.13" \
    --hash=sha256:02e111d1dc6a50abb8eed6bf31c3e48ed8b0830d1ea2a1b78c61765c2513fdd8 \
    --hash=sha256:99522ca3e365cac527b44bde033f64c6945d90eb9f769703caaec52b09bbd3ff
exceptiongroup==1.2.0 ; python_full_version == "3.10.13" \
    --hash=sha256:4bfd3996ac73b41e9b9628b04e079f193850720ea5945fc96a08633c66912f14 \
    --hash=sha256:91f5c769735f051a4290d52edd0858999b57e5876e9f85937691bd4c9fa3ed68
fastapi==0.104.1 ; python_full_version == "3.10.13" \
    --hash=sha256:752dc31160cdbd0436bb93bad51560b57e525cbb1d4bbf6f4904ceee75548241 \
    --hash=sha256:e5e4540a7c5e1dcfbbcf5b903c234feddcdcd881f191977a1c5dfd917487e7ae
filelock==3.13.1 ; python_full_version == "3.10.13" \
    --hash=sha256:521f5f56c50f8426f5e03ad3b281b490a87ef15bc6c526f168290f0c7148d44e \
    --hash=sha256:57dbda9b35157b05fb3e58ee91448612eb674172fab98ee235ccb0b5bee19a1c
flask-babelex==0.9.4 ; python_full_version == "3.10.13" \
    --hash=sha256:39a59ccee9386a9d52d80b9101224402036aedc2c7873b11deef6e4e21cace27 \
    --hash=sha256:f744d0557cb04cafed733cfa96e7373b46263d4cf79a2c5988c65085f360d873
flask-jwt-extended==4.5.3 ; python_full_version == "3.10.13" \
    --hash=sha256:061ef3d25ed5743babe4964ab38f36d870e6d2fd8a126bab5d77ddef8a01932b \
    --hash=sha256:eaec42af107dcb919785a4b3766c09ffba9f286b92a8d58603933f28fd4db6a3
flask-login==0.6.3 ; python_full_version == "3.10.13" \
    --hash=sha256:5e23d14a607ef12806c699590b89d0f0e0d67baeec599d75947bf9c147330333 \
    --hash=sha256:849b25b82a436bf830a054e74214074af59097171562ab10bfa999e6b78aae5d
flask-mail==0.9.1 ; python_full_version == "3.10.13" \
    --hash=sha256:22e5eb9a940bf407bcf30410ecc3708f3c56cc44b29c34e1726fe85006935f41
flask-migrate==4.0.5 ; python_full_version == "3.10.13" \
    --hash=sha256:613a2df703998e78716cace68cd83972960834424457f5b67f56e74fff950aef \
    --hash=sha256:d3f437a8b5f3849d1bb1b60e1b818efc564c66e3fefe90b62e5db08db295e1b1
flask-principal==0.4.0 ; python_full_version == "3.10.13" \
    --hash=sha256:f5d6134b5caebfdbb86f32d56d18ee44b080876a27269560a96ea35f75c99453
flask-restful==0.3.10 ; python_full_version == "3.10.13" \
    --hash=sha256:1cf93c535172f112e080b0d4503a8d15f93a48c88bdd36dd87269bdaf405051b \
    --hash=sha256:fe4af2ef0027df8f9b4f797aba20c5566801b6ade995ac63b588abf1a59cec37
flask-security==3.0.0 ; python_full_version == "3.10.13" \
    --hash=sha256:d61daa5f5a48f89f30f50555872bdf581b2c65804668b0313345cd7beff26432 \
    --hash=sha256:ef837c03558db41335c8dabd16ae4977af0a5ef0c2cdecf738e33ef5202ce489
flask-sqlalchemy==3.1.1 ; python_full_version == "3.10.13" \
    --hash=sha256:4ba4be7f419dc72f4efd8802d69974803c37259dd42f3913b0dcf75c9447e0a0 \
    --hash=sha256:e4b68bb881802dda1a7d878b2fc84c06d1ee57fb40b874d3dc97dabfa36b8312
flask-swagger-ui==4.11.1 ; python_full_version == "3.10.13" \
    --hash=sha256:a370199a780d678b32e38f1be10d4d81efa0ee63e9fe2fb766ff1a4b6c37dac8 \
    --hash=sha256:c951928fe4592d3561b543e0e1ca32703f55d3474de86c894a9d27f795d96c83
flask-wtf==1.2.1 ; python_full_version == "3.10.13" \
    --hash=sha256:8bb269eb9bb46b87e7c8233d7e7debdf1f8b74bf90cc1789988c29b37a97b695 \
    --hash=sha256:fa6793f2fb7e812e0fe9743b282118e581fb1b6c45d414b8af05e659bd653287
flask==3.0.0 ; python_full_version == "3.10.13" \
    --hash=sha256:21128f47e4e3b9d597a3e8521a329bf56909b690fcc3fa3e477725aa81367638 \
    --hash=sha256:cfadcdb638b609361d29ec22360d6070a77d7463dcb3ab08d2c2f2f168845f58
flatbuffers==23.5.26 ; python_full_version == "3.10.13" \
    --hash=sha256:9ea1144cac05ce5d86e2859f431c6cd5e66cd9c78c558317c7955fb8d4c78d89 \
    --hash=sha256:c0ff356da363087b915fde4b8b45bdda73432fc17cddb3c8157472eab1422ad1
fsspec==2023.10.0 ; python_full_version == "3.10.13" \
    --hash=sha256:330c66757591df346ad3091a53bd907e15348c2ba17d63fd54f5c39c4457d2a5 \
    --hash=sha256:346a8f024efeb749d2a5fca7ba8854474b1ff9af7c3faaf636a4548781136529
gast==0.5.4 ; python_full_version == "3.10.13" \
    --hash=sha256:6fc4fa5fa10b72fb8aab4ae58bcb023058386e67b6fa2e3e34cec5c769360316 \
    --hash=sha256:9c270fe5f4b130969b54174de7db4e764b09b4f7f67ccfc32480e29f78348d97
google-auth-oauthlib==1.1.0 ; python_full_version == "3.10.13" \
    --hash=sha256:089c6e587d36f4803ac7e0720c045c6a8b1fd1790088b8424975b90d0ee61c12 \
    --hash=sha256:83ea8c3b0881e453790baff4448e8a6112ac8778d1de9da0b68010b843937afb
google-auth==2.23.4 ; python_full_version == "3.10.13" \
    --hash=sha256:79905d6b1652187def79d491d6e23d0cbb3a21d3c7ba0dbaa9c8a01906b13ff3 \
    --hash=sha256:d4bbc92fe4b8bfd2f3e8d88e5ba7085935da208ee38a134fc280e7ce682a05f2
google-pasta==0.2.0 ; python_full_version == "3.10.13" \
    --hash=sha256:4612951da876b1a10fe3960d7226f0c7682cf901e16ac06e473b267a5afa8954 \
    --hash=sha256:b32482794a366b5366a32c92a9a9201b107821889935a02b3e51f6b432ea84ed \
    --hash=sha256:c9f2c8dfc8f96d0d5808299920721be30c9eec37f2389f28904f454565c8a16e
grafana==0.0.1 ; python_full_version == "3.10.13" \
    --hash=sha256:45cf2b43c4f8194369de4c077084b982b1be06bbae25f5a6ad1d1bcf3af8b919 \
    --hash=sha256:a6ad951495bb03f628c849eda3aeb9bfd29870c9fbe179f4cb2dfaea3218ac8a
greenlet==3.0.1 ; (platform_machine == "aarch64" or platform_machine == "ppc64le" or platform_machine ==
"x86_64" or platform_machine == "amd64" or platform_machine == "AMD64" or platform_machine == "win32" or
platform_machine == "WIN32") and python_full_version == "3.10.13" \
    --hash=sha256:0a02d259510b3630f330c86557331a3b0e0c79dac3d166e449a39363beaae174 \
    --hash=sha256:0b6f9f8ca7093fd4433472fd99b5650f8a26dcd8ba410e14094c1e44cd3ceddd \
    --hash=sha256:100f78a29707ca1525ea47388cec8a049405147719f47ebf3895e7509c6446aa \
    --hash=sha256:1757936efea16e3f03db20efd0cd50a1c86b06734f9f7338a90c4ba85ec2ad5a \
    --hash=sha256:19075157a10055759066854a973b3d1325d964d498a805bb68a1f9af4aaef8ec \
    --hash=sha256:19bbdf1cce0346ef7341705d71e2ecf6f41a35c311137f29b8a2dc2341374565 \
    --hash=sha256:20107edf7c2c3644c67c12205dc60b1bb11d26b2610b276f97d666110d1b511d \
    --hash=sha256:22f79120a24aeeae2b4471c711dcf4f8c736a2bb2fabad2a67ac9a55ea72523c \
    --hash=sha256:2847e5d7beedb8d614186962c3d774d40d3374d580d2cbdab7f184580a39d234 \
    --hash=sha256:28e89e232c7593d33cac35425b58950789962011cc274aa43ef8865f2e11f46d \
    --hash=sha256:329c5a2e5a0ee942f2992c5e3ff40be03e75f745f48847f118a3cfece7a28546 \
    --hash=sha256:337322096d92808f76ad26061a8f5fccb22b0809bea39212cd6c406f6a7060d2 \
    --hash=sha256:3fcc780ae8edbb1d050d920ab44790201f027d59fdbd21362340a85c79066a74 \
    --hash=sha256:41bdeeb552d814bcd7fb52172b304898a35818107cc8778b5101423c9017b3de \
    --hash=sha256:4eddd98afc726f8aee1948858aed9e6feeb1758889dfd869072d4465973f6bfd \
    --hash=sha256:52e93b28db27ae7d208748f45d2db8a7b6a380e0d703f099c949d0f0d80b70e9 \
    --hash=sha256:55d62807f1c5a1682075c62436702aaba941daa316e9161e4b6ccebbbf38bda3 \
    --hash=sha256:5805e71e5b570d490938d55552f5a9e10f477c19400c38bf1d5190d760691846 \
    --hash=sha256:599daf06ea59bfedbec564b1692b0166a0045f32b6f0933b0dd4df59a854caf2 \
    --hash=sha256:60d5772e8195f4e9ebf74046a9121bbb90090f6550f81d8956a05387ba139353 \
    --hash=sha256:696d8e7d82398e810f2b3622b24e87906763b6ebfd90e361e88eb85b0e554dc8 \
    --hash=sha256:6e6061bf1e9565c29002e3c601cf68569c450be7fc3f7336671af7ddb4657166 \
    --hash=sha256:80ac992f25d10aaebe1ee15df45ca0d7571d0f70b645c08ec68733fb7a020206 \
    --hash=sha256:816bd9488a94cba78d93e1abb58000e8266fa9cc2aa9ccdd6eb0696acb24005b \
    --hash=sha256:85d2b77e7c9382f004b41d9c72c85537fac834fb141b0296942d52bf03fe4a3d \
    --hash=sha256:87c8ceb0cf8a5a51b8008b643844b7f4a8264a2c13fcbcd8a8316161725383fe \
    --hash=sha256:89ee2e967bd7ff85d84a2de09df10e021c9b38c7d91dead95b406ed6350c6997 \
    --hash=sha256:8bef097455dea90ffe855286926ae02d8faa335ed8e4067326257cb571fc1445 \
    --hash=sha256:8d11ebbd679e927593978aa44c10fc2092bc454b7d13fdc958d3e9d508aba7d0 \
    --hash=sha256:91e6c7db42638dc45cf2e13c73be16bf83179f7859b07cfc139518941320be96 \
    --hash=sha256:97e7ac860d64e2dcba5c5944cfc8fa9ea185cd84061c623536154d5a89237884 \
    --hash=sha256:990066bff27c4fcf3b69382b86f4c99b3652bab2a7e685d968cd4d0cfc6f67c6 \
    --hash=sha256:9fbc5b8f3dfe24784cee8ce0be3da2d8a79e46a276593db6868382d9c50d97b1 \
    --hash=sha256:ac4a39d1abae48184d420aa8e5e63efd1b75c8444dd95daa3e03f6c6310e9619 \
    --hash=sha256:b2c02d2ad98116e914d4f3155ffc905fd0c025d901ead3f6ed07385e19122c94 \
    --hash=sha256:b2d3337dcfaa99698aa2377c81c9ca72fcd89c07e7eb62ece3f23a3fe89b2ce4 \
    --hash=sha256:b489c36d1327868d207002391f662a1d163bdc8daf10ab2e5f6e41b9b96de3b1 \
    --hash=sha256:b641161c302efbb860ae6b081f406839a8b7d5573f20a455539823802c655f63 \
    --hash=sha256:b8ba29306c5de7717b5761b9ea74f9c72b9e2b834e24aa984da99cbfc70157fd \
    --hash=sha256:b9934adbd0f6e476f0ecff3c94626529f344f57b38c9a541f87098710b18af0a \
    --hash=sha256:ce85c43ae54845272f6f9cd8320d034d7a946e9773c693b27d620edec825e376 \
    --hash=sha256:cf868e08690cb89360eebc73ba4be7fb461cfbc6168dd88e2fbbe6f31812cd57 \
    --hash=sha256:d2905ce1df400360463c772b55d8e2518d0e488a87cdea13dd2c71dcb2a1fa16 \
    --hash=sha256:d57e20ba591727da0c230ab2c3f200ac9d6d333860d85348816e1dca4cc4792e \
    --hash=sha256:d6a8c9d4f8692917a3dc7eb25a6fb337bff86909febe2f793ec1928cd97bedfc \
    --hash=sha256:d923ff276f1c1f9680d32832f8d6c040fe9306cbfb5d161b0911e9634be9ef0a \
    --hash=sha256:daa7197b43c707462f06d2c693ffdbb5991cbb8b80b5b984007de431493a319c \
    --hash=sha256:dbd4c177afb8a8d9ba348d925b0b67246147af806f0b104af4d24f144d461cd5 \
    --hash=sha256:dc4d815b794fd8868c4d67602692c21bf5293a75e4b607bb92a11e821e2b859a \
    --hash=sha256:e9d21aaa84557d64209af04ff48e0ad5e28c5cca67ce43444e939579d085da72 \
    --hash=sha256:ea6b8aa9e08eea388c5f7a276fabb1d4b6b9d6e4ceb12cc477c3d352001768a9 \
    --hash=sha256:eabe7090db68c981fca689299c2d116400b553f4b713266b130cfc9e2aa9c5a9 \
    --hash=sha256:f2f6d303f3dee132b322a14cd8765287b8f86cdc10d2cb6a6fae234ea488888e \
    --hash=sha256:f33f3258aae89da191c6ebaa3bc517c6c4cbc9b9f689e5d8452f7aedbb913fa8 \
    --hash=sha256:f7bfb769f7efa0eefcd039dd19d843a4fbfbac52f1878b1da2ed5793ec9b1a65 \
    --hash=sha256:f89e21afe925fcfa655965ca8ea10f24773a1791400989ff32f467badfe4a064 \
    --hash=sha256:fa24255ae3c0ab67e613556375a4341af04a084bd58764731972bcbc8baeba36
grpcio==1.59.3 ; python_full_version == "3.10.13" \
    --hash=sha256:00912ce19914d038851be5cd380d94a03f9d195643c28e3ad03d355cc02ce7e8 \
    --hash=sha256:0511af8653fbda489ff11d542a08505d56023e63cafbda60e6e00d4e0bae86ea \
    --hash=sha256:0814942ba1bba269db4e760a34388640c601dece525c6a01f3b4ff030cc0db69 \
    --hash=sha256:0d42048b8a3286ea4134faddf1f9a59cf98192b94aaa10d910a25613c5eb5bfb \
    --hash=sha256:0e735ed002f50d4f3cb9ecfe8ac82403f5d842d274c92d99db64cfc998515e07 \
    --hash=sha256:16da0e40573962dab6cba16bec31f25a4f468e6d05b658e589090fe103b03e3d \
    --hash=sha256:1736496d74682e53dd0907fd515f2694d8e6a96c9a359b4080b2504bf2b2d91b \
    --hash=sha256:19ad26a7967f7999c8960d2b9fe382dae74c55b0c508c613a6c2ba21cddf2354 \
    --hash=sha256:33b8fd65d4e97efa62baec6171ce51f9cf68f3a8ba9f866f4abc9d62b5c97b79 \
    --hash=sha256:36636babfda14f9e9687f28d5b66d349cf88c1301154dc71c6513de2b6c88c59 \
    --hash=sha256:3996aaa21231451161dc29df6a43fcaa8b332042b6150482c119a678d007dd86 \
    --hash=sha256:45dddc5cb5227d30fa43652d8872dc87f086d81ab4b500be99413bad0ae198d7 \
    --hash=sha256:4619fea15c64bcdd9d447cdbdde40e3d5f1da3a2e8ae84103d94a9c1df210d7e \
    --hash=sha256:52cc38a7241b5f7b4a91aaf9000fdd38e26bb00d5e8a71665ce40cfcee716281 \
    --hash=sha256:575d61de1950b0b0699917b686b1ca108690702fcc2df127b8c9c9320f93e069 \
    --hash=sha256:5f9b2e591da751ac7fdd316cc25afafb7a626dededa9b414f90faad7f3ccebdb \
    --hash=sha256:60cddafb70f9a2c81ba251b53b4007e07cca7389e704f86266e22c4bffd8bf1d \
    --hash=sha256:6a5c3a96405966c023e139c3bcccb2c7c776a6f256ac6d70f8558c9041bdccc3 \
    --hash=sha256:6c75a1fa0e677c1d2b6d4196ad395a5c381dfb8385f07ed034ef667cdcdbcc25 \
    --hash=sha256:72b71dad2a3d1650e69ad42a5c4edbc59ee017f08c32c95694172bc501def23c \
    --hash=sha256:73afbac602b8f1212a50088193601f869b5073efa9855b3e51aaaec97848fc8a \
    --hash=sha256:7800f99568a74a06ebdccd419dd1b6e639b477dcaf6da77ea702f8fb14ce5f80 \
    --hash=sha256:8022ca303d6c694a0d7acfb2b472add920217618d3a99eb4b14edc7c6a7e8fcf \
    --hash=sha256:8239b853226e4824e769517e1b5232e7c4dda3815b200534500338960fcc6118 \
    --hash=sha256:83113bcc393477b6f7342b9f48e8a054330c895205517edc66789ceea0796b53 \
    --hash=sha256:8cd76057b5c9a4d68814610ef9226925f94c1231bbe533fdf96f6181f7d2ff9e \
    --hash=sha256:8d993399cc65e3a34f8fd48dd9ad7a376734564b822e0160dd18b3d00c1a33f9 \
    --hash=sha256:95b5506e70284ac03b2005dd9ffcb6708c9ae660669376f0192a710687a22556 \
    --hash=sha256:95d6fd804c81efe4879e38bfd84d2b26e339a0a9b797e7615e884ef4686eb47b \
    --hash=sha256:9e17660947660ccfce56c7869032910c179a5328a77b73b37305cd1ee9301c2e \
    --hash=sha256:a93a82876a4926bf451db82ceb725bd87f42292bacc94586045261f501a86994 \
    --hash=sha256:aca028a6c7806e5b61e5f9f4232432c52856f7fcb98e330b20b6bc95d657bdcc \
    --hash=sha256:b1f00a3e6e0c3dccccffb5579fc76ebfe4eb40405ba308505b41ef92f747746a \
    --hash=sha256:b36683fad5664283755a7f4e2e804e243633634e93cd798a46247b8e54e3cb0d \
    --hash=sha256:b491e5bbcad3020a96842040421e508780cade35baba30f402df9d321d1c423e \
    --hash=sha256:c0bd141f4f41907eb90bda74d969c3cb21c1c62779419782a5b3f5e4b5835718 \
    --hash=sha256:c0f0a11d82d0253656cc42e04b6a149521e02e755fe2e4edd21123de610fd1d4 \
    --hash=sha256:c4b0076f0bf29ee62335b055a9599f52000b7941f577daa001c7ef961a1fbeab \
    --hash=sha256:c82ca1e4be24a98a253d6dbaa216542e4163f33f38163fc77964b0f0d255b552 \
    --hash=sha256:cb4e9cbd9b7388fcb06412da9f188c7803742d06d6f626304eb838d1707ec7e3 \
    --hash=sha256:cdbc6b32fadab9bebc6f49d3e7ec4c70983c71e965497adab7f87de218e84391 \
    --hash=sha256:ce31fa0bfdd1f2bb15b657c16105c8652186eab304eb512e6ae3b99b2fdd7d13 \
    --hash=sha256:d1d1a17372fd425addd5812049fa7374008ffe689585f27f802d0935522cf4b7 \
    --hash=sha256:d787ecadea865bdf78f6679f6f5bf4b984f18f659257ba612979df97a298b3c3 \
    --hash=sha256:ddbd1a16138e52e66229047624de364f88a948a4d92ba20e4e25ad7d22eef025 \
    --hash=sha256:e1d8e01438d5964a11167eec1edb5f85ed8e475648f36c834ed5db4ffba24ac8 \
    --hash=sha256:e58b3cadaa3c90f1efca26ba33e0d408b35b497307027d3d707e4bcd8de862a6 \
    --hash=sha256:e78dc982bda74cef2ddfce1c91d29b96864c4c680c634e279ed204d51e227473 \
    --hash=sha256:ea40ce4404e7cca0724c91a7404da410f0144148fdd58402a5942971e3469b94 \
    --hash=sha256:eb8ba504c726befe40a356ecbe63c6c3c64c9a439b3164f5a718ec53c9874da0 \
    --hash=sha256:ed26826ee423b11477297b187371cdf4fa1eca874eb1156422ef3c9a60590dd9 \
    --hash=sha256:f2eb8f0c7c0c62f7a547ad7a91ba627a5aa32a5ae8d930783f7ee61680d7eb8d \
    --hash=sha256:fb111aa99d3180c361a35b5ae1e2c63750220c584a1344229abc139d5c891881 \
    --hash=sha256:fcfa56f8d031ffda902c258c84c4b88707f3a4be4827b4e3ab8ec7c24676320d
h11==0.14.0 ; python_full_version == "3.10.13" \
    --hash=sha256:8f19fbbe99e72420ff35c00b27a34cb9937e902a8b810e2c88300c6f0a3b699d \
    --hash=sha256:e3fe4ac4b851c468cc8363d500db52c2ead036020723024a109d37346efaa761
h5py==3.10.0 ; python_full_version == "3.10.13" \
    --hash=sha256:012ab448590e3c4f5a8dd0f3533255bc57f80629bf7c5054cf4c87b30085063c \
    --hash=sha256:212bb997a91e6a895ce5e2f365ba764debeaef5d2dca5c6fb7098d66607adf99 \
    --hash=sha256:2381e98af081b6df7f6db300cd88f88e740649d77736e4b53db522d8874bf2dc \
    --hash=sha256:2c8e4fda19eb769e9a678592e67eaec3a2f069f7570c82d2da909c077aa94339 \
    --hash=sha256:3074ec45d3dc6e178c6f96834cf8108bf4a60ccb5ab044e16909580352010a97 \
    --hash=sha256:3c97d03f87f215e7759a354460fb4b0d0f27001450b18b23e556e7856a0b21c3 \
    --hash=sha256:43a61b2c2ad65b1fabc28802d133eed34debcc2c8b420cb213d3d4ef4d3e2229 \
    --hash=sha256:492305a074327e8d2513011fa9fffeb54ecb28a04ca4c4227d7e1e9616d35641 \
    --hash=sha256:5dfc65ac21fa2f630323c92453cadbe8d4f504726ec42f6a56cf80c2f90d6c52 \
    --hash=sha256:667fe23ab33d5a8a6b77970b229e14ae3bb84e4ea3382cc08567a02e1499eedd \
    --hash=sha256:6c013d2e79c00f28ffd0cc24e68665ea03ae9069e167087b2adb5727d2736a52 \
    --hash=sha256:781a24263c1270a62cd67be59f293e62b76acfcc207afa6384961762bb88ea03 \
    --hash=sha256:86df4c2de68257b8539a18646ceccdcf2c1ce6b1768ada16c8dcfb489eafae20 \
    --hash=sha256:90286b79abd085e4e65e07c1bd7ee65a0f15818ea107f44b175d2dfe1a4674b7 \
    --hash=sha256:92273ce69ae4983dadb898fd4d3bea5eb90820df953b401282ee69ad648df684 \
    --hash=sha256:93dd840bd675787fc0b016f7a05fc6efe37312a08849d9dd4053fd0377b1357f \
    --hash=sha256:9450464b458cca2c86252b624279115dcaa7260a40d3cb1594bf2b410a2bd1a3 \
    --hash=sha256:ae2f0201c950059676455daf92700eeb57dcf5caaf71b9e1328e6e6593601770 \
    --hash=sha256:aece0e2e1ed2aab076c41802e50a0c3e5ef8816d60ece39107d68717d4559824 \
    --hash=sha256:b963fb772964fc1d1563c57e4e2e874022ce11f75ddc6df1a626f42bd49ab99f \
    --hash=sha256:ba9ab36be991119a3ff32d0c7cbe5faf9b8d2375b5278b2aea64effbeba66039 \
    --hash=sha256:d4682b94fd36ab217352be438abd44c8f357c5449b8995e63886b431d260f3d3 \
    --hash=sha256:d93adc48ceeb33347eb24a634fb787efc7ae4644e6ea4ba733d099605045c049 \
    --hash=sha256:f42e6c30698b520f0295d70157c4e202a9e402406f50dc08f5a7bc416b24e52d \
    --hash=sha256:fd6f6d1384a9f491732cee233b99cd4bfd6e838a8815cc86722f9d2ee64032af
httpcore==1.0.2 ; python_full_version == "3.10.13" \
    --hash=sha256:096cc05bca73b8e459a1fc3dcf585148f63e534eae4339559c9b8a8d6399acc7 \
    --hash=sha256:9fc092e4799b26174648e54b74ed5f683132a464e95643b226e00c2ed2fa6535
httplib2==0.22.0 ; python_full_version == "3.10.13" \
    --hash=sha256:14ae0a53c1ba8f3d37e9e27cf37eabb0fb9980f435ba405d546948b009dd64dc \
    --hash=sha256:d7a10bc5ef5ab08322488bde8c726eeee5c8618723fdb399597ec58f3d82df81
httpx==0.25.2 ; python_full_version == "3.10.13" \
    --hash=sha256:8b8fcaa0c8ea7b05edd69a094e63a2094c4efcb48129fb757361bc423c0ad9e8 \
    --hash=sha256:a05d3d052d9b2dfce0e3896636467f8a5342fb2b902c819428e1ac65413ca118
huggingface-hub==0.19.4 ; python_full_version == "3.10.13" \
    --hash=sha256:176a4fc355a851c17550e7619488f383189727eab209534d7cef2114dae77b22 \
    --hash=sha256:dba013f779da16f14b606492828f3760600a1e1801432d09fe1c33e50b825bb5
idna==3.6 ; python_full_version == "3.10.13" \
    --hash=sha256:9ecdbbd083b06798ae1e86adcbfe8ab1479cf864e4ee30fe4e46a003d12491ca \
    --hash=sha256:c05567e9c24a6b9faaa835c4821bad0590fbb9d5779e7caa6e1cc4978e7eb24f
iniconfig==2.0.0 ; python_full_version == "3.10.13" \
    --hash=sha256:2d91e135bf72d31a410b17c16da610a82cb55f6b0477d1a902134b24a455b8b3 \
    --hash=sha256:b6a85871a79d2e3b22d2d1b94ac2824226a63c6b741c88f7ae975f18b6778374
itsdangerous==2.1.2 ; python_full_version == "3.10.13" \
    --hash=sha256:2c2349112351b88699d8d4b6b075022c0808887cb7ad10069318a8b0bc88db44 \
    --hash=sha256:5dbbc68b317e5e42f327f9021763545dc3fc3bfe22e6deb96aaf1fc38874156a
jinja2==3.1.2 ; python_full_version == "3.10.13" \
    --hash=sha256:31351a702a408a9e7595a8fc6150fc3f43bb6bf7e319770cbc0db9df9437e852 \
    --hash=sha256:6088930bfe239f0e6710546ab9c19c9ef35e29792895fed6e6e31a023a182a61
joblib==1.3.2 ; python_full_version == "3.10.13" \
    --hash=sha256:92f865e621e17784e7955080b6d042489e3b8e294949cc44c6eac304f59772b1 \
    --hash=sha256:ef4331c65f239985f3f2220ecc87db222f08fd22097a3dd5698f693875f8cbb9
keras==2.15.0 ; python_full_version == "3.10.13" \
    --hash=sha256:2dcc6d2e30cf9c951064b63c1f4c404b966c59caf09e01f3549138ec8ee0dd1f \
    --hash=sha256:81871d298c064dc4ac6b58440fdae67bfcf47c8d7ad28580fab401834c06a575
kombu==5.3.4 ; python_full_version == "3.10.13" \
    --hash=sha256:0bb2e278644d11dea6272c17974a3dbb9688a949f3bb60aeb5b791329c44fadc \
    --hash=sha256:63bb093fc9bb80cfb3a0972336a5cec1fa7ac5f9ef7e8237c6bf8dda9469313e
libclang==16.0.6 ; python_full_version == "3.10.13" \
    --hash=sha256:1e940048f51d0b0999099a9b78629ab8a64b62af5e9ff1b2b062439c21ee244d \
    --hash=sha256:4a9acbfd9c135a72f80d5dbff7588dfb0c81458244a89b9e83526e8595880e0a \
    --hash=sha256:4acdde39dfe410c877b4ccc0d4b57eb952100e4ee26bbdf6cfdb88e2033a7d31 \
    --hash=sha256:8130482120500476a027171f8f3c8dfc2536b591716eea71fc5da22cae13131b \
    --hash=sha256:88bc7e7b393c32e41e03ba77ef02fdd647da1f764c2cd028e69e0837080b79f6 \
    --hash=sha256:9dcdc730939788b8b69ffd6d5d75fe5366e3ee007f1e36a99799ec0b0c001492 \
    --hash=sha256:d80ed5827736ed5ec2bcedf536720476fd9d4fa4c79ef0cb24aea4c59332f361 \
    --hash=sha256:da9e47ebc3f0a6d90fb169ef25f9fbcd29b4a4ef97a8b0e3e3a17800af1423f4 \
    --hash=sha256:daab4a11dae228f1efa9efa3fe638b493b14d8d52c71fb3c7019e2f1df4514c2 \
    --hash=sha256:e1a5ad1e895e5443e205568c85c04b4608e4e973dae42f4dfd9cb46c81d1486b \
    --hash=sha256:f04e3060ae1f207f234d0608900c99c50edcb743e5e18276d78da2ddd727d39f
mako==1.3.0 ; python_full_version == "3.10.13" \
    --hash=sha256:57d4e997349f1a92035aa25c17ace371a4213f2ca42f99bee9a602500cfd54d9 \
    --hash=sha256:e3a9d388fd00e87043edbe8792f45880ac0114e9c4adc69f6e9bfb2c55e3b11b
markdown-it-py==3.0.0 ; python_full_version == "3.10.13" \
    --hash=sha256:355216845c60bd96232cd8d8c40e8f9765cc86f46880e43a8fd22dc1a1a8cab1 \
    --hash=sha256:e3f60a94fa066dc52ec76661e37c851cb232d92f9886b15cb560aaada2df8feb
markdown==3.5.1 ; python_full_version == "3.10.13" \
    --hash=sha256:5874b47d4ee3f0b14d764324d2c94c03ea66bee56f2d929da9f2508d65e722dc \
    --hash=sha256:b65d7beb248dc22f2e8a31fb706d93798093c308dc1aba295aedeb9d41a813bd
markupsafe==2.1.3 ; python_full_version == "3.10.13" \
    --hash=sha256:05fb21170423db021895e1ea1e1f3ab3adb85d1c2333cbc2310f2a26bc77272e \
    --hash=sha256:0a4e4a1aff6c7ac4cd55792abf96c915634c2b97e3cc1c7129578aa68ebd754e \
    --hash=sha256:10bbfe99883db80bdbaff2dcf681dfc6533a614f700da1287707e8a5d78a8431 \
    --hash=sha256:134da1eca9ec0ae528110ccc9e48041e0828d79f24121a1a146161103c76e686 \
    --hash=sha256:14ff806850827afd6b07a5f32bd917fb7f45b046ba40c57abdb636674a8b559c \
    --hash=sha256:1577735524cdad32f9f694208aa75e422adba74f1baee7551620e43a3141f559 \
    --hash=sha256:1b40069d487e7edb2676d3fbdb2b0829ffa2cd63a2ec26c4938b2d34391b4ecc \
    --hash=sha256:1b8dd8c3fd14349433c79fa8abeb573a55fc0fdd769133baac1f5e07abf54aeb \
    --hash=sha256:1f67c7038d560d92149c060157d623c542173016c4babc0c1913cca0564b9939 \
    --hash=sha256:282c2cb35b5b673bbcadb33a585408104df04f14b2d9b01d4c345a3b92861c2c \
    --hash=sha256:2c1b19b3aaacc6e57b7e25710ff571c24d6c3613a45e905b1fde04d691b98ee0 \
    --hash=sha256:2ef12179d3a291be237280175b542c07a36e7f60718296278d8593d21ca937d4 \
    --hash=sha256:338ae27d6b8745585f87218a3f23f1512dbf52c26c28e322dbe54bcede54ccb9 \
    --hash=sha256:3c0fae6c3be832a0a0473ac912810b2877c8cb9d76ca48de1ed31e1c68386575 \
    --hash=sha256:3fd4abcb888d15a94f32b75d8fd18ee162ca0c064f35b11134be77050296d6ba \
    --hash=sha256:42de32b22b6b804f42c5d98be4f7e5e977ecdd9ee9b660fda1a3edf03b11792d \
    --hash=sha256:47d4f1c5f80fc62fdd7777d0d40a2e9dda0a05883ab11374334f6c4de38adffd \
    --hash=sha256:504b320cd4b7eff6f968eddf81127112db685e81f7e36e75f9f84f0df46041c3 \
    --hash=sha256:525808b8019e36eb524b8c68acdd63a37e75714eac50e988180b169d64480a00 \
    --hash=sha256:56d9f2ecac662ca1611d183feb03a3fa4406469dafe241673d521dd5ae92a155 \
    --hash=sha256:5bbe06f8eeafd38e5d0a4894ffec89378b6c6a625ff57e3028921f8ff59318ac \
    --hash=sha256:65c1a9bcdadc6c28eecee2c119465aebff8f7a584dd719facdd9e825ec61ab52 \
    --hash=sha256:68e78619a61ecf91e76aa3e6e8e33fc4894a2bebe93410754bd28fce0a8a4f9f \
    --hash=sha256:69c0f17e9f5a7afdf2cc9fb2d1ce6aabdb3bafb7f38017c0b77862bcec2bbad8 \
    --hash=sha256:6b2b56950d93e41f33b4223ead100ea0fe11f8e6ee5f641eb753ce4b77a7042b \
    --hash=sha256:715d3562f79d540f251b99ebd6d8baa547118974341db04f5ad06d5ea3eb8007 \
    --hash=sha256:787003c0ddb00500e49a10f2844fac87aa6ce977b90b0feaaf9de23c22508b24 \
    --hash=sha256:7ef3cb2ebbf91e330e3bb937efada0edd9003683db6b57bb108c4001f37a02ea \
    --hash=sha256:8023faf4e01efadfa183e863fefde0046de576c6f14659e8782065bcece22198 \
    --hash=sha256:8758846a7e80910096950b67071243da3e5a20ed2546e6392603c096778d48e0 \
    --hash=sha256:8afafd99945ead6e075b973fefa56379c5b5c53fd8937dad92c662da5d8fd5ee \
    --hash=sha256:8c41976a29d078bb235fea9b2ecd3da465df42a562910f9022f1a03107bd02be \
    --hash=sha256:8e254ae696c88d98da6555f5ace2279cf7cd5b3f52be2b5cf97feafe883b58d2 \
    --hash=sha256:8f9293864fe09b8149f0cc42ce56e3f0e54de883a9de90cd427f191c346eb2e1 \
    --hash=sha256:9402b03f1a1b4dc4c19845e5c749e3ab82d5078d16a2a4c2cd2df62d57bb0707 \
    --hash=sha256:962f82a3086483f5e5f64dbad880d31038b698494799b097bc59c2edf392fce6 \
    --hash=sha256:9aad3c1755095ce347e26488214ef77e0485a3c34a50c5a5e2471dff60b9dd9c \
    --hash=sha256:9dcdfd0eaf283af041973bff14a2e143b8bd64e069f4c383416ecd79a81aab58 \
    --hash=sha256:aa57bd9cf8ae831a362185ee444e15a93ecb2e344c8e52e4d721ea3ab6ef1823 \
    --hash=sha256:aa7bd130efab1c280bed0f45501b7c8795f9fdbeb02e965371bbef3523627779 \
    --hash=sha256:ab4a0df41e7c16a1392727727e7998a467472d0ad65f3ad5e6e765015df08636 \
    --hash=sha256:ad9e82fb8f09ade1c3e1b996a6337afac2b8b9e365f926f5a61aacc71adc5b3c \
    --hash=sha256:af598ed32d6ae86f1b747b82783958b1a4ab8f617b06fe68795c7f026abbdcad \
    --hash=sha256:b076b6226fb84157e3f7c971a47ff3a679d837cf338547532ab866c57930dbee \
    --hash=sha256:b7ff0f54cb4ff66dd38bebd335a38e2c22c41a8ee45aa608efc890ac3e3931bc \
    --hash=sha256:bfce63a9e7834b12b87c64d6b155fdd9b3b96191b6bd334bf37db7ff1fe457f2 \
    --hash=sha256:c011a4149cfbcf9f03994ec2edffcb8b1dc2d2aede7ca243746df97a5d41ce48 \
    --hash=sha256:c9c804664ebe8f83a211cace637506669e7890fec1b4195b505c214e50dd4eb7 \
    --hash=sha256:ca379055a47383d02a5400cb0d110cef0a776fc644cda797db0c5696cfd7e18e \
    --hash=sha256:cb0932dc158471523c9637e807d9bfb93e06a95cbf010f1a38b98623b929ef2b \
    --hash=sha256:cd0f502fe016460680cd20aaa5a76d241d6f35a1c3350c474bac1273803893fa \
    --hash=sha256:ceb01949af7121f9fc39f7d27f91be8546f3fb112c608bc4029aef0bab86a2a5 \
    --hash=sha256:d080e0a5eb2529460b30190fcfcc4199bd7f827663f858a226a81bc27beaa97e \
    --hash=sha256:dd15ff04ffd7e05ffcb7fe79f1b98041b8ea30ae9234aed2a9168b5797c3effb \
    --hash=sha256:df0be2b576a7abbf737b1575f048c23fb1d769f267ec4358296f31c2479db8f9 \
    --hash=sha256:e09031c87a1e51556fdcb46e5bd4f59dfb743061cf93c4d6831bf894f125eb57 \
    --hash=sha256:e4dd52d80b8c83fdce44e12478ad2e85c64ea965e75d66dbeafb0a3e77308fcc \
    --hash=sha256:f698de3fd0c4e6972b92290a45bd9b1536bffe8c6759c62471efaa8acb4c37bc \
    --hash=sha256:fec21693218efe39aa7f8599346e90c705afa52c5b31ae019b2e57e8f6542bb2 \
    --hash=sha256:ffcc3f7c66b5f5b7931a5aa68fc9cecc51e685ef90282f4a82f0f5e9b704ad11
marshmallow==3.20.1 ; python_full_version == "3.10.13" \
    --hash=sha256:5d2371bbe42000f2b3fb5eaa065224df7d8f8597bc19a1bbfa5bfe7fba8da889 \
    --hash=sha256:684939db93e80ad3561392f47be0230743131560a41c5110684c16e21ade0a5c
mdurl==0.1.2 ; python_full_version == "3.10.13" \
    --hash=sha256:84008a41e51615a49fc9966191ff91509e3c40b939176e643fd50a5c2196b8f8 \
    --hash=sha256:bb413d29f5eea38f31dd4754dd7377d4465116fb207585f97bf925588687c1ba
ml-dtypes==0.2.0 ; python_full_version == "3.10.13" \
    --hash=sha256:022d5a4ee6be14569c2a9d1549e16f1ec87ca949681d0dca59995445d5fcdd5b \
    --hash=sha256:1749b60348da71fd3c2ab303fdbc1965958dc50775ead41f5669c932a341cafd \
    --hash=sha256:32107e7fa9f62db9a5281de923861325211dfff87bd23faefb27b303314635ab \
    --hash=sha256:35b984cddbe8173b545a0e3334fe56ea1a5c3eb67c507f60d0cfde1d3fa8f8c2 \
    --hash=sha256:36d28b8861a8931695e5a31176cad5ae85f6504906650dea5598fbec06c94606 \
    --hash=sha256:50845af3e9a601810751b55091dee6c2562403fa1cb4e0123675cf3a4fc2c17a \
    --hash=sha256:6488eb642acaaf08d8020f6de0a38acee7ac324c1e6e92ee0c0fea42422cb797 \
    --hash=sha256:75015818a7fccf99a5e8ed18720cb430f3e71a8838388840f4cdf225c036c983 \
    --hash=sha256:80d304c836d73f10605c58ccf7789c171cc229bfb678748adfb7cea2510dfd0e \
    --hash=sha256:832a019a1b6db5c4422032ca9940a990fa104eee420f643713241b3a518977fa \
    --hash=sha256:8faaf0897942c8253dd126662776ba45f0a5861968cf0f06d6d465f8a7bc298a \
    --hash=sha256:bc29a0524ef5e23a7fbb8d881bdecabeb3fc1d19d9db61785d077a86cb94fab2 \
    --hash=sha256:df6a76e1c8adf484feb138ed323f9f40a7b6c21788f120f7c78bec20ac37ee81 \
    --hash=sha256:e70047ec2c83eaee01afdfdabee2c5b0c133804d90d0f7db4dd903360fcc537c \
    --hash=sha256:e85ba8e24cf48d456e564688e981cf379d4c8e644db0a2f719b78de281bac2ca \
    --hash=sha256:f00c71c8c63e03aff313bc6a7aeaac9a4f1483a921a6ffefa6d4404efd1af3d0 \
    --hash=sha256:f08c391c2794f2aad358e6f4c70785a9a7b1df980ef4c232b3ccd4f6fe39f719
numpy==1.26.2 ; python_full_version == "3.10.13" \
    --hash=sha256:06fa1ed84aa60ea6ef9f91ba57b5ed963c3729534e6e54055fc151fad0423f0a \
    --hash=sha256:174a8880739c16c925799c018f3f55b8130c1f7c8e75ab0a6fa9d41cab092fd6 \
    --hash=sha256:1a13860fdcd95de7cf58bd6f8bc5a5ef81c0b0625eb2c9a783948847abbef2c2 \
    --hash=sha256:1cc3d5029a30fb5f06704ad6b23b35e11309491c999838c31f124fee32107c79 \
    --hash=sha256:22f8fc02fdbc829e7a8c578dd8d2e15a9074b630d4da29cda483337e300e3ee9 \
    --hash=sha256:26c9d33f8e8b846d5a65dd068c14e04018d05533b348d9eaeef6c1bd787f9919 \
    --hash=sha256:2b3fca8a5b00184828d12b073af4d0fc5fdd94b1632c2477526f6bd7842d700d \
    --hash=sha256:2beef57fb031dcc0dc8fa4fe297a742027b954949cabb52a2a376c144e5e6060 \
    --hash=sha256:36340109af8da8805d8851ef1d74761b3b88e81a9bd80b290bbfed61bd2b4f75 \
    --hash=sha256:3703fc9258a4a122d17043e57b35e5ef1c5a5837c3db8be396c82e04c1cf9b0f \
    --hash=sha256:3ced40d4e9e18242f70dd02d739e44698df3dcb010d31f495ff00a31ef6014fe \
    --hash=sha256:4a06263321dfd3598cacb252f51e521a8cb4b6df471bb12a7ee5cbab20ea9167 \
    --hash=sha256:4eb8df4bf8d3d90d091e0146f6c28492b0be84da3e409ebef54349f71ed271ef \
    --hash=sha256:5d5244aabd6ed7f312268b9247be47343a654ebea52a60f002dc70c769048e75 \
    --hash=sha256:64308ebc366a8ed63fd0bf426b6a9468060962f1a4339ab1074c228fa6ade8e3 \
    --hash=sha256:6a3cdb4d9c70e6b8c0814239ead47da00934666f668426fc6e94cce869e13fd7 \
    --hash=sha256:854ab91a2906ef29dc3925a064fcd365c7b4da743f84b123002f6139bcb3f8a7 \
    --hash=sha256:94cc3c222bb9fb5a12e334d0479b97bb2df446fbe622b470928f5284ffca3f8d \
    --hash=sha256:96ca5482c3dbdd051bcd1fce8034603d6ebfc125a7bd59f55b40d8f5d246832b \
    --hash=sha256:a2bbc29fcb1771cd7b7425f98b05307776a6baf43035d3b80c4b0f29e9545186 \
    --hash=sha256:a4cd6ed4a339c21f1d1b0fdf13426cb3b284555c27ac2f156dfdaaa7e16bfab0 \
    --hash=sha256:aa18428111fb9a591d7a9cc1b48150097ba6a7e8299fb56bdf574df650e7d1f1 \
    --hash=sha256:aa317b2325f7aa0a9471663e6093c210cb2ae9c0ad824732b307d2c51983d5b6 \
    --hash=sha256:b04f5dc6b3efdaab541f7857351aac359e6ae3c126e2edb376929bd3b7f92d7e \
    --hash=sha256:b272d4cecc32c9e19911891446b72e986157e6a1809b7b56518b4f3755267523 \
    --hash=sha256:b361d369fc7e5e1714cf827b731ca32bff8d411212fccd29ad98ad622449cc36 \
    --hash=sha256:b96e7b9c624ef3ae2ae0e04fa9b460f6b9f17ad8b4bec6d7756510f1f6c0c841 \
    --hash=sha256:baf8aab04a2c0e859da118f0b38617e5ee65d75b83795055fb66c0d5e9e9b818 \
    --hash=sha256:bcc008217145b3d77abd3e4d5ef586e3bdfba8fe17940769f8aa09b99e856c00 \
    --hash=sha256:bd3f0091e845164a20bd5a326860c840fe2af79fa12e0469a12768a3ec578d80 \
    --hash=sha256:cc392fdcbd21d4be6ae1bb4475a03ce3b025cd49a9be5345d76d7585aea69440 \
    --hash=sha256:d73a3abcac238250091b11caef9ad12413dab01669511779bc9b29261dd50210 \
    --hash=sha256:f43740ab089277d403aa07567be138fc2a89d4d9892d113b76153e0e412409f8 \
    --hash=sha256:f65738447676ab5777f11e6bbbdb8ce11b785e105f690bc45966574816b6d3ea \
    --hash=sha256:f79b231bf5c16b1f39c7f4875e1ded36abee1591e98742b05d8a0fb55d8a3eec \
    --hash=sha256:fe6b44fb8fcdf7eda4ef4461b97b3f63c466b27ab151bec2366db8b197387841
nvidia-cublas-cu11==11.10.3.66 ; platform_system == "Linux" and python_full_version == "3.10.13" \
    --hash=sha256:8ac17ba6ade3ed56ab898a036f9ae0756f1e81052a317bf98f8c6d18dc3ae49e \
    --hash=sha256:d32e4d75f94ddfb93ea0a5dda08389bcc65d8916a25cb9f37ac89edaeed3bded
```

```
nvidia-cuda-nvrtc-cu11==11.7.99 ; platform_system == "Linux" and python_full_version == "3.10.13" \
    --hash=sha256:9f1562822ea264b7e34ed5930567e89242d266448e936b85bc97a3370feabb03 \
    --hash=sha256:f2effeb1309bdd1b3854fc9b17eaf997808f8b25968ce0c7070945c4265d64a3 \
    --hash=sha256:f7d9610d9b7c331fa0da2d1b2858a4a8315e6d49765091d28711c8946e7425e7
nvidia-cuda-runtime-cu11==11.7.99 ; platform_system == "Linux" and python_full_version == "3.10.13" \
    --hash=sha256:bc77fa59a7679310df9d5c70ab13c4e34c64ae2124dd1efd7e5474b71be125c7 \
    --hash=sha256:cc768314ae58d2641f07eac350f40f99dcb35719c4faff4bc458a7cd2b119e31
nvidia-cudnn-cu11==8.5.0.96 ; platform_system == "Linux" and python_full_version == "3.10.13" \
    --hash=sha256:402f40adfc6f418f9dae9ab402e773cfed9beae52333f6d86ae3107a1b9527e7 \
    --hash=sha256:71f8111eb830879ff2836db3cccf03bbd735df9b0d17cd93761732ac50a8a108
oauth2==1.9.0.post1 ; python_full_version == "3.10.13" \
    --hash=sha256:15b5c42301f46dd63113f1214b0d81a8b16254f65a86d3c32a1b52297f3266e6 \
    --hash=sha256:c006a85e7c60107c7cc6da1b184b5c719f6dd7202098196dfa6e55df669b59bf
oauthlib==3.2.2 ; python_full_version == "3.10.13" \
    --hash=sha256:8139f29aac13e25d502680e9e19963e83f16838d48a0d71c287fe40e7067fbca \
    --hash=sha256:9859c40929662bec5d64f34d01c99e093149682a3f38915dc0655d5a633dd918
openai==1.3.5 ; python_full_version == "3.10.13" \
    --hash=sha256:163e7ece4af76e961f58b75ea20a42b0d0c2a240c2f81b41a3d1c5962463cdf8 \
    --hash=sha256:9437458978fb502e61336c3082e02b09c49feebe0e8516a2b8fb4563e6e4af4e
opt-einsum==3.3.0 ; python_full_version == "3.10.13" \
    --hash=sha256:2455e59e3947d3c275477df7f5205b30635e266fe6dc300e3d9f9646bfcea147 \
    --hash=sha256:59f6475f77bbc37dcf7cd748519c0ec60722e91e63ca114e68821c0c54a46549
packaging==23.2 ; python_full_version == "3.10.13" \
    --hash=sha256:048fb0e9405036518eaaf48a55953c750c11e1a1b68e0dd1a9d62ed0c092cfc5 \
    --hash=sha256:8c491190033a9af7e1d931d0b5dacc2ef47509b34dd0de67ed209b5203fc88c7
passlib==1.7.4 ; python_full_version == "3.10.13" \
    --hash=sha256:aa6bca462b8d8bda89c70b382f0c298a20b5560af6cbfa2dce410c0a2fb669f1 \
    --hash=sha256:defd50f72b65c5402ab2c573830a6978e5f202ad0d984793c8dde2c4152ebe04
pluggy==1.3.0 ; python_full_version == "3.10.13" \
    --hash=sha256:cf61ae8f126ac6f7c451172cf30e3e43d3ca77615509771b3a984a0730651e12 \
    --hash=sha256:d89c696a773f8bd377d18e5ecda92b7a3793cbe66c87060a6fb58c7b6e1061f7
prompt-toolkit==3.0.41 ; python_full_version == "3.10.13" \
    --hash=sha256:941367d97fc815548822aa26c2a269fdc4eb21e9ec05fc5d447cf09bad5d75f0 \
    --hash=sha256:f36fe301fafb7470e86aaf90f036eef600a3210be4decf461a5b1ca8403d3cb2
protobuf==4.23.4 ; python_full_version == "3.10.13" \
    --hash=sha256:0a5759f5696895de8cc913f084e27fd4125e8fb0914bb729a17816a33819f474 \
    --hash=sha256:351cc90f7d10839c480aeb9b870a211e322bf05f6ab3f55fcb2f51331f80a7d2 \
    --hash=sha256:5fea3c64d41ea5ecf5697b83e41d09b9589e6f20b677ab3c48e5f242d9b7897b \
    --hash=sha256:6dd9b9940e3f17077e820b75851126615ee38643c2c5332aa7a359988820c720 \
    --hash=sha256:7b19b6266d92ca6a2a87effa88ecc4af73ebc5cfde194dc737cf8ef23a9a3b12 \
    --hash=sha256:8547bf44fe8cec3c69e3042f5c4fb3e36eb2a7a013bb0a44c018fc1e427aafbd \
    --hash=sha256:9053df6df8e5a76c84339ee4a9f5a2661ceee4a0dab019e8663c50ba324208b0 \
    --hash=sha256:c3e0939433c40796ca4cfc0fac08af50b00eb66a40bbbc5dee711998fb0bbc1e \
    --hash=sha256:ccd9430c0719dce806b93f89c91de7977304729e55377f872a92465d548329a9 \
    --hash=sha256:e1c915778d8ced71e26fcf43c0866d7499891bca14c4368448a82edc61fdbc70 \
    --hash=sha256:e9d0be5bf34b275b9f87ba7407796556abeeba635455d036c7351f7c183ef8ff \
    --hash=sha256:effeac51ab79332d44fba74660d40ae79985901ac21bca408f8dc335a81aa597 \
    --hash=sha256:fee88269a090ada09ca63551bf2f573eb2424035bcf2cb1b121895b01a46594a
pyasn1-modules==0.3.0 ; python_full_version == "3.10.13" \
    --hash=sha256:5bd01446b736eb9d31512a30d46c1ac3395d676c6f3cafa4c03eb54b9925631c \
    --hash=sha256:d3ccd6ed470d9ffbc716be08bd90efbd44d0734bc9303818f7336070984a162d
pyasn1==0.5.1 ; python_full_version == "3.10.13" \
    --hash=sha256:4439847c58d40b1d0a573d07e3856e95333f1976294494c325775aeca506eb58 \
    --hash=sha256:6d391a96e59b23130a5cfa74d6fd7f388dbbe26cc8f1edf39fdddf08d9d6676c
```

```
pydantic-core==2.14.5 ; python_full_version == "3.10.13" \
    --hash=sha256:038c9f763e650712b899f983076ce783175397c848da04985658e7628cbe873b \
    --hash=sha256:074f3d86f081ce61414d2dc44901f4f83617329c6f3ab49d2bc6c96948b2c26b \
    --hash=sha256:079206491c435b60778cf2b0ee5fd645e61ffd6e70c47806c9ed51fc75af078d \
    --hash=sha256:09b0e985fbaf13e6b06a56d21694d12ebca6ce5414b9211edf6f17738d82b0f8 \
    --hash=sha256:0f6116a558fd06d1b7c2902d1c4cf64a5bd49d67c3540e61eccca93f41418124 \
    --hash=sha256:103ef8d5b58596a731b690112819501ba1db7a36f4ee99f7892c40da02c3e189 \
    --hash=sha256:16e29bad40bcf97aac682a58861249ca9dcc57c3f6be22f506501833ddb8939c \
    --hash=sha256:206ed23aecd67c71daf5c02c3cd19c0501b01ef3cbf7782db9e4e051426b3d0d \
    --hash=sha256:2248485b0322c75aee7565d95ad0e16f1c67403a470d02f94da7344184be770f \
    --hash=sha256:27548e16c79702f1e03f5628589c6057c9ae17c95b4c449de3c66b589ead0520 \
    --hash=sha256:2d0ae0d8670164e10accbeb31d5ad45adb71292032d0fdb9079912907f0085f4 \
    --hash=sha256:3128e0bbc8c091ec4375a1828d6118bc20404883169ac95ffa8d983b293611e6 \
    --hash=sha256:3387277f1bf659caf1724e1afe8ee7dbc9952a82d90f858ebb931880216ea955 \
    --hash=sha256:34708cc82c330e303f4ce87758828ef6e457681b58ce0e921b6e97937dd1e2a3 \
    --hash=sha256:35613015f0ba7e14c29ac6c2483a657ec740e5ac5758d993fdd5870b07a61d8b \
    --hash=sha256:3ad873900297bb36e4b6b3f7029d88ff9829ecdc15d5cf20161775ce12306f8a \
    --hash=sha256:40180930807ce806aa71eda5a5a5447abb6b6a3c0b4b3b1b1962651906484d68 \
    --hash=sha256:439c9afe34638ace43a49bf72d201e0ffc1a800295bed8420c2a9ca8d5e3dbb3 \
    --hash=sha256:45e95333b8418ded64745f14574aa9bfc212cb4fbeed7a687b0c6e53b5e188cd \
    --hash=sha256:4641e8ad4efb697f38a9b64ca0523b557c7931c5f84e0fd377a9a3b05121f0de \
    --hash=sha256:49b08aae5013640a3bfa25a8eebbd95638ec3f4b2eaf6ed82cf0c7047133f03b \
    --hash=sha256:4bc536201426451f06f044dfbf341c09f540b4ebdb9fd8d2c6164d733de5e634 \
    --hash=sha256:4ce601907e99ea5b4adb807ded3570ea62186b17f88e271569144e8cca4409c7 \
    --hash=sha256:4e40f2bd0d57dac3feb3a3aed50f17d83436c9e6b09b16af271b6230a2915459 \
    --hash=sha256:4e47a76848f92529879ecfc417ff88a2806438f57be4a6a8bf2961e8f9ca9ec7 \
    --hash=sha256:513b07e99c0a267b1d954243845d8a833758a6726a3b5d8948306e3fe14675e3 \
    --hash=sha256:531f4b4252fac6ca476fbe0e6f60f16f5b65d3e6b583bc4d87645e4e5ddde331 \
    --hash=sha256:57d52fa717ff445cb0a5ab5237db502e6be50809b43a596fb569630c665abddf \
    --hash=sha256:59986de5710ad9613ff61dd9b02bdd2f615f1a7052304b79cc8fa2eb4e336d2d \
    --hash=sha256:5baab5455c7a538ac7e8bf1feec4278a66436197592a9bed538160a2e7d11e36 \
    --hash=sha256:5c7d5b5005f177764e96bd584d7bf28d6e26e96f2a541fdddb934c486e36fd59 \
    --hash=sha256:60b7607753ba62cf0739177913b858140f11b8af72f22860c28eabb2f0a61937 \
    --hash=sha256:615a0a4bff11c45eb3c1996ceed5bdaa2f7b432425253a7c2eed33bb86d80abc \
    --hash=sha256:61ea96a78378e3bd5a0be99b0e5ed00057b71f66115f5404d0dae4819f495093 \
    --hash=sha256:652c1988019752138b974c28f43751528116bcceadad85f33a258869e641d753 \
    --hash=sha256:6637560562134b0e17de333d18e69e312e0458ee4455bdad12c37100b7cad706 \
    --hash=sha256:678265f7b14e138d9a541ddabbe033012a2953315739f8cfa6d754cc8063e8ca \
    --hash=sha256:699156034181e2ce106c89ddb4b6504c30db8caa86e0c30de47b3e0654543260 \
    --hash=sha256:6b9ff467ffbab9110e80e8c8de3bcfce8e8b0fd5661ac44a09ae5901668ba997 \
    --hash=sha256:6c327e9cd849b564b234da821236e6bcbe4f359a42ee05050dc79d8ed2a91588 \
    --hash=sha256:6d30226dfc816dd0fdf120cae611dd2215117e4f9b124af8c60ab9093b6e8e71 \
    --hash=sha256:6e227c40c02fd873c2a73a98c1280c10315cbebe26734c196ef4514776120aeb \
    --hash=sha256:6e4d090e73e0725b2904fdbdd8d73b8802ddd691ef9254577b708d413bf3006e \
    --hash=sha256:70f4b4851dbb500129681d04cc955be2a90b2248d69273a787dda120d5cf1f69 \
    --hash=sha256:70f947628e074bb2526ba1b151cee10e4c3b9670af4dbb4d73bc8a89445916b5 \
    --hash=sha256:774de879d212db5ce02dfbf5b0da9a0ea386aeba12b0b95674a4ce0593df3d07 \
    --hash=sha256:77fa384d8e118b3077cccfcaf91bf83c31fe4dc850b5e6ee3dc14dc3d61bdba1 \
    --hash=sha256:79e0a2cdbdc7af3f4aee3210b1172ab53d7ddb6a2d8c24119b5706e622b346d0 \
    --hash=sha256:7e88f5696153dc516ba6e79f82cc4747e87027205f0e02390c21f7cb3bd8abfd \
    --hash=sha256:7f8210297b04e53bc3da35db08b7302a6a1f4889c79173af69b72ec9754796b8 \
    --hash=sha256:81982d78a45d1e5396819bbb4ece1fadfe5f079335dd28c4ab3427cd95389944 \
    --hash=sha256:823fcc638f67035137a5cd3f1584a4542d35a951c3cc68c6ead1df7dac825c26 \
    --hash=sha256:853a2295c00f1d4429db4c0fb9475958543ee80cfd310814b5c0ef502de24dda \
    --hash=sha256:88e74ab0cdd84ad0614e2750f903bb0d610cc8af2cc17f72c28163acfcf372a4 \
    --hash=sha256:8aa1768c151cf562a9992462239dfc356b3d1037cc5a3ac829bb7f3bda7cc1f9 \
    --hash=sha256:8c8a8812fe6f43a3a5b054af6ac2d7b8605c7bcab2804a8a7d68b53f3cd86e00 \
    --hash=sha256:95b15e855ae44f0c6341ceb74df61b606e11f1087e87dcb7482377374aac6abe \
    --hash=sha256:96581cfefa9123accc465a5fd0cc833ac4d75d55cc30b633b402e00e7ced00a6 \
    --hash=sha256:9bd18fee0923ca10f9a3ff67d4851c9d3e22b7bc63d1eddc12f439f436f2aada \
    --hash=sha256:a33324437018bf6ba1bb0f921788788641439e0ed654b233285b9c69704c27b4 \
    --hash=sha256:a6a16f4a527aae4f49c875da3cdc9508ac7eef26e7977952608610104244e1b7 \
    --hash=sha256:a717aef6971208f0851a2420b075338e33083111d92041157bbe0e2713b37325 \
    --hash=sha256:a71891847f0a73b1b9eb86d089baee301477abef45f7eaf303495cd1473613e4 \
    --hash=sha256:aae7ea3a1c5bb40c93cad361b3e869b180ac174656120c42b9fadebf685d121b \
    --hash=sha256:ab1cdb0f14dc161ebc268c09db04d2c9e6f70027f3b42446fa11c153521c0e88 \
    --hash=sha256:ab4ea451082e684198636565224bbb179575efc1658c48281b2c866bfd4ddf04 \
    --hash=sha256:abf058be9517dc877227ec3223f0300034bd0e9f53aebd63cf4456c8cb1e0863 \
    --hash=sha256:af36f36538418f3806048f3b242a1777e2540ff9efaa667c27da63d2749dbce0 \
    --hash=sha256:b53e9ad053cd064f7e473a5f29b37fc4cc9dc6d35f341e6afc0155ea257fc911 \
    --hash=sha256:b7851992faf25eac90bfcb7bfd19e1f5ffa00afd57daec8a0042e63c74a4551b \
    --hash=sha256:b9b759b77f5337b4ea024f03abc6464c9f35d9718de01cfe6bae9f2e139c397e \
    --hash=sha256:ba39688799094c75ea8a16a6b544eb57b5b0f3328697084f3f2790892510d144 \
    --hash=sha256:ba6b6b3846cfc10fdb4c971980a954e49d447cd215ed5a77ec8190bc93dd7bc5 \
    --hash=sha256:bb4c2eda937a5e74c38a41b33d8c77220380a388d689bcdb9b187cf6224c9720 \
    --hash=sha256:c0b97ec434041827935044bbbe52b03d6018c2897349670ff8fe11ed24d1d4ab \
    --hash=sha256:c1452a1acdf914d194159439eb21e56b89aa903f2e1c65c60b9d874f9b950e5d \
    --hash=sha256:c2027d05c8aebe61d898d4cffd774840a9cb82ed356ba47a90d99ad768f39789 \
    --hash=sha256:c2adbe22ab4babbca99c75c5d07aaf74f43c3195384ec07ccbd2f9e3bddaecec \
    --hash=sha256:c2d97e906b4ff36eb464d52a3bc7d720bd6261f64bc4bcdbcd2c557c02081ed2 \
    --hash=sha256:c339dabd8ee15f8259ee0f202679b6324926e5bc9e9a40bf981ce77c038553db \
    --hash=sha256:c6eae413494a1c3f89055da7a5515f32e05ebc1a234c27674a6956755fb2236f \
    --hash=sha256:c949f04ecad823f81b1ba94e7d189d9dfb81edbb94ed3f8acfce41e682e48cef \
    --hash=sha256:c97bee68898f3f4344eb02fec316db93d9700fb1e6a5b760ffa20d71d9a46ce3 \
    --hash=sha256:ca61d858e4107ce5e1330a74724fe757fc7135190eb5ce5c9d0191729f033209 \
    --hash=sha256:cb4679d4c2b089e5ef89756bc73e1926745e995d76e11925e3e96a76d5fa51fc \
    --hash=sha256:cb774298da62aea5c80a89bd58c40205ab4c2abf4834453b5de207d59d2e1651 \
    --hash=sha256:ccd4d5702bb90b84df13bd491be8d900b92016c5a455b7e14630ad7449eb03f8 \
    --hash=sha256:cf9d3fe53b1ee360e2421be95e62ca9b3296bf3f2fb2d3b83ca49ad3f925835e \
    --hash=sha256:d2ae91f50ccc5810b2f1b6b858257c9ad2e08da70bf890dee02de1775a387c66 \
    --hash=sha256:d37f8ec982ead9ba0a22a996129594938138a1503237b87318392a48882d50b7 \
    --hash=sha256:d81e6987b27bc7d101c8597e1cd2bcaa2fee5e8e0f356735c7ed34368c471550 \
    --hash=sha256:dcf4e6d85614f7a4956c2de5a56531f44efb973d2fe4a444d7251df5d5c4dcfd \
    --hash=sha256:de790a3b5aa2124b8b78ae5faa033937a72da8efe74b9231698b5a1dd9be3405 \
    --hash=sha256:e47e9a08bcc04d20975b6434cc50bf82665fbc751bcce739d04a3120428f3e27 \
    --hash=sha256:e60f112ac88db9261ad3a52032ea46388378034f3279c643499edb982536a093 \
    --hash=sha256:e87fc540c6cac7f29ede02e0f989d4233f88ad439c5cdee56f693cc9c1c78077 \
    --hash=sha256:eac5c82fc632c599f4639a5886f96867ffced74458c7db61bc9a66ccb8ee3113 \
    --hash=sha256:ebb4e035e28f49b6f1a7032920bb9a0c064aedbbabe52c543343d39341a5b2a3 \
    --hash=sha256:ec1e72d6412f7126eb7b2e3bfca42b15e6e389e1bc88ea0069d0cc1742f477c6 \
    --hash=sha256:ef98ca7d5995a82f43ec0ab39c4caf6a9b994cb0b53648ff61716370eadc43cf \
    --hash=sha256:f0cbc7fff06a90bbd875cc201f94ef0ee3929dfbd5c55a06674b60857b8b85ed \
    --hash=sha256:f4791cf0f8c3104ac668797d8c514afb3431bc3305f5638add0ba1a5a37e0d88 \
    --hash=sha256:f5e412d717366e0677ef767eac93566582518fe8be923361a5c204c1a62eaafe \
    --hash=sha256:fb2ed8b3fe4bf4506d6dab3b93b83bbc22237e230cba03866d561c3577517d18 \
    --hash=sha256:fe0a5a1025eb797752136ac8b4fa21aa891e3d74fd340f864ff982d649691867
pydantic==2.5.2 ; python_full_version == "3.10.13" \
    --hash=sha256:80c50fb8e3dcecfddae1adbcc00ec5822918490c99ab31f6cf6140ca1c1429f0 \
    --hash=sha256:ff177ba64c6faf73d7afa2e8cad38fd456c0dbe01c9954e71038001cd15a6edd
pygments==2.17.2 ; python_full_version == "3.10.13" \
    --hash=sha256:b27c2826c47d0f3219f29554824c30c5e8945175d888647acd804ddd04af846c \
    --hash=sha256:da46cec9fd2de5be3a8a784f434e4c4ab670b4ff54d605c4c2717e9d49c4c367
pyjwt==2.8.0 ; python_full_version == "3.10.13" \
    --hash=sha256:57e28d156e3d5c10088e0c68abb90bfac3df82b40a71bd0daa20c65ccd5c23de \
    --hash=sha256:59127c392cc44c2da5bb3192169a91f429924e17aff6534d70fdc02ab3e04320
pyparsing==3.1.1 ; python_full_version == "3.10.13" \
    --hash=sha256:32c7c0b711493c72ff18a981d24f28aaf9c1fb7ed5e9667c9e84e3db623bdbfb \
    --hash=sha256:ede28a1a32462f5a9705e07aea48001a08f7cf81a021585011deba701581a0db
pytest==7.4.3 ; python_full_version == "3.10.13" \
    --hash=sha256:0d009c083ea859a71b76adf7c1d502e4bc170b80a8ef002da5806527b9591fac \
    --hash=sha256:d989d136982de4e3b29dabcc838ad581c64e8ed52c11fbe86ddebd9da0818cd5
python-dateutil==2.8.2 ; python_full_version == "3.10.13" \
    --hash=sha256:0123cacc1627ae19ddf3c27a5de5bd67ee4586fbdd6440d9748f8abb483d3e86 \
    --hash=sha256:961d03dc3453ebbc59dbdea9e4e11c5651520a876d0f4db161e8674aae935da9
python-dotenv==1.0.0 ; python_full_version == "3.10.13" \
    --hash=sha256:a8df96034aae6d2d50a4ebe8216326c61c3eb64836776504fcca410e5937a3ba \
    --hash=sha256:f5971a9226b701070a4bf2c38c89e5a3f0d64de8debda981d1db98583009122a
pytz==2023.3.post1 ; python_full_version == "3.10.13" \
    --hash=sha256:7b4fddbeb94a1eba4b557da24f19fdf9db575192544270a9101d8509f9f43d7b \
    --hash=sha256:ce42d816b81b68506614c11e8937d3aa9e41007ceb50bfdcb0749b921bf646c7
pyyaml==6.0.1 ; python_full_version == "3.10.13" \
    --hash=sha256:04ac92ad1925b2cff1db0cfebffb6ffc43457495c9b3c39d3fcae417d7125dc5 \
    --hash=sha256:062582fca9fabdd2c8b54a3ef1c978d786e0f6b3a1510e0ac93ef59e0ddae2bc \
    --hash=sha256:0d3304d8c0adc42be59c5f8a4d9e3d7379e6955ad754aa9d6ab7a398b59dd1df \
    --hash=sha256:1635fd110e8d85d55237ab316b5b011de701ea0f29d07611174a1b42f1444741 \
    --hash=sha256:184c5108a2aca3c5b3d3bf9395d50893a7ab82a38004c8f61c258d4428e80206 \
    --hash=sha256:18aeb1bf9a78867dc38b259769503436b7c72f7a1f1f4c93ff9a17de54319b27 \
    --hash=sha256:1d4c7e777c441b20e32f52bd377e0c409713e8bb1386e1099c2415f26e479595 \
    --hash=sha256:1e2722cc9fbb45d9b87631ac70924c11d3a401b2d7f410cc0e3bbf249f2dca62 \
    --hash=sha256:1fe35611261b29bd1de0070f0b2f47cb6ff71fa6595c077e42bd0c419fa27b98 \
    --hash=sha256:28c119d996beec18c05208a8bd78cbe4007878c6dd15091efb73a30e90539696 \
    --hash=sha256:326c013efe8048858a6d312ddd31d56e468118ad4cdeda36c719bf5bb6192290 \
    --hash=sha256:40df9b996c2b73138957fe23a16a4f0ba614f4c0efce1e9406a184b6d07fa3a9 \
    --hash=sha256:42f8152b8dbc4fe7d96729ec2b99c7097d656dc1213a3229ca5383f973a5ed6d \
    --hash=sha256:49a183be227561de579b4a36efbb21b3eab9651dd81b1858589f796549873dd6 \
    --hash=sha256:4fb147e7a67ef577a588a0e2c17b6db51dda102c71de36f8549b6816a96e1867 \
    --hash=sha256:50550eb667afee136e9a77d6dc71ae76a44df8b3e51e41b77f6de2932bfe0f47 \
    --hash=sha256:510c9deebc5c0225e8c96813043e62b680ba2f9c50a08d3724c7f28a747d1486 \
    --hash=sha256:5773183b6446b2c99bb77e77595dd486303b4faab2b086e7b17bc6bef28865f6 \
    --hash=sha256:596106435fa6ad000c2991a98fa58eeb8656ef2325d7e158344fb33864ed87e3 \
    --hash=sha256:6965a7bc3cf88e5a1c3bd2e0b5c22f8d677dc88a455344035f03399034eb3007 \
    --hash=sha256:69b023b2b4daa7548bcfbd4aa3da05b3a74b772db9e23b982788168117739938 \
    --hash=sha256:6c22bec3fbe2524cde73d7ada88f6566758a8f7227bfbf93a408a9d86bcc12a0 \
    --hash=sha256:704219a11b772aea0d8ecd7058d0082713c3562b4e271b849ad7dc4a5c90c13c \
    --hash=sha256:7e07cbde391ba96ab58e532ff4803f79c4129397514e1413a7dc761ccd755735 \
    --hash=sha256:81e0b275a9ecc9c0c0c07b4b90ba548307583c125f54d5b6946cfee6360c733d \
    --hash=sha256:855fb52b0dc35af121542a76b9a84f8d1cd886ea97c84703eaa6d88e37a2ad28 \
    --hash=sha256:8d4e9c88387b0f5c7d5f281e55304de64cf7f9c0021a3525bd3b1c542da3b0e4 \
    --hash=sha256:9046c58c4395dff28dd494285c82ba00b546adfc7ef001486fbf0324bc174fba \
    --hash=sha256:9eb6caa9a297fc2c2fb8862bc5370d0303ddba53ba97e71f08023b6cd73d16a8 \
    --hash=sha256:a0cd17c15d3bb3fa06978b4e8958dcdc6e0174ccea823003a106c7d4d7899ac5 \
    --hash=sha256:afd7e57eddb1a54f0f1a974bc4391af8bcce0b444685d936840f125cf046d5bd \
    --hash=sha256:b1275ad35a5d18c62a7220633c913e1b42d44b46ee12554e5fd39c70a243d6a3 \
    --hash=sha256:b786eecbdf8499b9ca1d697215862083bd6d2a99965554781d0d8d1ad31e13a0 \
    --hash=sha256:ba336e390cd8e4d1739f42dfe9bb83a3cc2e80f567d8805e11b46f4a943f5515 \
    --hash=sha256:baa90d3f661d43131ca170712d903e6295d1f7a0f595074f151c0aed377c9b9c \
    --hash=sha256:bc1bf2925a1ecd43da378f4db9e4f799775d6367bdb94671027b73b393a7c42c \
    --hash=sha256:bd4af7373a854424dabd882decdc5579653d7868b8fb26dc7d0e99f823aa5924 \
    --hash=sha256:bf07ee2fef7014951eeb99f56f39c9bb4af143d8aa3c21b1677805985307da34 \
    --hash=sha256:bfdf460b1736c775f2ba9f6a92bca30bc2095067b8a9d77876d1fad6cc3b4a43 \
    --hash=sha256:c8098ddcc2a85b61647b2590f825f3db38891662cfc2fc776415143f599bb859 \
    --hash=sha256:d2b04aac4d386b172d5b9692e2d2da8de7bfb6c387fa4f801fbf6fb2e6ba4673 \
    --hash=sha256:d483d2cdf104e7c9fa60c544d92981f12ad66a457afae824d146093b8c294c54 \
    --hash=sha256:d858aa552c999bc8a8d57426ed01e40bef403cd8ccdd0fc5f6f04a00414cac2a \
    --hash=sha256:e7d73685e87afe9f3b36c799222440d6cf362062f78be1013661b00c5c6f678b \
    --hash=sha256:f003ed9ad21d6a4713f0a9b5a7a0a79e08dd0f221aff4525a2be4c346ee60aab \
    --hash=sha256:f22ac1c3cac4dbc50079e965eba2c1058622631e526bd9afd45fedd49ba781fa \
    --hash=sha256:faca3bdcf85b2fc05d06ff3fbc1f83e1391b3e724afa3feba7d13eeab355484c \
    --hash=sha256:fca0e3a251908a499833aa292323f32437106001d436eca0e6e7833256674585 \
    --hash=sha256:fd1592b3fdf65fff2ad0004b5e363300ef59ced41c2e6b3a99d4089fa8c5435d \
    --hash=sha256:fd66fc5d0da6d9815ba2cebeb4205f95818ff4b79c3ebe268e75d961704af52f
regex==2023.10.3 ; python_full_version == "3.10.13" \
    --hash=sha256:00ba3c9818e33f1fa974693fb55d24cdc8ebafcb2e4207680669d8f8d7cca79a \
    --hash=sha256:00e871d83a45eee2f8688d7e6849609c2ca2a04a6d48fba3dff4deef35d14f07 \
    --hash=sha256:06e9abc0e4c9ab4779c74ad99c3fc10d3967d03114449acc2c2762ad4472b8ca \
    --hash=sha256:0b9ac09853b2a3e0d0082104036579809679e7715671cfbf89d83c1cb2a30f58 \
    --hash=sha256:0d47840dc05e0ba04fe2e26f15126de7c755496d5a8aae4a08bda4dd8d646c54 \
    --hash=sha256:0f649fa32fe734c4abdfd4edbb8381c74abf5f34bc0b3271ce687b23729299ed \
    --hash=sha256:107ac60d1bfdc3edb53be75e2a52aff7481b92817cfdddd9b4519ccf0e54a6ff \
    --hash=sha256:11175910f62b2b8c055f2b089e0fedd694fe2be3941b3e2633653bc51064c528 \
    --hash=sha256:12bd4bc2c632742c7ce20db48e0d99afdc05e03f0b4c1af90542e05b809a03d9 \
    --hash=sha256:16f8740eb6dbacc7113e3097b0a36065a02e37b47c936b551805d40340fb9971 \
    --hash=sha256:1c0e8fae5b27caa34177bdfa5a960c46ff2f78ee2d45c6db15ae3f64ecadde14 \
    --hash=sha256:2c54e23836650bdf2c18222c87f6f840d4943944146ca479858404fedeb9f9af \
    --hash=sha256:3367007ad1951fde612bf65b0dffc8fd681a4ab98ac86957d16491400d661302 \
    --hash=sha256:36362386b813fa6c9146da6149a001b7bd063dabc4d49522a1f7aa65b725c7ec \
    --hash=sha256:39807cbcbe406efca2a233884e169d056c35aa7e9f343d4e78665246a332f597 \
    --hash=sha256:39cdf8d141d6d44e8d5a12a8569d5a227f645c87df4f92179bd06e2e2705e76b \
    --hash=sha256:3b2c3502603fab52d7619b882c25a6850b766ebd1b18de3df23b2f939360e1bd \
    --hash=sha256:3ccf2716add72f80714b9a63899b67fa711b654be3fcdd34fa391d2d274ce767 \
    --hash=sha256:3fef4f844d2290ee0ba57addcec17eec9e3df73f10a2748485dfd6a3a188cc0f \
    --hash=sha256:4023e2efc35a30e66e938de5aef42b520c20e7eda7bb5fb12c35e5d09a4c43f6 \
    --hash=sha256:4a3ee019a9befe84fa3e917a2dd378807e423d013377a884c1970a3c2792d293 \
    --hash=sha256:4a8bf76e3182797c6b1afa5b822d1d5802ff30284abe4599e1247be4fd6b03be \
    --hash=sha256:4a992f702c9be9c72fa46f01ca6e18d131906a7180950958f766c2aa294d4b41 \
    --hash=sha256:4c34d4f73ea738223a094d8e0ffd6d2c1a1b4c175da34d6b0de3d8d69bee6bcc \
    --hash=sha256:4cd1bccf99d3ef1ab6ba835308ad85be040e6a11b0977ef7ea8c8005f01a3c29 \
    --hash=sha256:4ef80829117a8061f974b2fda8ec799717242353bff55f8a29411794d635d964 \
    --hash=sha256:58837f9d221744d4c92d2cf7201c6acd19623b50c643b56992cbd2b745485d3d \
    --hash=sha256:5a8f91c64f390ecee09ff793319f30a0f32492e99f5dc1c72bc361f23ccd0a9a \
    --hash=sha256:5addc9d0209a9afca5fc070f93b726bf7003bd63a427f65ef797a931782e7edc \
    --hash=sha256:6239d4e2e0b52c8bd38c51b760cd870069f0bdf99700a62cd509d7a031749a55 \
    --hash=sha256:66e2fe786ef28da2b28e222c89502b2af984858091675044d93cb50e6f46d7af \
    --hash=sha256:69c0771ca5653c7d4b65203cbfc5e66db9375f1078689459fe196fe08b7b4930 \
    --hash=sha256:6ac965a998e1388e6ff2e9781f499ad1eaa41e962a40d11c7823c9952c77123e \
    --hash=sha256:6c56c3d47da04f921b73ff9415fbaa939f684d47293f071aa9cbb13c94afc17d \
    --hash=sha256:6f85739e80d13644b981a88f529d79c5bdf646b460ba190bffcaf6d57b2a9863 \
    --hash=sha256:706e7b739fdd17cb89e1fbf712d9dc21311fc2333f6d435eac2d4ee81985098c \
    --hash=sha256:741ba2f511cc9626b7561a440f87d658aabb3d6b744a86a3c025f866b4d19e7f \
    --hash=sha256:7434a61b158be563c1362d9071358f8ab91b8d928728cd2882af060481244c9e \
    --hash=sha256:76066d7ff61ba6bf3cb5efe2428fc82aac91802844c022d849a1f0f53820502d \
    --hash=sha256:7979b834ec7a33aafae34a90aad9f914c41fd6eaa8474e66953f3f6f7cbd4368 \
    --hash=sha256:7eece6fbd3eae4a92d7c748ae825cbc1ee41a89bb1c3db05b5578ed3cfcfd7cb \
    --hash=sha256:7ef1e014eed78ab650bef9a6a9cbe50b052c0aebe553fb2881e0453717573f52 \
    --hash=sha256:81dce2ddc9f6e8f543d94b05d56e70d03a0774d32f6cca53e978dc01e4fc75b8 \
    --hash=sha256:82fcc1f1cc3ff1ab8a57ba619b149b907072e750815c5ba63e7aa2e1163384a4 \
    --hash=sha256:8d1f21af4c1539051049796a0f50aa342f9a27cde57318f2fc41ed50b0dbc4ac \
    --hash=sha256:90a79bce019c442604662d17bf69df99090e24cdc6ad95b18b6725c2988a490e \
    --hash=sha256:9145f092b5d1977ec8c0ab46e7b3381b2fd069957b9862a43bd383e5c01d18c2 \
    --hash=sha256:91dc1d531f80c862441d7b66c4505cd6ea9d312f01fb2f4654f40c6fdf5cc37a \
    --hash=sha256:979c24cbefaf2420c4e377ecd1f165ea08cc3d1fbb44bdc51bccbbf7c66a2cb4 \
    --hash=sha256:994645a46c6a740ee8ce8df7911d4aee458d9b1bc5639bc968226763d07f00fa \
    --hash=sha256:9b98b7681a9437262947f41c7fac567c7e1f6eddd94b0483596d320092004533 \
    --hash=sha256:9c6b4d23c04831e3ab61717a707a5d763b300213db49ca680edf8bf13ab5d91b \
    --hash=sha256:9c6d0ced3c06d0f183b73d3c5920727268d2201aa0fe6d55c60d68c792ff3588 \
    --hash=sha256:9fd88f373cb71e6b59b7fa597e47e518282455c2734fd4306a05ca219a1991b0 \
    --hash=sha256:a8f4e49fc3ce020f65411432183e6775f24e02dff617281094ba6ab079ef0915 \
    --hash=sha256:a9e908ef5889cda4de038892b9accc36d33d72fb3e12c747e2799a0e806ec841 \
    --hash=sha256:ad08a69728ff3c79866d729b095872afe1e0557251da4abb2c5faff15a91d19a \
    --hash=sha256:adbccd17dcaff65704c856bd29951c58a1bd4b2b0f8ad6b826dbd543fe740988 \
    --hash=sha256:b0c7d2f698e83f15228ba41c135501cfe7d5740181d5903e250e47f617eb4292 \
    --hash=sha256:b3ab05a182c7937fb374f7e946f04fb23a0c0699c0450e9fb02ef567412d2fa3 \
    --hash=sha256:b6104f9a46bd8743e4f738afef69b153c4b8b592d35ae46db07fc28ae3d5fb7c \
    --hash=sha256:ba7cd6dc4d585ea544c1412019921570ebd8a597fabf475acc4528210d7c4a6f \
    --hash=sha256:bc72c231f5449d86d6c7d9cc7cd819b6eb30134bb770b8cfdc0765e48ef9c420 \
    --hash=sha256:bce8814b076f0ce5766dc87d5a056b0e9437b8e0cd351b9a6c4e1134a7dfbda9 \
    --hash=sha256:be5e22bbb67924dea15039c3282fa4cc6cdfbe0cbbd1c0515f9223186fc2ec5f \
    --hash=sha256:be6b7b8d42d3090b6c80793524fa66c57ad7ee3fe9722b258aec6d0672543fd0 \
    --hash=sha256:bfe50b61bab1b1ec260fa7cd91106fa9fece57e6beba05630afe27c71259c59b \
    --hash=sha256:bff507ae210371d4b1fe316d03433ac099f184d570a1a611e541923f78f05037 \
    --hash=sha256:c148bec483cc4b421562b4bcedb8e28a3b84fcc8f0aa4418e10898f3c2c0eb9b \
    --hash=sha256:c15ad0aee158a15e17e0495e1e18741573d04eb6da06d8b84af726cfc1ed02ee \
    --hash=sha256:c2169b2dcabf4e608416f7f9468737583ce5f0a6e8677c4efbf795ce81109d7c \
    --hash=sha256:c55853684fe08d4897c37dfc5faeff70607a5f1806c8be148f1695be4a63414b \
    --hash=sha256:c65a3b5330b54103e7d21cac3f6bf3900d46f6d50138d73343d9e5b2900b2353 \
    --hash=sha256:c7964c2183c3e6cce3f497e3a9f49d182e969f2dc3aeeadfa18945ff7bdd7051 \
    --hash=sha256:cc3f1c053b73f20c7ad88b0d1d23be7e7b3901229ce89f5000a8399746a6e039 \
    --hash=sha256:ce615c92d90df8373d9e13acddd154152645c0dc060871abf6bd43809673d20a \
    --hash=sha256:d29338556a59423d9ff7b6eb0cb89ead2b0875e08fe522f3e068b955c3e7b59b \
    --hash=sha256:d8a993c0a0ffd5f2d3bda23d0cd75e7086736f8f8268de8a82fbc4bd0ac6791e \
    --hash=sha256:d9c727bbcf0065cbb20f39d2b4f932f8fa1631c3e01fcedc979bd4f51fe051c5 \
    --hash=sha256:dac37cf08fcf2094159922edc7a2784cfcc5c70f8354469f79ed085f0328ebdf \
    --hash=sha256:dd829712de97753367153ed84f2de752b86cd1f7a88b55a3a775eb52eafe8a94 \
    --hash=sha256:e54ddd0bb8fb626aa1f9ba7b36629564544954fff9669b15da3610c22b9a0991 \
    --hash=sha256:e77c90ab5997e85901da85131fd36acd0ed2221368199b65f0d11bca44549711 \
    --hash=sha256:ebedc192abbc7fd13c5ee800e83a6df252bec691eb2c4bedc9f8b2e2903f5e2a \
    --hash=sha256:ef71561f82a89af6cfcbee47f0fabfdb6e63788a9258e913955d89fdd96902ab \
    --hash=sha256:f0a47efb1dbef13af9c9a54a94a0b814902e547b7f21acb29434504d18f36e3a \
    --hash=sha256:f4f2ca6df64cbdd27f27b34f35adb640b5d2d77264228554e68deda54456eb11 \
    --hash=sha256:fb02e4257376ae25c6dd95a5aec377f9b18c09be6ebdefa7ad209b9137b73d48
```

```
requests-oauthlib==1.3.1 ; python_full_version == "3.10.13" \
    --hash=sha256:2577c501a2fb8d05a304c09d090d6e47c306fef15809d102b327cf8364bddab5 \
    --hash=sha256:75beac4a47881eeb94d5ea5d6ad31ef88856affe2332b9aafb52c6452ccf0d7a
requests==2.31.0 ; python_full_version == "3.10.13" \
    --hash=sha256:58cd2187c01e70e6e26505bca751777aa9f2ee0b7f4300988b709f44e013003f \
    --hash=sha256:942c5a758f98d790eaed1a29cb6eefc7ffb0d1cf7af05c3d2791656dbd6ad1e1
rich==13.7.0 ; python_full_version == "3.10.13" \
    --hash=sha256:5cb5123b5cf9ee70584244246816e9114227e0b98ad9176eede6ad54bf5403fa \
    --hash=sha256:6da14c108c4866ee9520bbffa71f6fe3962e193b7da68720583850cd4548e235
rsa==4.9 ; python_full_version == "3.10.13" \
    --hash=sha256:90260d9058e514786967344d0ef75fa8727eed8a7d2e43ce9f4bcf1b536174f7 \
    --hash=sha256:e38464a49c6c85d7f1351b0126661487a7e0a14a50f1675ec50eb34d4f20ef21
safetensors==0.4.1 ; python_full_version == "3.10.13" \
    --hash=sha256:04157d008385bea66d12fe90844a80d4a76dc25ec5230b5bd9a630496d1b7c03 \
    --hash=sha256:04dd14f53f5500eb4c4149674216ba1000670efbcf4b1b5c2643eb244e7882ea \
    --hash=sha256:097e9af2efa8778cd2f0cba451784253e62fa7cc9fc73c0744d27212f7294e25 \
    --hash=sha256:0bd0afd95c1e497f520e680ea01e0397c0868a3a3030e128438cf6e9e3fcd671 \
    --hash=sha256:0ddd050e01f3e843aa8c1c27bf68675b8a08e385d0045487af4d70418c3cb356 \
    --hash=sha256:16d8bbb7344e39cb9d4762e85c21df94ebeb03edac923dd94bb9ed8c10eac070 \
    --hash=sha256:1a45dbf03e8334d3a5dc93687d98b6dc422f5d04c7d519dac09b84a3c87dd7c6 \
    --hash=sha256:1d568628e9c43ca15eb96c217da73737c9ccb07520fafd8a1eba3f2750614105 \
    --hash=sha256:1faf5111c66a6ba91f85dff2e36edaaf36e6966172703159daeef330de4ddc7b \
    --hash=sha256:2297b359d91126c0f9d4fd17bae3cfa2fe3a048a6971b8db07db746ad92f850c \
    --hash=sha256:2304658e6ada81a5223225b4efe84748e760c46079bffedf7e321763cafb36c9 \
    --hash=sha256:2536b11ce665834201072e9397404170f93f3be10cca9995b909f023a04501ee \
    --hash=sha256:257d59e40a1b367cb544122e7451243d65b33c3f34d822a347f4eea6fdf97fdf \
    --hash=sha256:25a043cbb59d4f75e9dd87fdf5c009dd8830105a2c57ace49b72167dd9808111 \
    --hash=sha256:270b99885ec14abfd56c1d7f28ada81740a9220b4bae960c3de1c6fe84af9e4d \
    --hash=sha256:285b52a481e7ba93e29ad4ec5841ef2c4479ef0a6c633c4e2629e0508453577b \
    --hash=sha256:2b6a2814278b6660261aa9a9aae524616de9f1ec364e3716d219b6ed8f91801f \
    --hash=sha256:2d54c2f1826e790d1eb2d2512bfd0ee443f0206b423d6f27095057c7f18a0687 \
    --hash=sha256:2d87d993eaefe6611a9c241a8bd364a5f1ffed5771c74840363a6c4ed8d868f6 \
    --hash=sha256:2fe6926110e3d425c4b684a4379b7796fdc26ad7d16922ea1696c8e6ea7e920f \
    --hash=sha256:303d2c0415cf15a28f8d7f17379ea3c34c2b466119118a34edd9965983a1a8a6 \
    --hash=sha256:313e8472197bde54e3ec54a62df184c414582979da8f3916981b6a7954910a1b \
    --hash=sha256:35803201d980efcf964b75a0a2aee97fe5e9ecc5f3ad676b38fafdfe98e0620d \
    --hash=sha256:39d36f1d88468a87c437a1bc27c502e71b6ca44c385a9117a9f9ba03a75cc9c6 \
    --hash=sha256:3b0b7b2d5976fbed8a05e2bbdce5816a59e6902e9e7c7e07dc723637ed539787 \
    --hash=sha256:3b30abd0cddfe959d1daedf92edcd1b445521ebf7ddefc20860ed01486b33c90 \
    --hash=sha256:3c1b1d510c7aba71504ece87bf393ea82638df56303e371e5e2cf09d18977dd7 \
    --hash=sha256:3cfd1ca35eacc635f0eaa894e5c5ed83ffebd0f95cac298fd430014fa7323631 \
    --hash=sha256:3f6a520af7f2717c5ecba112041f2c8af1ca6480b97bf957aba81ed9642e654c \
    --hash=sha256:413e1f6ac248f7d1b755199a06635e70c3515493d3b41ba46063dec33aa2ebb7 \
    --hash=sha256:4177b456c6b0c722d82429127b5beebdaf07149d265748e97e0a34ff0b3694c8 \
    --hash=sha256:42c3710cec7e5c764c7999697516370bee39067de0aa089b7e2cfb97ac8c6b20 \
    --hash=sha256:44e230fbbe120de564b64f63ef3a8e6ff02840fa02849d9c443d56252a1646d4 \
    --hash=sha256:48901bd540f8a3c1791314bc5c8a170927bf7f6acddb75bf0a263d081a3637d4 \
    --hash=sha256:53134226053e56bd56e73f7db42596e7908ed79f3c9a1016e4c1dade593ac8e5 \
    --hash=sha256:573b6023a55a2f28085fc0a84e196c779b6cbef4d9e73acea14c8094fee7686f \
    --hash=sha256:5d95ea4d8b32233910734a904123bdd3979c137c461b905a5ed32511defc075f \
    --hash=sha256:5f25297148ec665f0deb8bd67e9564634d8d6841041ab5393ccfe203379ea88b \
    --hash=sha256:645b3f1138fce6e818e79d4128afa28f0657430764cc045419c1d069ff93f732 \
    --hash=sha256:660ca1d8bff6c7bc7c6b30b9b32df74ef3ab668f5df42cefd7588f0d40feadcb \
    --hash=sha256:6ace9e66a40f98a216ad661245782483cf79cf56eb2b112650bb904b0baa9db5 \
    --hash=sha256:6fd80f7794554091836d4d613d33a7d006e2b8d6ba014d06f97cebdfda744f64 \
    --hash=sha256:780dc21eb3fd32ddd0e8c904bdb0290f2454f4ac21ae71e94f9ce72db1900a5a \
    --hash=sha256:791edc10a3c359a2f5f52d5cddab0df8a45107d91027d86c3d44e57162e5d934 \
    --hash=sha256:7a8f6f679d97ea0135c7935c202feefbd042c149aa70ee759855e890c01c7814 \
    --hash=sha256:7ef010e9afcb4057fb6be3d0a0cfa07aac04fe97ef73fe4a23138d8522ba7c17 \
    --hash=sha256:7ff8a36e0396776d3ed9a106fc9a9d7c55d4439ca9a056a24bf66d343041d3e6 \
    --hash=sha256:82571d20288c975c1b30b08deb9b1c3550f36b31191e1e81fae87669a92217d0 \
    --hash=sha256:82cbb8f4d022f2e94498cbefca900698b8ded3d4f85212f47da614001ff06652 \
    --hash=sha256:83c2cfbe8c6304f0891e7bb378d56f66d2148972eeb5f747cd8a2246886f0d8c \
    --hash=sha256:845be0aafabf2a60c2d482d4e93023fecffe5e5443d801d7a7741bae9de41233 \
    --hash=sha256:88b4653059c903015284a9722f9a46838c654257173b279c8f6f46dbe80b612d \
    --hash=sha256:8b58ba13a9e82b4bc3fc221914f6ef237fe6c2adb13cede3ace64d1aacf49610 \
    --hash=sha256:8f69903ff49cb30b9227fb5d029bea276ea20d04b06803877a420c5b1b74c689 \
    --hash=sha256:8ff8e41c8037db17de0ea2a23bc684f43eaf623be7d34906fe1ac10985b8365e \
    --hash=sha256:911b48dc09e321a194def3a7431662ff4f03646832f3a8915bbf0f449b8a5fcb \
    --hash=sha256:998fbac99ca956c3a09fe07cc0b35fac26a521fa8865a690686d889f0ff4e4a6 \
    --hash=sha256:9a82bc2bd7a9a0e08239bdd6d7774d64121f136add93dfa344a2f1a6d7ef35fa \
    --hash=sha256:9d16b3b2fcc6fca012c74bd01b5619c655194d3e3c13e4d4d0e446eefa39a463 \
    --hash=sha256:a257de175c254d39ccd6a21341cd62eb7373b05c1e618a78096a56a857e0c316 \
    --hash=sha256:a79e16222106b2f5edbca1b8185661477d8971b659a3c814cc6f15181a9b34c8 \
    --hash=sha256:ae2d5a31cfb8a973a318f7c4d2cffe0bd1fe753cdf7bb41a1939d45a0a06f964 \
    --hash=sha256:ae2f67f04ed0bb2e56fd380a8bd3eef03f609df53f88b6f5c7e89c08e52aae00 \
    --hash=sha256:ae5497adc68669db2fed7cb2dad81e6a6106e79c9a132da3efdb6af1db1014fa \
    --hash=sha256:b287304f2b2220d51ccb51fd857761e78bcffbeabe7b0238f8dc36f2edfd9542 \
    --hash=sha256:b2f8877990a72ff595507b80f4b69036a9a1986a641f8681adf3425d97d3d2a5 \
    --hash=sha256:bb4cb3e37a9b961ddd68e873b29fe9ab4a081e3703412e34aedd2b7a8e9cafd9 \
    --hash=sha256:bbc2ce1f5ae5143a7fb72b71fa71db6a42b4f6cf912aa3acdc6b914084778e68 \
    --hash=sha256:bda3d98e2bcece388232cfc551ebf063b55bdb98f65ab54df397da30efc7dcc5 \
    --hash=sha256:bdc0d039e44a727824639824090bd8869535f729878fa248addd3dc01db30eae \
    --hash=sha256:bfa2e20342b81921b98edba52f8deb68843fa9c95250739a56b52ceda5ea5c61 \
    --hash=sha256:c3807ac3b16288dffebb3474b555b56fe466baa677dfc16290dcd02dca1ab228 \
    --hash=sha256:c3c9f0ca510e0de95abd6424789dcbc879942a3a4e29b0dfa99d9427bf1da75c \
    --hash=sha256:c8ed5d2c04cdc1afc6b3c28d59580448ac07732c50d94c15e14670f9c473a2ce \
    --hash=sha256:cba01c6b76e01ec453933b3b3c0157c59b52881c83eaa0f7666244e71aa75fd1 \
    --hash=sha256:ce7a28bc8af685a69d7e869d09d3e180a275e3281e29cf5f1c7319e231932cc7 \
    --hash=sha256:d10a9f7bae608ccfdc009351f01dc3d8535ff57f9488a58a4c38e45bf954fe93 \
    --hash=sha256:d3ac139377cfe71ba04573f1cda66e663b7c3e95be850e9e6c2dd4b5984bd513 \
    --hash=sha256:d5b3defa74f3723a388bfde2f5d488742bc4879682bd93267c09a3bcdf8f869b \
    --hash=sha256:d784938534e255473155e4d9f276ee69eb85455b6af1292172c731409bf9adee \
    --hash=sha256:d784a98c492c751f228a4a894c3b8a092ff08b24e73b5568938c28b8c0e8f8df \
    --hash=sha256:d8a85e3e47e0d4eebfaf9a58b40aa94f977a56050cb5598ad5396a9ee7c087c6 \
    --hash=sha256:d93321eea0dd7e81b283e47a1d20dee6069165cc158286316d0d06d340de8fe8 \
    --hash=sha256:da52ee0dc8ba03348ffceab767bd8230842fdf78f8a996e2a16445747143a778 \
    --hash=sha256:dab431699b5d45e0ca043bc580651ce9583dda594e62e245b7497adb32e99809 \
    --hash=sha256:dac4bb42f8679aadc59bd91a4c5a1784a758ad49d0912995945cd674089f628e \
    --hash=sha256:e056fb9e22d118cc546107f97dc28b449d88274207dd28872bd668c86216e4f6 \
    --hash=sha256:e09000b2599e1836314430f81a3884c66a5cbabdff5d9f175b5d560d4de38d78 \
    --hash=sha256:e0ccb5aa0f3be2727117e5631200fbb3a5b3a2b3757545a92647d6dd8be6658f \
    --hash=sha256:e57a5ab08b0ec7a7caf30d2ac79bb30c89168431aca4f8854464bb9461686925 \
    --hash=sha256:e9a7ffb1e551c6df51d267f5a751f042b183df22690f6feceac8d27364fd51d7 \
    --hash=sha256:e9c80ce0001efa16066358d2dd77993adc25f5a6c61850e4ad096a2232930bce \
    --hash=sha256:eb2c1da1cc39509d1a55620a5f4d14f8911c47a89c926a96e6f4876e864375a3 \
    --hash=sha256:edcf3121890b5f0616aa5a54683b1a5d2332037b970e507d6bb7841a3a596556 \
    --hash=sha256:f603bdd8deac6726d39f41688ed353c532dd53935234405d79e9eb53f152fbfb \
    --hash=sha256:f8934bdfd202ebd0697040a3dff40dd77bc4c5bbf3527ede0532f5e7fb4d970f \
    --hash=sha256:fdb4adb76e21bad318210310590de61c9f4adcef77ee49b4a234f9dc48867869 \
    --hash=sha256:fdb58dee173ef33634c3016c459d671ca12d11e6acf9db008261cbe58107e579
scikit-learn==1.3.2 ; python_full_version == "3.10.13" \
    --hash=sha256:0402638c9a7c219ee52c94cbebc8fcb5eb9fe9c773717965c1f4185588ad3107 \
    --hash=sha256:0ee107923a623b9f517754ea2f69ea3b62fc898a3641766cb7deb2f2ce450161 \
    --hash=sha256:1215e5e58e9880b554b01187b8c9390bf4dc4692eedeaf542d3273f4785e342c \
    --hash=sha256:15e1e94cc23d04d39da797ee34236ce2375ddea158b10bee3c343647d615581d \
    --hash=sha256:18424efee518a1cde7b0b53a422cde2f6625197de6af36da0b57ec502f126157 \
    --hash=sha256:1d08ada33e955c54355d909b9c06a4789a729977f165b8bae6f225ff0a60ec4a \
    --hash=sha256:3271552a5eb16f208a6f7f617b8cc6d1f137b52c8a1ef8edf547db0259b2c9fb \
    --hash=sha256:35a22e8015048c628ad099da9df5ab3004cdbf81edc75b396fd0cff8699ac58c \
    --hash=sha256:535805c2a01ccb40ca4ab7d081d771aea67e535153e35a1fd99418fcedd1648a \
    --hash=sha256:5b2de18d86f630d68fe1f87af690d451388bb186480afc719e5f770590c2ef6c \
    --hash=sha256:61a6efd384258789aa89415a410dcdb39a50e19d3d8410bd29be365bcdd512d5 \
    --hash=sha256:64381066f8aa63c2710e6b56edc9f0894cc7bf59bd71b8ce5613a4559b6145e0 \
    --hash=sha256:67f37d708f042a9b8d59551cf94d30431e01374e00dc2645fa186059c6c5d78b \
    --hash=sha256:6c43290337f7a4b969d207e620658372ba3c1ffb611f8bc2b6f031dc5c6d1d03 \
    --hash=sha256:6fb6bc98f234fda43163ddbe36df8bcde1d13ee176c6dc9b92bb7d3fc842eb66 \
    --hash=sha256:763f0ae4b79b0ff9cca0bf3716bcc9915bdacff3cebea15ec79652d1cc4fa5c9 \
    --hash=sha256:785a2213086b7b1abf037aeadbbd6d67159feb3e30263434139c98425e3dcfcf \
    --hash=sha256:8db94cd8a2e038b37a80a04df8783e09caac77cbe052146432e67800e430c028 \
    --hash=sha256:a19f90f95ba93c1a7f7924906d0576a84da7f3b2282ac3bfb7a08a32801add93 \
    --hash=sha256:a2f54c76accc15a34bfb9066e6c7a56c1e7235dda5762b990792330b52ccfb05 \
    --hash=sha256:b8692e395a03a60cd927125eef3a8e3424d86dde9b2370d544f0ea35f78a8073 \
    --hash=sha256:cb06f8dce3f5ddc5dee1715a9b9f19f20d295bed8e3cd4fa51e1d050347de525 \
    --hash=sha256:dc9002fc200bed597d5d34e90c752b74df516d592db162f756cc52836b38fe0e \
    --hash=sha256:e326c0eb5cf4d6ba40f93776a20e9a7a69524c4db0757e7ce24ba222471ee8a1 \
    --hash=sha256:ed932ea780517b00dae7431e031faae6b49b20eb6950918eb83bd043237950e0 \
    --hash=sha256:fc4144a5004a676d5022b798d9e573b05139e77f271253a4703eed295bde0433
scipy==1.11.4 ; python_full_version == "3.10.13" \
    --hash=sha256:00150c5eae7b610c32589dda259eacc7c4f1665aedf25d921907f4d08a951b1c \
    --hash=sha256:028eccd22e654b3ea01ee63705681ee79933652b2d8f873e7949898dda6d11b6 \
    --hash=sha256:1b7c3dca977f30a739e0409fb001056484661cb2541a01aba0bb0029f7b68db8 \
    --hash=sha256:2c6ff6ef9cc27f9b3db93a6f8b38f97387e6e0591600369a297a50a8e96e835d \
    --hash=sha256:36750b7733d960d7994888f0d148d31ea3017ac15eef664194b4ef68d36a4a97 \
    --hash=sha256:530f9ad26440e85766509dbf78edcfe13ffd0ab7fec2560ee5c36ff74d6269ff \
    --hash=sha256:5e347b14fe01003d3b78e196e84bd3f48ffe4c8a7b8a1afbcb8f5505cb710993 \
    --hash=sha256:6550466fbeec7453d7465e74d4f4b19f905642c89a7525571ee91dd7adabb5a3 \
    --hash=sha256:6df1468153a31cf55ed5ed39647279beb9cfb5d3f84369453b49e4b8502394fd \
    --hash=sha256:6e619aba2df228a9b34718efb023966da781e89dd3d21637b27f2e54db0410d7 \
    --hash=sha256:8fce70f39076a5aa62e92e69a7f62349f9574d8405c0a5de6ed3ef72de07f446 \
    --hash=sha256:90a2b78e7f5733b9de748f589f09225013685f9b218275257f8a8168ededaeaa \
    --hash=sha256:91af76a68eeae0064887a48e25c4e616fa519fa0d38602eda7e0f97d65d57937 \
    --hash=sha256:933baf588daa8dc9a92c20a0be32f56d43faf3d1a60ab11b3f08c356430f6e56 \
    --hash=sha256:acf8ed278cc03f5aff035e69cb511741e0418681d25fbbb86ca65429c4f4d9cd \
    --hash=sha256:ad669df80528aeca5f557712102538f4f37e503f0c5b9541655016dd0932ca79 \
    --hash=sha256:b030c6674b9230d37c5c60ab456e2cf12f6784596d15ce8da9365e70896effc4 \
    --hash=sha256:b9999c008ccf00e8fbcce1236f85ade5c569d13144f77a1946bef8863e8f6eb4 \
    --hash=sha256:bc9a714581f561af0848e6b69947fda0614915f072dfd14142ed1bfe1b806710 \
    --hash=sha256:ce7fff2e23ab2cc81ff452a9444c215c28e6305f396b2ba88343a567feec9660 \
    --hash=sha256:cf00bd2b1b0211888d4dc75656c0412213a8b25e80d73898083f402b50f47e41 \
    --hash=sha256:d10e45a6c50211fe256da61a11c34927c68f277e03138777bdebedd933712fea \
    --hash=sha256:ee410e6de8f88fd5cf6eadd73c135020bfbbbdfcd0f6162c36a7638a1ea8cc65 \
    --hash=sha256:f313b39a7e94f296025e3cffc2c567618174c0b1dde173960cf23808f9fae4be \
    --hash=sha256:f3cd9e7b3c2c1ec26364856f9fbe78695fe631150f94cd1c22228456404cf1ec
setuptools==69.0.2 ; python_full_version == "3.10.13" \
    --hash=sha256:1e8fdff6797d3865f37397be788a4e3cba233608e9b509382a2777d25ebde7f2 \
    --hash=sha256:735896e78a4742605974de002ac60562d286fa8051a7e2299445e8e8fbb01aa6
shellingham==1.5.4 ; python_full_version == "3.10.13" \
    --hash=sha256:7ecfff8f2fd72616f7481040475a65b2bf8af90a56c89140852d1120324e8686 \
    --hash=sha256:8dbca0739d487e5bd35ab3ca4b36e11c4078f3a234bfce294b0a0291363404de
six==1.16.0 ; python_full_version == "3.10.13" \
    --hash=sha256:1e61c37477a1626458e36f7b1d82aa5c9b094fa4802892072e49de9c60c4c926 \
    --hash=sha256:8abb2f1d86890a2dfb989f9a77cfcfd3e47c2a354b01111771326f8aa26e0254
sniffio==1.3.0 ; python_full_version == "3.10.13" \
    --hash=sha256:e60305c5e5d314f5389259b7f22aaa33d8f7dee49763119234af3755c55b9101 \
    --hash=sha256:eecefdce1e5bbfb7ad2eeaabf7c1eeb404d7757c379bd1f7e5cce9d8bf425384
speaklater==1.3 ; python_full_version == "3.10.13" \
    --hash=sha256:59fea336d0eed38c1f0bf3181ee1222d0ef45f3a9dd34ebe65e6bfffdd6a65a9
sqlalchemy==2.0.23 ; python_full_version == "3.10.13" \
    --hash=sha256:0666031df46b9badba9bed00092a1ffa3aa063a5e68fa244acd9f08070e936d3 \
    --hash=sha256:0a8c6aa506893e25a04233bc721c6b6cf844bafd7250535abb56cb6cc1368884 \
    --hash=sha256:0e680527245895aba86afbd5bef6c316831c02aa988d1aad83c47ffe92655e74 \
    --hash=sha256:14aebfe28b99f24f8a4c1346c48bc3d63705b1f919a24c27471136d2f219f02d \
    --hash=sha256:1e018aba8363adb0599e745af245306cb8c46b9ad0a6fc0a86745b6ff7d940fc \
    --hash=sha256:227135ef1e48165f37590b8bfc44ed7ff4c074bf04dc8d6f8e7f1c14a94aa6ca \
    --hash=sha256:31952bbc527d633b9479f5f81e8b9dfada00b91d6baba021a869095f1a97006d \
    --hash=sha256:3e983fa42164577d073778d06d2cc5d020322425a509a08119bdcee70ad856bf \
    --hash=sha256:42d0b0290a8fb0165ea2c2781ae66e95cca6e27a2fbe1016ff8db3112ac1e846 \
    --hash=sha256:42ede90148b73fe4ab4a089f3126b2cfae8cfefc955c8174d697bb46210c8306 \
    --hash=sha256:4895a63e2c271ffc7a81ea424b94060f7b3b03b4ea0cd58ab5bb676ed02f4221 \
    --hash=sha256:4af79c06825e2836de21439cb2a6ce22b2ca129bad74f359bddd173f39582bf5 \
    --hash=sha256:5f94aeb99f43729960638e7468d4688f6efccb837a858b34574e01143cf11f89 \
    --hash=sha256:616fe7bcff0a05098f64b4478b78ec2dfa03225c23734d83d6c169eb41a93e55 \
    --hash=sha256:62d9e964870ea5ade4bc870ac4004c456efe75fb50404c03c5fd61f8bc669a72 \
    --hash=sha256:638c2c0b6b4661a4fd264f6fb804eccd392745c5887f9317feb64bb7cb03b3ea \
    --hash=sha256:63bfc3acc970776036f6d1d0e65faa7473be9f3135d37a463c5eba5efcdb24c8 \
    --hash=sha256:6463aa765cf02b9247e38b35853923edbf2f6fd1963df88706bc1d02410a5577 \
    --hash=sha256:64ac935a90bc479fee77f9463f298943b0e60005fe5de2aa654d9cdef46c54df \
    --hash=sha256:683ef58ca8eea4747737a1c35c11372ffeb84578d3aab8f3e10b1d13d66f2bc4 \
    --hash=sha256:75eefe09e98043cff2fb8af9796e20747ae870c903dc61d41b0c2e55128f958d \
    --hash=sha256:787af80107fb691934a01889ca8f82a44adedbf5ef3d6ad7d0f0b9ac557e0c34 \
    --hash=sha256:7c424983ab447dab126c39d3ce3be5bee95700783204a72549c3dceffe0fc8f4 \
    --hash=sha256:7e0dc9031baa46ad0dd5a269cb7a92a73284d1309228be1d5935dac8fb3cae24 \
    --hash=sha256:87a3d6b53c39cd173990de2f5f4b83431d534a74f0e2f88bd16eabb5667e65c6 \
    --hash=sha256:89a01238fcb9a8af118eaad3ffcc5dedaacbd429dc6fdc43fe430d3a941ff965 \
    --hash=sha256:9585b646ffb048c0250acc7dad92536591ffe35dba624bb8fd9b471e25212a35 \
    --hash=sha256:964971b52daab357d2c0875825e36584d58f536e920f2968df8d581054eada4b \
    --hash=sha256:967c0b71156f793e6662dd839da54f884631755275ed71f1539c95bbada9aaab \
    --hash=sha256:9ca922f305d67605668e93991aaf2c12239c78207bca3b891cd51a4515c72e22 \
    --hash=sha256:a86cb7063e2c9fb8e774f77fbf8475516d270a3e989da55fa05d08089d77f8c4 \
    --hash=sha256:aeb397de65a0a62f14c257f36a726945a7f7bb60253462e8602d9b97b5cbe204 \
    --hash=sha256:b41f5d65b54cdf4934ecede2f41b9c60c9f785620416e8e6c48349ab18643855 \
    --hash=sha256:bd45a5b6c68357578263d74daab6ff9439517f87da63442d244f9f23df56138d \
    --hash=sha256:c14eba45983d2f48f7546bb32b47937ee2cafae353646295f0e99f35b14286ab \
    --hash=sha256:c1bda93cbbe4aa2aa0aa8655c5aeda505cd219ff3e8da91d1d329e143e4aff69 \
    --hash=sha256:c4722f3bc3c1c2fcc3702dbe0016ba31148dd6efcd2a2fd33c1b4897c6a19693 \
    --hash=sha256:c80c38bd2ea35b97cbf7c21aeb129dcbebbf344ee01a7141016ab7b851464f8e \
    --hash=sha256:cabafc7837b6cec61c0e1e5c6d14ef250b675fa9c3060ed8a7e38653bd732ff8 \
    --hash=sha256:cc1d21576f958c42d9aec68eba5c1a7d715e5fc07825a629015fe8e3b0657fb0 \
    --hash=sha256:d0f7fb0c7527c41fa6fcae2be537ac137f636a41b4c5a4c58914541e2f436b45 \
    --hash=sha256:d4041ad05b35f1f4da481f6b811b4af2f29e83af253bf37c3c4582b2c68934ab \
    --hash=sha256:d5578e6863eeb998980c212a39106ea139bdc0b3f73291b96e27c929c90cd8e1 \
    --hash=sha256:e3b5036aa326dc2df50cba3c958e29b291a80f604b1afa4c8ce73e78e1c9f01d \
    --hash=sha256:e599a51acf3cc4d31d1a0cf248d8f8d863b6386d2b6782c5074427ebb7803bda \
    --hash=sha256:f3420d00d2cb42432c1d0e44540ae83185ccbbc67a6054dcc8ab5387add6620b \
    --hash=sha256:f48ed89dd11c3c586f45e9eec1e437b355b3b6f6884ea4a4c3111a3358fd0c18 \
    --hash=sha256:f508ba8f89e0a5ecdfd3761f82dda2a3d7b678a626967608f4273e0dba8f07ac \
    --hash=sha256:fd54601ef9cc455a0c61e5245f690c8a3ad67ddb03d3b91c361d076def0b4c60
starlette==0.27.0 ; python_full_version == "3.10.13" \
    --hash=sha256:6a6b0d042acb8d469a01eba54e9cda6cbd24ac602c4cd016723117d6a7e73b75 \
    --hash=sha256:918416370e846586541235ccd38a474c08b80443ed31c578a418e2209b3eef91
tensorboard-data-server==0.7.2 ; python_full_version == "3.10.13" \
    --hash=sha256:7e0610d205889588983836ec05dc098e80f97b7e7bbff7e994ebb78f578d0ddb \
    --hash=sha256:9fe5d24221b29625dbc7328b0436ca7fc1c23de4acf4d272f1180856e32f9f60 \
    --hash=sha256:ef687163c24185ae9754ed5650eb5bc4d84ff257aabdc33f0cc6f74d8ba54530
tensorboard==2.15.1 ; python_full_version == "3.10.13" \
    --hash=sha256:c46c1d1cf13a458c429868a78b2531d8ff5f682058d69ec0840b0bc7a38f1c0f
tensorflow-estimator==2.15.0 ; python_full_version == "3.10.13" \
    --hash=sha256:aedf21eec7fb2dc91150fc91a1ce12bc44dbb72278a08b58e79ff87c9e28f153
tensorflow-io-gcs-filesystem==0.34.0 ; python_full_version == "3.10.13" \
    --hash=sha256:027a07553367187f918a99661f63ae0506b91b77a70bee9c7ccaf3920bf7cfe7 \
    --hash=sha256:0dafed144673e1173528768fe208a7c5a6e8edae40208381cac420ee7c918ec9 \
    --hash=sha256:182b0fbde7e9a537fda0b354c28b0b6c035736728de8fe2db7ef49cf90352014 \
    --hash=sha256:2b035f4c92639657b6d376929d550ac3dee9e6c0523eb434eefe0a27bae3d05b \
    --hash=sha256:396bfff61b49f80b86ddebe0c76ae0f2731689cee49ad7d782625180b50b13af \
    --hash=sha256:3f346b287ed2400e09b13cfd8524222fd70a66aadb9164c645286c2087007e9f \
    --hash=sha256:44ad387a812a78e7424bb8bee3820521ae1c044bddf72b1e163e8df95c124a74 \
    --hash=sha256:5813c336b4f7cb0a01ff4cc6cbd3edf11ef67305baf0e3cf634911b702f493f8 \
    --hash=sha256:6e6353123a5b51397950138a118876af833a7db66b531123bb86f82e80ab0e72 \
    --hash=sha256:7f60183473f0ca966451bb1d1bb5dc29b3cf9c74d1d0e7f2ed46760ed56bd4af \
    --hash=sha256:8d8664bddbe4e7b56ce94db8b93ea9077a158fb5e15364e11e29f93015ceea24 \
    --hash=sha256:a17a616d2c7fae83de4424404815843507d40d4eb0d507c636a5493a20c3d958 \
    --hash=sha256:b20622f8572fcb6c93e8f7d626327472f263e47ebd63d2153ef09162ef5ef7b5 \
    --hash=sha256:b9a93fcb01db269bc845a1ced431f3c61201755ce5f9ec4885760f30122276ef \
    --hash=sha256:cbe26c4a3332589c7b724f147df453b5c226993aa8d346a15536358d77b364c4 \
    --hash=sha256:d3feba2dd76f7c188137c34642d68d378f0eed81636cb95090ecb1496722707c \
    --hash=sha256:d831702fbb270996b27cda7fde06e0825b2ea81fd8dd3ead35242f4f8b3889b8 \
    --hash=sha256:ec4604c99cbb5b708f4516dee27aa655abae222b876c98b740f4c2f89dd5c001 \
    --hash=sha256:f211d2b3db8f9931765992b607b71cbfb98c8cd6169079d004a67a94ab10ecb4
tensorflow==2.15.0 ; python_full_version == "3.10.13" \
    --hash=sha256:01108746e1bbfcd48dfabf7f51ddca7693b91ea6821f6f62a27b5a5ebf0817c5 \
    --hash=sha256:124930e7d4f5d74c61a5c80d642a26c22fe0c42fdd383fe9ee5803c3ac9ed4ce \
    --hash=sha256:1e0716622ed7af867d8b1997b00a2940f1a1587dee923ff53efa2ee506992f32 \
    --hash=sha256:2cfcdde1ff3c01be617e99ce9783c49cb11da5796ce32a31855412bd092c0bcf \
    --hash=sha256:2d88f8b71f4a8d9ab9dc7c8e42b14ca0f53d1daab0f989b8f2918907c2891f41 \
    --hash=sha256:3fa865956d96b7614f247c36e4c22b1543ba5ce656fbe8e4f6266ae7a4917132 \
    --hash=sha256:852efeb4d18beedac0120c4f2d4f4dccf4c090bb6740c5199d395ff609e85e98 \
    --hash=sha256:896bda03f722700a9918d144aee5152a75f1be5e6c5045fd0683b8318a3fc9d9 \
    --hash=sha256:9b248e0f4316b3a3c54cd1f83edfb7a761d473060c1972a8ea31a90d5de3aa72 \
    --hash=sha256:dee8ec2b2c6c942ae65d25746e53cdc475e82d5fcbbb3009ce47f5963d69ebfc \
    --hash=sha256:e05a48006930e4e9e68468e7affed3bbce8a1c7fe6df86500496ad1558804a78 \
    --hash=sha256:e7697b005ce48fec8b2ee8cf25bcbd138f16b5e17f99f7c01a6ea3f2429f86c6 \
    --hash=sha256:e98aab454fc73ff1900314821e5bafbf20840ada2004c8caccf4d92e0e12a628 \
    --hash=sha256:eaf420d8b8ec1d4bd75859be7d7545d8e7052726eed8456fdbba63718e7e07ea \
    --hash=sha256:ed601b43df9b7d9bed0203b34bcb9356efd4f671eaaac1046b7166a2afee0cf8
termcolor==2.3.0 ; python_full_version == "3.10.13" \
    --hash=sha256:3afb05607b89aed0ffe25202399ee0867ad4d3cb4180d98aaf8eefa6a5f7d475 \
    --hash=sha256:b5b08f68937f138fe92f6c089b99f1e2da0ae56c52b78bf7075fd95420fd9a5a
threadpoolctl==3.2.0 ; python_full_version == "3.10.13" \
    --hash=sha256:2b7818516e423bdaebb97c723f86a7c6b0a83d3f3b0970328d66f4d9104dc032 \
    --hash=sha256:c96a0ba3bdddeaca37dc4cc7344aafad41cdb8c313f74fdfe387a867bba93355
tokenizers==0.15.0 ; python_full_version == "3.10.13" \
    --hash=sha256:01a3aa332abc4bee7640563949fcfedca4de8f52691b3b70f2fc6ca71bfc0f4e \
    --hash=sha256:0344d6602740e44054a9e5bbe9775a5e149c4dddaff15959bb07dcce95a5a859 \
    --hash=sha256:05accb9162bf711a941b1460b743d62fec61c160daf25e53c5eea52c74d77814 \
    --hash=sha256:05b83896a893cdfedad8785250daa3ba9f0504848323471524d4783d7291661e \
    --hash=sha256:0a1a3c973e4dc97797fc19e9f11546c95278ffc55c4492acb742f69e035490bc \
    --hash=sha256:0ea480d943297df26f06f508dab6e012b07f42bf3dffdd36e70799368a5f5229 \
    --hash=sha256:10361e9c7864b22dd791ec5126327f6c9292fb1d23481d4895780688d5e298ac \
    --hash=sha256:10c7e6e7b4cabd757da59e93f5f8d1126291d16f8b54f28510825ef56a3e5d0e \
    --hash=sha256:1574a5a4af22c3def93fe8fe4adcc90a39bf5797ed01686a4c46d1c3bc677d2f \
    --hash=sha256:160f9d1810f2c18fffa94aa98bf17632f6bd2dabc67fcb01a698ca80c37d52ee \
    --hash=sha256:1ab96ab7dc706e002c32b2ea211a94c1c04b4f4de48354728c3a6e22401af322 \
    --hash=sha256:1eef39a502fad3bf104b9e1906b4fb0cee20e44e755e51df9a98f8922c3bf6d4 \
    --hash=sha256:22c27672c27a059a5f39ff4e49feed8c7f2e1525577c8a7e3978bd428eb5869d \
    --hash=sha256:26a2ef890740127cb115ee5260878f4a677e36a12831795fd7e85887c53b430b \
    --hash=sha256:2a0dd641a72604486cd7302dd8f87a12c8a9b45e1755e47d2682733f097c1af5 \
    --hash=sha256:2a5f4543a35889679fc3052086e69e81880b2a5a28ff2a52c5a604be94b77a3f \
    --hash=sha256:2dd681b53cf615e60a31a115a3fda3980e543d25ca183797f797a6c3600788a3 \
    --hash=sha256:309445d10d442b7521b98083dc9f0b5df14eca69dbbfebeb98d781ee2cef5d30 \
    --hash=sha256:309cfcccfc7e502cb1f1de2c9c1c94680082a65bfd3a912d5a5b2c90c677eb60 \
    --hash=sha256:32371008788aeeb0309a9244809a23e4c0259625e6b74a103700f6421373f395 \
    --hash=sha256:331dd786d02fc38698f835fff61c99480f98b73ce75a4c65bd110c9af5e4609a \
    --hash=sha256:3661862df7382c5eb23ac4fbf7c75e69b02dc4f5784e4c5a734db406b5b24596 \
    --hash=sha256:3768829861e964c7a4556f5f23307fce6a23872c2ebf030eb9822dbbbf7e9b2a \
    --hash=sha256:3b22cd714706cc5b18992a232b023f736e539495f5cc61d2d28d176e55046f6c \
    --hash=sha256:3bb0f4df6dce41a1c7482087b60d18c372ef4463cb99aa8195100fcd41e0fd64 \
    --hash=sha256:3c2b60b12fdd310bf85ce5d7d3f823456b9b65eed30f5438dd7761879c495983 \
    --hash=sha256:4525f6997d81d9b6d9140088f4f5131f6627e4c960c2c87d0695ae7304233fc3 \
    --hash=sha256:4a0a94bc3370e6f1cc8a07a8ae867ce13b7c1b4291432a773931a61f256d44ea \
    --hash=sha256:4a522612d5c88a41563e3463226af64e2fa00629f65cdcc501d1995dd25d23f5 \
    --hash=sha256:4b31807cb393d6ea31926b307911c89a1209d5e27629aa79553d1599c8ffdefe \
    --hash=sha256:5d37e7f4439b4c46192ab4f2ff38ab815e4420f153caa13dec9272ef14403d34 \
    --hash=sha256:65975094fef8cc68919644936764efd2ce98cf1bacbe8db2687155d2b0625bee \
    --hash=sha256:65f80be77f6327a86d8fd35a4467adcfe6174c159b4ab52a1a8dd4c6f2d7d9e1 \
    --hash=sha256:669b8ed653a578bcff919566631156f5da3aab84c66f3c0b11a6281e8b4731c7 \
    --hash=sha256:6fdcc55339df7761cd52e1fbe8185d3b3963bc9e3f3545faa6c84f9e8818259a \
    --hash=sha256:6fe143939f3b596681922b2df12a591a5b010e7dcfbee2202482cd0c1c2f2459 \
    --hash=sha256:7286f3df10de840867372e3e64b99ef58c677210e3ceb653cd0e740a5c53fe78 \
    --hash=sha256:72f78b0e0e276b1fc14a672fa73f3acca034ba8db4e782124a2996734a9ba9cf \
    --hash=sha256:76f1bed992e396bf6f83e3df97b64ff47885e45e8365f8983afed8556a0bc51f \
    --hash=sha256:77606994e793ca54ecf3a3619adc8a906a28ca223d9354b38df41cb8766a0ed6 \
    --hash=sha256:78104f5d035c9991f92831fc0efe9e64a05d4032194f2a69f67aaa05a4d75bbb \
    --hash=sha256:7c7982fd0ec9e9122d03b209dac48cebfea3de0479335100ef379a9a959b9a5a \
    --hash=sha256:7f17363141eb0c53752c89e10650b85ef059a52765d0802ba9613dbd2d21d425 \
    --hash=sha256:82641ffb13a4da1293fcc9f437d457647e60ed0385a9216cd135953778b3f0a1 \
    --hash=sha256:8413e994dd7d875ab13009127fc85633916c71213917daf64962bafd488f15dc \
    --hash=sha256:85ddae17570ec7e5bfaf51ffa78d044f444a8693e1316e1087ee6150596897ee \
    --hash=sha256:88dd0961c437d413ab027f8b115350c121d49902cfbadf08bb8f634b15fa1814 \
    --hash=sha256:8a765db05581c7d7e1280170f2888cda351760d196cc059c37ea96f121125799 \
    --hash=sha256:8a922c492c721744ee175f15b91704be2d305569d25f0547c77cd6c9f210f9dc \
    --hash=sha256:8d7d6eea831ed435fdeeb9bcd26476226401d7309d115a710c65da4088841948 \
    --hash=sha256:8edcc90a36eab0705fe9121d6c77c6e42eeef25c7399864fd57dfb27173060bf \
    --hash=sha256:9680b0ecc26e7e42f16680c1aa62e924d58d1c2dd992707081cc10a374896ea2 \
    --hash=sha256:9855e6c258918f9cf62792d4f6ddfa6c56dccd8c8118640f867f6393ecaf8bd7 \
    --hash=sha256:9a3241acdc9b44cff6e95c4a55b9be943ef3658f8edb3686034d353734adba05 \
    --hash=sha256:9c91588a630adc88065e1c03ac6831e3e2112558869b9ebcb2b8afd8a14c944d \
    --hash=sha256:a40b73dc19d82c3e3ffb40abdaacca8fbc95eeb26c66b7f9f860aebc07a73998 \
    --hash=sha256:a79f17027f24fe9485701c8dbb269b9c713954ec3bdc1e7075a66086c0c0cd3c \
    --hash=sha256:a8da7533dbe66b88afd430c56a2f2ce1fd82e2681868f857da38eeb3191d7498 \
    --hash=sha256:a9fcaad9ab0801f14457d7c820d9f246b5ab590c407fc6b073819b1573097aa7 \
    --hash=sha256:aab16c4a26d351d63e965b0c792f5da7227a37b69a6dc6d922ff70aa595b1b0c \
    --hash=sha256:aabc83028baa5a36ce7a94e7659250f0309c47fa4a639e5c2c38e6d5ea0de564 \
    --hash=sha256:ab806ad521a5e9de38078b7add97589c313915f6f5fec6b2f9f289d14d607bd6 \
    --hash=sha256:ae17884aafb3e94f34fb7cfedc29054f5f54e142475ebf8a265a4e388fee3f8b \
    --hash=sha256:af7e9be8c05d30bb137b9fd20f9d99354816599e5fd3d58a4b1e28ba3b36171f \
    --hash=sha256:b3cdf29e6f9653da330515dc8fa414be5a93aae79e57f8acc50d4028dd843edf \
    --hash=sha256:b7bee0f1795e3e3561e9a557061b1539e5255b8221e3f928f58100282407e090 \
    --hash=sha256:b8034f1041fd2bd2b84ff9f4dc4ae2e1c3b71606820a9cd5c562ebd291a396d1 \
    --hash=sha256:babe42635b8a604c594bdc56d205755f73414fce17ba8479d142a963a6c25cbc \
    --hash=sha256:bc80a0a565ebfc7cd89de7dd581da8c2b3238addfca6280572d27d763f135f2f \
    --hash=sha256:c1e4664c5b797e093c19b794bbecc19d2367e782b4a577d8b7c1821db5dc150d \
    --hash=sha256:c3045d191dad49647f5a5039738ecf1c77087945c7a295f7bcf051c37067e883 \
    --hash=sha256:c3d7343fa562ea29661783344a2d83662db0d3d17a6fa6a403cac8e512d2d9fd \
    --hash=sha256:c9cce6ee149a3d703f86877bc2a6d997e34874b2d5a2d7839e36b2273f31d3d9 \
    --hash=sha256:ca003fb5f3995ff5cf676db6681b8ea5d54d3b30bea36af1120e78ee1a4a4cdf \
    --hash=sha256:ca9db64c7c9954fbae698884c5bb089764edc549731e5f9b7fa1dd4e4d78d77f \
    --hash=sha256:caadf255cf7f951b38d10097836d1f3bcff4aeaaffadfdf748bab780bf5bff95 \
    --hash=sha256:cbbf2489fcf25d809731ba2744ff278dd07d9eb3f8b7482726bd6cae607073a4 \
    --hash=sha256:cd3cd0299aaa312cd2988957598f80becd04d5a07338741eca076057a2b37d6e \
    --hash=sha256:cdd945e678bbdf4517d5d8de66578a5030aeefecdb46f5320b034de9cad8d4dd \
    --hash=sha256:d0ebf9430f901dbdc3dcb06b493ff24a3644c9f88c08e6a1d6d0ae2228b9b818 \
    --hash=sha256:d3125a6499226d4d48efc54f7498886b94c418e93a205b673bc59364eecf0804 \
    --hash=sha256:d4fab75642aae4e604e729d6f78e0addb9d7e7d49e28c8f4d16b24da278e5263 \
    --hash=sha256:d801d1368188c74552cd779b1286e67cb9fd96f4c57a9f9a2a09b6def9e1ab37 \
    --hash=sha256:dbed5944c31195514669cf6381a0d8d47f164943000d10f93d6d02f0d45c25e0 \
    --hash=sha256:de9529fe75efcd54ba8d516aa725e1851df9199f0669b665c55e90df08f5af86 \
    --hash=sha256:e54c5f26df14913620046b33e822cb3bcd091a332a55230c0e63cc77135e2169 \
    --hash=sha256:e58a38c4e6075810bdfb861d9c005236a72a152ebc7005941cc90d1bbf16aca9 \
    --hash=sha256:ed56ddf0d54877bb9c6d885177db79b41576e61b5ef6defeb579dcb803c04ad5 \
    --hash=sha256:edde9aa964145d528d0e0dbf14f244b8a85ebf276fb76869bc02e2530fa37a96 \
    --hash=sha256:f1480b0051d8ab5408e8e4db2dc832f7082ea24aa0722c427bde2418c6f3bd07 \
    --hash=sha256:f17cbd88dab695911cbdd385a5a7e3709cc61dff982351f5d1b5939f074a2466 \
    --hash=sha256:f21c9eb71c9a671e2a42f18b456a3d118e50c7f0fc4dd9fa8f4eb727fea529bf \
    --hash=sha256:f6456bec6c557d63d8ec0023758c32f589e1889ed03c055702e84ce275488bed \
    --hash=sha256:f8aa81afec893e952bd39692b2d9ef60575ed8c86fce1fd876a06d2e73e82dca \
    --hash=sha256:f8d16b647032df2ce2c1f9097236e046ea9fedd969b25637b9d5d734d78aa53b \
    --hash=sha256:fa8eb4584fc6cbe6a84d7a7864be3ed28e23e9fd2146aa8ef1814d579df91958 \
    --hash=sha256:fac2719b1e9bc8e8e7f6599b99d0a8e24f33d023eb8ef644c0366a596f0aa926 \
    --hash=sha256:ff5d2159c5d93015f5a4542aac6c315506df31853123aa39042672031768c301
tomli==2.0.1 ; python_full_version == "3.10.13" \
    --hash=sha256:939de3e7a6161af0c887ef91b7d41a53e7c5a1ca976325f429cb46ea9bc30ecc \
    --hash=sha256:de526c12914f0c550d15924c62d72abc48d6fe7364aa87328337a31007fe8a4f
torch==1.13.1 ; python_full_version == "3.10.13" \
    --hash=sha256:0122806b111b949d21fa1a5f9764d1fd2fcc4a47cb7f8ff914204fd4fc752ed5 \
    --hash=sha256:0aa46f0ac95050c604bcf9ef71da9f1172e5037fdf2ebe051962d47b123848e7 \
    --hash=sha256:0d9b8061048cfb78e675b9d2ea8503bfe30db43d583599ae8626b1263a0c1380 \
    --hash=sha256:22128502fd8f5b25ac1cd849ecb64a418382ae81dd4ce2b5cebaa09ab15b0d9b \
    --hash=sha256:2c3581a3fd81eb1f0f22997cddffea569fea53bafa372b2c0471db373b26aafc \
    --hash=sha256:2ee7b81e9c457252bddd7d3da66fb1f619a5d12c24d7074de91c4ddafb832c93 \
    --hash=sha256:33e67eea526e0bbb9151263e65417a9ef2d8fa53cbe628e87310060c9dcfa312 \
    --hash=sha256:393a6273c832e047581063fb74335ff50b4c566217019cc6ace318cd79eb0566 \
    --hash=sha256:50ff5e76d70074f6653d191fe4f6a42fdbe0cf942fbe2a3af0b75eaa414ac038 \
    --hash=sha256:5e1e722a41f52a3f26f0c4fcec227e02c6c42f7c094f32e49d4beef7d1e213ea \
    --hash=sha256:6930791efa8757cb6974af73d4996b6b50c592882a324b8fb0589c6a9ba2ddaf \
    --hash=sha256:727dbf00e2cf858052364c0e2a496684b9cb5aa01dc8a8bc8bbb7c54502bdcdd \
    --hash=sha256:76024be052b659ac1304ab8475ab03ea0a12124c3e7626282c9c86798ac7bc11 \
    --hash=sha256:98124598cdff4c287dbf50f53fb455f0c1e3a88022b39648102957f3445e9b76 \
    --hash=sha256:d9fe785d375f2e26a5d5eba5de91f89e6a3be5d11efb497e76705fdf93fa3c2e \
    --hash=sha256:df8434b0695e9ceb8cc70650afc1310d8ba949e6db2a0525ddd9c3b2b181e5fe \
    --hash=sha256:e0df902a7c7dd6c795698532ee5970ce898672625635d885eade9976e5a04949 \
    --hash=sha256:ea8dda84d796094eb8709df0fcd6b56dc20b58fdd6bc4e8d7109930dafc8e419 \
    --hash=sha256:eeeb204d30fd40af6a2d80879b46a7efbe3cf43cdbeb8838dd4f3d126cc90b2b \
    --hash=sha256:f402ca80b66e9fbd661ed4287d7553f7f3899d9ab54bf5c67faada1555abde28 \
    --hash=sha256:fd12043868a34a8da7d490bf6db66991108b00ffbeecb034228bfcbbd4197143
tqdm==4.66.1 ; python_full_version == "3.10.13" \
    --hash=sha256:d302b3c5b53d47bce91fea46679d9c3c6508cf6332229aa1e7d8653723793386 \
    --hash=sha256:d88e651f9db8d8551a62556d3cff9e3034274ca5d66e93197cf2490e2dcb69c7
transformers==4.35.2 ; python_full_version == "3.10.13" \
    --hash=sha256:2d125e197d77b0cdb6c9201df9fa7e2101493272e448b9fba9341c695bee2f52 \
    --hash=sha256:9dfa76f8692379544ead84d98f537be01cd1070de75c74efb13abcbc938fbe2f
typer[all]==0.9.0 ; python_full_version == "3.10.13" \
    --hash=sha256:50922fd79aea2f4751a8e0408ff10d2662bd0c8bbfa84755a699f3bada2978b2 \
    --hash=sha256:5d96d986a21493606a358cae4461bd8cdf83cbf33a5aa950ae629ca3b51467ee
typing-extensions==4.8.0 ; python_full_version == "3.10.13" \
    --hash=sha256:8f92fc8806f9a6b641eaa5318da32b44d401efaac0f6678c9bc448ba3605faa0 \
    --hash=sha256:df8e4339e9cb77357558cbdbceca33c303714cf861d1eef15e1070055ae8b7ef
tzdata==2023.3 ; python_full_version == "3.10.13" \
    --hash=sha256:11ef1e08e54acb0d4f95bdb1be05da659673de4acbd21bf9c69e94cc5e907a3a \
    --hash=sha256:7e65763eef3120314099b6939b5546db7adce1e7d6f2e179e3df563c70511eda
urllib3==2.1.0 ; python_full_version == "3.10.13" \
    --hash=sha256:55901e917a5896a349ff771be919f8bd99aff50b79fe58fec595eb37bbc56bb3 \
    --hash=sha256:df7aa8afb0148fa78488e7899b2c59b5f4ffcfa82e6c54ccb9dd37c1d7b52d54
uvicorn==0.24.0.post1 ; python_full_version == "3.10.13" \
    --hash=sha256:09c8e5a79dc466bdf28dead50093957db184de356fcdc48697bad3bde4c2588e \
    --hash=sha256:7c84fea70c619d4a710153482c0d230929af7bcf76c7bfa6de151f0a3a80121e
vine==5.1.0 ; python_full_version == "3.10.13" \
    --hash=sha256:40fdf3c48b2cfe1c38a49e9ae2da6fda88e4794c810050a728bd7413811fb1dc \
    --hash=sha256:8b62e981d35c41049211cf62a0a1242d8c1ee9bd15bb196ce38aefd6799e61e0
wcwidth==0.2.12 ; python_full_version == "3.10.13" \
    --hash=sha256:f01c104efdf57971bcb756f054dd58ddec5204dd15fa31d6503ea57947d97c02 \
    --hash=sha256:f26ec43d96c8cbfed76a5075dac87680124fa84e0855195a6184da9c187f133c
```

```
werkzeug==3.0.1 ; python_full_version == "3.10.13" \
    --hash=sha256:507e811ecea72b18a404947aded4b3390e1db8f826b494d76550ef45bb3b1dcc \
    --hash=sha256:90a285dc0e42ad56b34e696398b8122ee4c681833fb35b8334a095d82c56da10
wheel==0.42.0 ; python_full_version == "3.10.13" \
    --hash=sha256:177f9c9b0d45c47873b619f5b650346d632cdc35fb5e4d25058e09c9e581433d \
    --hash=sha256:c45be39f7882c9d34243236f2d63cbd58039e360f85d0913425fbd7ceea617a8
wrapt==1.14.1 ; python_full_version == "3.10.13" \
    --hash=sha256:00b6d4ea20a906c0ca56d84f93065b398ab74b927a7a3dbd470f6fc503f95dc3 \
    --hash=sha256:01c205616a89d09827986bc4e859bcabd64f5a0662a7fe95e0d359424e0e071b \
    --hash=sha256:02b41b633c6261feff8ddd8d11c711df6842aba629fdd3da10249a53211a72c4 \
    --hash=sha256:07f7a7d0f388028b2df1d916e94bbb40624c59b48ecc6cbc232546706fac74c2 \
    --hash=sha256:11871514607b15cfeb87c547a49bca19fde402f32e2b1c24a632506c0a756656 \
    --hash=sha256:1b376b3f4896e7930f1f772ac4b064ac12598d1c38d04907e696cc4d794b43d3 \
    --hash=sha256:2020f391008ef874c6d9e208b24f28e31bcb85ccff4f335f15a3251d222b92d9 \
    --hash=sha256:21ac0156c4b089b330b7666db40feee30a5d52634cc4560e1905d6529a3897ff \
    --hash=sha256:240b1686f38ae665d1b15475966fe0472f78e71b1b4903c143a842659c8e4cb9 \
    --hash=sha256:257fd78c513e0fb5cdbe058c27a0624c9884e735bbd131935fd49e9fe719d310 \
    --hash=sha256:26046cd03936ae745a502abf44dac702a5e6880b2b01c29aea8ddf3353b68224 \
    --hash=sha256:2b39d38039a1fdad98c87279b48bc5dce2c0ca0d73483b12cb72aa9609278e8a \
    --hash=sha256:2cf71233a0ed05ccdabe209c606fe0bac7379fdcf687f39b944420d2a09fdb57 \
    --hash=sha256:2fe803deacd09a233e4762a1adcea5db5d31e6be577a43352936179d14d90069 \
    --hash=sha256:2feecf86e1f7a86517cab34ae6c2f081fd2d0dac860cb0c0ded96d799d20b335 \
    --hash=sha256:3232822c7d98d23895ccc443bbdf57c7412c5a65996c30442ebe6ed3df335383 \
    --hash=sha256:34aa51c45f28ba7f12accd624225e2b1e5a3a45206aa191f6f9aac931d9d56fe \
    --hash=sha256:358fe87cc899c6bb0ddc185bf3dbfa4ba646f05b1b0b9b5a27c2cb92c2cea204 \
    --hash=sha256:36f582d0c6bc99d5f39cd3ac2a9062e57f3cf606ade29a0a0d6b323462f4dd87 \
    --hash=sha256:380a85cf89e0e69b7cfbe2ea9f765f004ff419f34194018a6827ac0e3edfed4d \
    --hash=sha256:40e7bc81c9e2b2734ea4bc1aceb8a8f0ceaac7c5299bc5d69e37c44d9081d43b \
    --hash=sha256:43ca3bbbe97af00f49efb06e352eae40434ca9d915906f77def219b88e85d907 \
    --hash=sha256:49ef582b7a1152ae2766557f0550a9fcbf7bbd76f43fbdc94dd3bf07cc7168be \
    --hash=sha256:4fcc4649dc762cddacd193e6b55bc02edca674067f5f98166d7713b193932b7f \
    --hash=sha256:5a0f54ce2c092aaf439813735584b9537cad479575a09892b8352fea5e988dc0 \
    --hash=sha256:5a9a0d155deafd9448baff28c08e150d9b24ff010e899311ddd63c45c2445e28 \
    --hash=sha256:5b02d65b9ccf0ef6c34cba6cf5bf2aab1bb2f49c6090bafeecc9cd81ad4ea1c1 \
    --hash=sha256:60db23fa423575eeb65ea430cee741acb7c26a1365d103f7b0f6ec412b893853 \
    --hash=sha256:642c2e7a804fcf18c222e1060df25fc210b9c58db7c91416fb055897fc27e8cc \
    --hash=sha256:6447e9f3ba72f8e2b985a1da758767698efa72723d5b59accefd716e9e8272bf \
    --hash=sha256:6a9a25751acb379b466ff6be78a315e2b439d4c94c1e99cb7266d40a537995d3 \
    --hash=sha256:6b1a564e6cb69922c7fe3a678b9f9a3c54e72b469875aa8018f18b4d1dd1adf3 \
    --hash=sha256:6d323e1554b3d22cfc03cd3243b5bb815a51f5249fdcbb86fda4bf62bab9e164 \
    --hash=sha256:6e743de5e9c3d1b7185870f480587b75b1cb604832e380d64f9504a0535912d1 \
    --hash=sha256:709fe01086a55cf79d20f741f39325018f4df051ef39fe921b1ebe780a66184c \
    --hash=sha256:7b7c050ae976e286906dd3f26009e117eb000fb2cf3533398c5ad9ccc86867b1 \
    --hash=sha256:7d2872609603cb35ca513d7404a94d6d608fc13211563571117046c9d2bcc3d7 \
    --hash=sha256:7ef58fb89674095bfc57c4069e95d7a31cfdc0939e2a579882ac7d55aadfd2a1 \
    --hash=sha256:80bb5c256f1415f747011dc3604b59bc1f91c6e7150bd7db03b19170ee06b320 \
    --hash=sha256:81b19725065dcb43df02b37e03278c011a09e49757287dca60c5aecdd5a0b8ed \
    --hash=sha256:833b58d5d0b7e5b9832869f039203389ac7cbf01765639c7309fd50ef619e0b1 \
    --hash=sha256:88bd7b6bd70a5b6803c1abf6bca012f7ed963e58c68d76ee20b9d751c74a3248 \
    --hash=sha256:8ad85f7f4e20964db4daadcab70b47ab05c7c1cf2a7c1e51087bfaa83831854c \
    --hash=sha256:8c0ce1e99116d5ab21355d8ebe53d9460366704ea38ae4d9f6933188f327b456 \
    --hash=sha256:8d649d616e5c6a678b26d15ece345354f7c2286acd6db868e65fcc5ff7c24a77 \
    --hash=sha256:903500616422a40a98a5a3c4ff4ed9d0066f3b4c951fa286018ecdf0750194ef \
    --hash=sha256:9736af4641846491aedb3c3f56b9bc5568d92b0692303b5a305301a95dfd38b1 \
    --hash=sha256:988635d122aaf2bdcef9e795435662bcd65b02f4f4c1ae37fbee7401c440b3a7 \
    --hash=sha256:9cca3c2cdadb362116235fdbd411735de4328c61425b0aa9f872fd76d02c4e86 \
    --hash=sha256:9e0fd32e0148dd5dea6af5fee42beb949098564cc23211a88d799e434255a1f4 \
    --hash=sha256:9f3e6f9e05148ff90002b884fbc2a86bd303ae847e472f44ecc06c2cd2fcdb2d \
    --hash=sha256:a85d2b46be66a71bedde836d9e41859879cc54a2a04fad1191eb50c2066f6e9d \
    --hash=sha256:a9008dad07d71f68487c91e96579c8567c98ca4c3881b9b113bc7b33e9fd78b8 \
    --hash=sha256:a9a52172be0b5aae932bef82a79ec0a0ce87288c7d132946d645eba03f0ad8a8 \
    --hash=sha256:aa31fdcc33fef9eb2552cbcbfee7773d5a6792c137b359e82879c101e98584c5 \
    --hash=sha256:acae32e13a4153809db37405f5eba5bac5fbe2e2ba61ab227926a22901051c0a \
    --hash=sha256:b014c23646a467558be7da3d6b9fa409b2c567d2110599b7cf9a0c5992b3b471 \
    --hash=sha256:b21bb4c09ffabfa0e85e3a6b623e19b80e7acd709b9f91452b8297ace2a8ab00 \
    --hash=sha256:b5901a312f4d14c59918c221323068fad0540e34324925c8475263841dbdfe68 \
    --hash=sha256:b9b7a708dd92306328117d8c4b62e2194d00c365f18eff11a9b53c6f923b01e3 \
    --hash=sha256:d1967f46ea8f2db647c786e78d8cc7e4313dbd1b0aca360592d8027b8508e24d \
    --hash=sha256:d52a25136894c63de15a35bc0bdc5adb4b0e173b9c0d07a2be9d3ca64a332735 \
    --hash=sha256:d77c85fedff92cf788face9bfa3ebaa364448ebb1d765302e9af11bf449ca36d \
    --hash=sha256:d79d7d5dc8a32b7093e81e97dad755127ff77bcc899e845f41bf71747af0c569 \
    --hash=sha256:dbcda74c67263139358f4d188ae5faae95c30929281bc6866d00573783c422b7 \
    --hash=sha256:ddaea91abf8b0d13443f6dac52e89051a5063c7d014710dcb4d4abb2ff811a59 \
    --hash=sha256:dee0ce50c6a2dd9056c20db781e9c1cfd33e77d2d569f5d1d9321c641bb903d5 \
    --hash=sha256:dee60e1de1898bde3b238f18340eec6148986da0455d8ba7848d50470a7a32fb \
    --hash=sha256:e2f83e18fe2f4c9e7db597e988f72712c0c3676d337d8b101f6758107c42425b \
    --hash=sha256:e3fb1677c720409d5f671e39bac6c9e0e422584e5f518bfd50aa4cbbea02433f \
    --hash=sha256:ecee4132c6cd2ce5308e21672015ddfed1ff975ad0ac8d27168ea82e71413f55 \
    --hash=sha256:ee2b1b1769f6707a8a445162ea16dddf74285c3964f605877a20e38545c3c462 \
    --hash=sha256:ee6acae74a2b91865910eef5e7de37dc6895ad96fa23603d1d27ea69df545015 \
    --hash=sha256:ef3f72c9666bba2bab70d2a8b79f2c6d2c1a42a7f7e2b0ec83bb2f9e383950af
wtforms==3.1.1 ; python_full_version == "3.10.13" \
    --hash=sha256:5e51df8af9a60f6beead75efa10975e97768825a82146a65c7cbf5b915990620 \
    --hash=sha256:ae7c54b29806c70f7bce8eb9f24afceb10ca5c32af3d9f04f74d2f66ccc5c7e0
```

```
aiohttp==3.8.6
aiosignal==1.3.1
astroid==2.15.5
async-timeout==4.0.2
attrs==23.1.0
bandit==1.7.5
bidict==0.22.1
blinker==1.6.2
certifi==2023.7.22
charset-normalizer==3.1.0
click==8.1.3
dill==0.3.6
dparse==0.6.3
Flask==2.3.2
Flask-SocketIO==5.3.4
frozenlist==1.3.3
gitdb==4.0.10
GitPython==3.1.37
idna==3.4
isort==5.12.0
itsdangerous==2.1.2
Jinja2==3.1.2
lazy-object-proxy==1.9.0
markdown-it-py==3.0.0
MarkupSafe==2.1.2
mccabe==0.7.0
mdurl==0.1.2
multidict==6.0.4
netifaces==0.10.6
openai==0.27.7
packaging==21.3
pbr==5.11.1
pipdeptree==2.13.0
platformdirs==3.5.1
Pygments==2.16.1
pylint==2.17.4
pyparsing==3.1.1
python-dotenv==1.0.0
python-engineio==4.4.1
python-socketio==5.8.0
PyYAML==6.0.1
requests==2.31.0
rich==13.6.0
ruamel.yaml==0.17.35
ruamel.yaml.clib==0.2.8
safety==2.3.5
smmap==5.0.1
stevedore==5.1.0
tomlkit==0.11.8
tqdm==4.65.0
urllib3==2.0.2
Werkzeug==2.3.4
wrapt==1.15.0
```

```
yarl==1.9.2
```

```
Total files processed: 79
File types distribution:
  File: 79
MIME types distribution:
  text/x-python: 71
  Unknown: 1
  text/plain: 7
```

```
 ╔╔════════════════════════════════════════════════════════════════════════════════════════════════════════════
════════════════════════════════════════════════════════════╗╗
  /    \    |                   |-      |- |        / \           /=\         |-
  | =|= | /=\ |  /=: /=\ /=\=\ /=\   |  /=\   |  |=\ /=\   |\| /=\ | | /=|  \  | | /== |  /=\ /=\=\
   \/ \/  \=  \= \=: \=/ | | | \=    \= \=/   \= | | \=    \ / \=/ \\/ \=| \=/ \=| ==/ \= \=  | | |
                                                                  \=|
 ╚╚════════════════════════════════════════════════════════════════════════════════════════════════════════════
════════════════════════════════════════════════════════════╝╝
```

/* Coolors Exported Palette - https://coolors.co/fe4a49-2ab7ca-fed766-e6e6ea-f4f4f8 */

- Tailwind

{ 'tomato': { DEFAULT: '#fe4a49', 100: '#410000', 200: '#820101', 300: '#c30101', 400: '#fe0707', 500: '#fe4a49', 600: '#fe6d6d', 700: '#fe9191', 800: '#ffb6b6', 900: '#ffdada' }, 'moonstone': { DEFAULT: '#2ab7ca', 100: '#082529', 200: '#114a51', 300: '#196f7a', 400: '#2193a3', 500: '#2ab7ca', 600: '#4fcbdb', 700: '#7bd8e4', 800: '#a7e5ed', 900: '#d3f2f6' }, 'mustard': { DEFAULT: '#fed766', 100: '#473500', 200: '#8e6b01', 300: '#d5a001', 400: '#fec620', 500: '#fed766', 600: '#fee085', 700: '#ffe8a4', 800: '#fff0c2', 900: '#fff7e1' }, 'platinum': { DEFAULT: '#e6e6ea', 100: '#2a2a33', 200: '#545465', 300: '#818196', 400: '#b3b3c0', 500: '#e6e6ea', 600: '#ebebee', 700: '#f0f0f2', 800: '#f5f5f7', 900: '#fafafb' }, 'ghost_white': { DEFAULT: '#f4f4f8', 100: '#26263c', 200: '#4c4c77', 300: '#7b7bab', 400: '#b7b7d1', 500: '#f4f4f8', 600: '#f5f5f9', 700: '#f8f8fa', 800: '#fafafc', 900: '#fdfdfd' } }

- CSV

fe4a49,2ab7ca,fed766,e6e6ea,f4f4f8

- With #

#fe4a49, #2ab7ca, #fed766, #e6e6ea, #f4f4f8

- Array

["fe4a49","2ab7ca","fed766","e6e6ea","f4f4f8"]

- Object

{"Tomato":"fe4a49","Moonstone":"2ab7ca","Mustard":"fed766","Platinum":"e6e6ea","Ghost white":"f4f4f8"}

- Extended Array

[{"name":"Tomato","hex":"fe4a49","rgb":[254,74,73],"cmyk":[0,71,71,0],"hsb":[0,71,100],"hsl":[0,99,64],"lab":[58,67,41]},{"name":"Moonstone","hex":"2ab7ca","rgb":[42,183,202],"cmyk":[79,9,0,21],"hsb":[187,79,79],"hsl":[187,66,48],"lab":[68,-29,-20]},{"name":"Mustard","hex":"fed766","rgb":[254,215,102],"cmyk":[0,15,60,0],"hsb":[45,60,100],"hsl":[45,99,70],"lab":[87,1,60]},{"name":"Platinum","hex":"e6e6ea","rgb":[230,230,234],"cmyk":[2,2,0,8],"hsb":[240,2,92],"hsl":[240,9,91],"lab":[91,1,-2]},{"name":"Ghost white","hex":"f4f4f8","rgb":[244,244,248],"cmyk":[2,2,0,3],"hsb":[240,2,97],"hsl":[240,22,96],"lab":[96,1,-2]}]

- XML

```
<palette>
  <color name="Tomato" hex="fe4a49" r="254" g="74" b="73" />
  <color name="Moonstone" hex="2ab7ca" r="42" g="183" b="202" />
  <color name="Mustard" hex="fed766" r="254" g="215" b="102" />
  <color name="Platinum" hex="e6e6ea" r="230" g="230" b="234" />
  <color name="Ghost white" hex="f4f4f8" r="244" g="244" b="248" />
</palette>
```

```
NovaSystem/
    Archive/
        backup_requirements.txt
        testfile.py
        requirements_backup.txt
        update_python_packages.sh
        SECURITY.md
        scratch_version/
            LICENSE
            README.md
            main.py
            brainstorming/
                game_log.py
                LICENSE
                requirements.txt
                game_blocks.py
                cool_main.py
                game.py
                loading_main.py
                busted_game.py
                bot_dungeon.py
                test.py
                stream_ai_response.py
                README.md
                room_block.py
                player.py
                level.py
                brainstorm_app.py
                welcome_ascii_art.py
                welcome_message.py
                robot.py
                boring_main.py
                room.py
            dev/
                pyvenv.cfg
                bin/
                    Activate.ps1
                    dotenv
                    python3
                    python
                    pip3
                    activate.fish
                    python3.11
                    pip
                    tqdm
                    pip3.11
                    activate
                    normalizer
                    openai
                    activate.csh
            apps/
                nova_prototype/
                    versions/
            Tomato/
              nova_tomato.py
              Utils/
                Logger.py
                debugger.py
                return_input_as_string.py
                formatters/
                  openai_message_formatter.py
                DataTransformer/
                  transformers.py
                  class.py
              NovaMessageConstructor/
                nova_messsage_constructor.py
                nova_system_message.py
                get_openai_chat_response.py
                nova_primer_text.py
                messages.py
                nova_continuation_text.py
  NovaSystem_v0.0.1/
    NovaSystem.py
    imports.py
    NovaTribunal.py
    NovaHelper.py
    main.py
  External_Modified_Libs/
    fig_autocomplete/
    Ghost/
  nova_system_tests/
    NovaChatBot.py
    nova_system_stream_test.py
    NovaConfigManager.py
    nova_system_CLI_test.py
    NovaCLI.py
    README.md
    Clog.py
    NovaHelper.py
    nova_chatbot.py
  .vscode/
    settings.json
    extensions.json
  External_Modified_Libs_2/
    README.md
    fig_autocomplete/
    Ghost/
  Media/
    winfonovacolorpalette.svg
    winfonovacolorpalette.txt
    nice_vanilla.svg
    soothing_purple.svg
    nice_bright_red.svg
    boat_colors.svg
    surreal_ice_cream.svg
    dark_purples.svg
bin/
```

```
novagpt-env/
    pyvenv.cfg
    bin/
        Activate.ps1
        python3
        python
        pip3
        activate.fish
        python3.11
        pip
        pip3.11
        activate
        activate.csh
    include/
        python3.11/
    lib/
        python3.11/
            site-packages/
                distutils-precedence.pth
                setuptools-68.1.2.dist-info/
                    RECORD
                    LICENSE
                    WHEEL
                    entry_points.txt
                    top_level.txt
                    REQUESTED
                    INSTALLER
                    METADATA
                pip/
                    __init__.py
                    py.typed
                    __pip-runner__.py
                    __main__.py
                    _internal/
                        configuration.py
                        pyproject.py
                        cache.py
                        __init__.py
                        exceptions.py
                        main.py
                        wheel_builder.py
                        self_outdated_check.py
                        build_env.py
                        network/
                            auth.py
                            xmlrpc.py
                            download.py
                            session.py
                            cache.py
                            __init__.py
                            utils.py
                            lazy_wheel.py
                            __pycache__/
                                session.cpython-311.pyc
                                download.cpython-311.pyc
                                utils.cpython-311.pyc
                                xmlrpc.cpython-311.pyc
                                cache.cpython-311.pyc
                                auth.cpython-311.pyc
                                lazy_wheel.cpython-311.pyc
                                __init__.cpython-311.pyc
                        utils/
                            logging.py
                            misc.py
                            egg_link.py
                            compat.py
                            encoding.py
                            models.py
                            deprecation.py
                            subprocess.py
                            filesystem.py
                            direct_url_helpers.py
                            __init__.py
                            _jaraco_text.py
                            temp_dir.py
                            appdirs.py
                            inject_securetransport.py
                            setuptools_build.py
                            packaging.py
                            entrypoints.py
                            filetypes.py
                            compatibility_tags.py
                            datetime.py
                            urls.py
                            hashes.py
                            virtualenv.py
                            _log.py
                            glibc.py
                            wheel.py
                            unpacking.py
                            __pycache__/
                                _jaraco_text.cpython-311.pyc
                                inject_securetransport.cpython-311.pyc
                                temp_dir.cpython-311.pyc
                                glibc.cpython-311.pyc
                                encoding.cpython-311.pyc
                                filetypes.cpython-311.pyc
                                deprecation.cpython-311.pyc
                                compat.cpython-311.pyc
                                appdirs.cpython-311.pyc
                                logging.cpython-311.pyc
                                packaging.cpython-311.pyc
                                unpacking.cpython-311.pyc
                                setuptools_build.cpython-311.pyc
                                urls.cpython-311.pyc
                                filesystem.cpython-311.pyc
                                misc.cpython-311.pyc
                                compatibility_tags.cpython-311.pyc
                                models.cpython-311.pyc
                                wheel.cpython-311.pyc
                                egg_link.cpython-311.pyc
                                datetime.cpython-311.pyc
                                direct_url_helpers.cpython-311.pyc
                                hashes.cpython-311.pyc
                                entrypoints.cpython-311.pyc
                                subprocess.cpython-311.pyc
                                _log.cpython-311.pyc
                                virtualenv.cpython-311.pyc
                                __init__.cpython-311.pyc
                        models/
                            link.py
                            selection_prefs.py
                            direct_url.py
                            index.py
                            target_python.py
                            __init__.py
                            search_scope.py
                            candidate.py
                            format_control.py
                            installation_report.py
                            scheme.py
                            wheel.py
                            __pycache__/
                                target_python.cpython-311.pyc
                                link.cpython-311.pyc
                                candidate.cpython-311.pyc
                                installation_report.cpython-311.pyc
                                selection_prefs.cpython-311.pyc
                                wheel.cpython-311.pyc
                                search_scope.cpython-311.pyc
                                format_control.cpython-311.pyc
                                index.cpython-311.pyc
                                scheme.cpython-311.pyc
                                __init__.cpython-311.pyc
                                direct_url.cpython-311.pyc
                        __pycache__/
                            main.cpython-311.pyc
                            build_env.cpython-311.pyc
                            self_outdated_check.cpython-311.pyc
                            pyproject.cpython-311.pyc
                            configuration.cpython-311.pyc
                            cache.cpython-311.pyc
                            exceptions.cpython-311.pyc
                            wheel_builder.cpython-311.pyc
                            __init__.cpython-311.pyc
                        cli/
                            cmdoptions.py
                            __init__.py
                            status_codes.py
                            parser.py
                            command_context.py
                            spinners.py
```

```
        autocompletion.py
        base_command.py
        main_parser.py
        progress_bars.py
        main.py
        req_command.py
        __pycache__/
            req_command.cpython-311.pyc
            main.cpython-311.pyc
            main_parser.cpython-311.pyc
            base_command.cpython-311.pyc
            cmdoptions.cpython-311.pyc
            autocompletion.cpython-311.pyc
            status_codes.cpython-311.pyc
            progress_bars.cpython-311.pyc
            command_context.cpython-311.pyc
            parser.cpython-311.pyc
            spinners.cpython-311.pyc
            __init__.cpython-311.pyc
    operations/
        check.py
        __init__.py
        freeze.py
        prepare.py
        install/
            editable_legacy.py
            __init__.py
            wheel.py
            __pycache__/
                wheel.cpython-311.pyc
                editable_legacy.cpython-311.pyc
                __init__.cpython-311.pyc
        __pycache__/
            check.cpython-311.pyc
            freeze.cpython-311.pyc
            prepare.cpython-311.pyc
            __init__.cpython-311.pyc
        build/
            wheel_legacy.py
            metadata.py
            metadata_editable.py
            wheel_editable.py
            __init__.py
            metadata_legacy.py
            wheel.py
            build_tracker.py
            __pycache__/
                wheel_legacy.cpython-311.pyc
                metadata.cpython-311.pyc
                wheel_editable.cpython-311.pyc
                metadata_legacy.cpython-311.pyc
                wheel.cpython-311.pyc
                metadata_editable.cpython-311.pyc
                build_tracker.cpython-311.pyc
                __init__.cpython-311.pyc
    req/
        req_install.py
        req_set.py
        req_uninstall.py
        __init__.py
        req_file.py
        constructors.py
        __pycache__/
            constructors.cpython-311.pyc
            req_install.cpython-311.pyc
            req_set.cpython-311.pyc
            req_uninstall.cpython-311.pyc
            __init__.cpython-311.pyc
            req_file.cpython-311.pyc
    resolution/
        __init__.py
        base.py
        legacy/
            __init__.py
            resolver.py
            __pycache__/
                resolver.cpython-311.pyc
                __init__.cpython-311.pyc
        __pycache__/
            base.cpython-311.pyc
            __init__.cpython-311.pyc
        resolvelib/
            provider.py
            found_candidates.py
            reporter.py
            __init__.py
            factory.py
            requirements.py
            resolver.py
            candidates.py
            base.py
            __pycache__/
                reporter.cpython-311.pyc
                resolver.cpython-311.pyc
                base.cpython-311.pyc
                requirements.cpython-311.pyc
                provider.cpython-311.pyc
                candidates.cpython-311.pyc
                factory.cpython-311.pyc
                found_candidates.cpython-311.pyc
                __init__.cpython-311.pyc
    vcs/
        git.py
        __init__.py
        mercurial.py
        bazaar.py
        versioncontrol.py
        subversion.py
        __pycache__/
            versioncontrol.cpython-311.pyc
            subversion.cpython-311.pyc
            bazaar.cpython-311.pyc
            __init__.cpython-311.pyc
            mercurial.cpython-311.pyc
            git.cpython-311.pyc
    locations/
        __init__.py
        _sysconfig.py
        _distutils.py
        base.py
        __pycache__/
            base.cpython-311.pyc
            _distutils.cpython-311.pyc
            _sysconfig.cpython-311.pyc
            __init__.cpython-311.pyc
    index/
        collector.py
        __init__.py
        sources.py
        package_finder.py
        __pycache__/
            sources.cpython-311.pyc
            collector.cpython-311.pyc
            package_finder.cpython-311.pyc
            __init__.cpython-311.pyc
    commands/
        configuration.py
        show.py
        list.py
        check.py
        index.py
        completion.py
        download.py
        cache.py
        __init__.py
        hash.py
        inspect.py
        debug.py
        uninstall.py
        freeze.py
        search.py
        install.py
        help.py
        wheel.py
        __pycache__/
            search.cpython-311.pyc
            check.cpython-311.pyc
            download.cpython-311.pyc
            list.cpython-311.pyc
            show.cpython-311.pyc
            debug.cpython-311.pyc
            hash.cpython-311.pyc
```

```
                freeze.cpython-311.pyc
                configuration.cpython-311.pyc
                install.cpython-311.pyc
                help.cpython-311.pyc
                cache.cpython-311.pyc
                wheel.cpython-311.pyc
                index.cpython-311.pyc
                completion.cpython-311.pyc
                uninstall.cpython-311.pyc
                inspect.cpython-311.pyc
                __init__.cpython-311.pyc
        metadata/
            _json.py
            __init__.py
            pkg_resources.py
            base.py
            __pycache__/
                base.cpython-311.pyc
                pkg_resources.cpython-311.pyc
                _json.cpython-311.pyc
                __init__.cpython-311.pyc
            importlib/
                _dists.py
                __init__.py
                _compat.py
                _envs.py
                __pycache__/
                    _compat.cpython-311.pyc
                    _envs.cpython-311.pyc
                    _dists.cpython-311.pyc
                    __init__.cpython-311.pyc
        distributions/
            __init__.py
            sdist.py
            installed.py
            base.py
            wheel.py
            __pycache__/
                base.cpython-311.pyc
                installed.cpython-311.pyc
                wheel.cpython-311.pyc
                __init__.cpython-311.pyc
                sdist.cpython-311.pyc
    _vendor/
        vendor.txt
        __init__.py
        six.py
        typing_extensions.py
        packaging/
            tags.py
            _musllinux.py
            version.py
            __init__.py
            utils.py
            requirements.py
            _structures.py
            markers.py
            __about__.py
            _manylinux.py
            specifiers.py
            __pycache__/
                markers.cpython-311.pyc
                requirements.cpython-311.pyc
                _musllinux.cpython-311.pyc
                tags.cpython-311.pyc
                utils.cpython-311.pyc
                __about__.cpython-311.pyc
                _manylinux.cpython-311.pyc
                version.cpython-311.pyc
                specifiers.cpython-311.pyc
                __init__.cpython-311.pyc
                _structures.cpython-311.pyc
        msgpack/
            __init__.py
            exceptions.py
            fallback.py
            ext.py
            __pycache__/
                fallback.cpython-311.pyc
                exceptions.cpython-311.pyc
                ext.cpython-311.pyc
                __init__.cpython-311.pyc
        chardet/
            resultdict.py
            enums.py
            langhungarianmodel.py
            mbcssm.py
            johabfreq.py
            langthaimodel.py
            version.py
            utf1632prober.py
            langbulgarianmodel.py
            euckrprober.py
            sjisprober.py
            cp949prober.py
            __init__.py
            euctwfreq.py
            langhebrewmodel.py
            chardistribution.py
            latin1prober.py
            charsetprober.py
            gb2312prober.py
            mbcharsetprober.py
            euctwprober.py
            langrussianmodel.py
            codingstatemachine.py
            escprober.py
            universaldetector.py
```

```
utf8prober.py
gb2312freq.py
mbcsgroupprober.py
langgreekmodel.py
eucjpprober.py
jisfreq.py
escsm.py
langturkishmodel.py
sbcharsetprober.py
big5freq.py
euckrfreq.py
codingstatemachinedict.py
big5prober.py
johabprober.py
hebrewprober.py
macromanprober.py
charsetgroupprober.py
sbcsgroupprober.py
jpcntx.py
__pycache__/
    sbcsgroupprober.cpython-311.pyc
    macromanprober.cpython-311.pyc
    resultdict.cpython-311.pyc
    jpcntx.cpython-311.pyc
    big5prober.cpython-311.pyc
    sjisprober.cpython-311.pyc
    eucjpprober.cpython-311.pyc
    langthaimodel.cpython-311.pyc
    utf1632prober.cpython-311.pyc
    johabprober.cpython-311.pyc
    codingstatemachinedict.cpython-311.pyc
    mbcssm.cpython-311.pyc
    codingstatemachine.cpython-311.pyc
    euctwprober.cpython-311.pyc
    euckrprober.cpython-311.pyc
    hebrewprober.cpython-311.pyc
    charsetgroupprober.cpython-311.pyc
    utf8prober.cpython-311.pyc
    universaldetector.cpython-311.pyc
    latin1prober.cpython-311.pyc
    langturkishmodel.cpython-311.pyc
    mbcharsetprober.cpython-311.pyc
    big5freq.cpython-311.pyc
    gb2312freq.cpython-311.pyc
    gb2312prober.cpython-311.pyc
    enums.cpython-311.pyc
    escprober.cpython-311.pyc
    langhungarianmodel.cpython-311.pyc
    euckrfreq.cpython-311.pyc
    mbcsgroupprober.cpython-311.pyc
    johabfreq.cpython-311.pyc
    charsetprober.cpython-311.pyc
    jisfreq.cpython-311.pyc
    cp949prober.cpython-311.pyc
```

```
        langgreekmodel.cpython-311.pyc
        euctwfreq.cpython-311.pyc
        sbcharsetprober.cpython-311.pyc
        langhebrewmodel.cpython-311.pyc
        langrussianmodel.cpython-311.pyc
        chardistribution.cpython-311.pyc
        escsm.cpython-311.pyc
        version.cpython-311.pyc
        __init__.cpython-311.pyc
        langbulgarianmodel.cpython-311.pyc
    cli/
        __init__.py
        chardetect.py
        __pycache__/
            chardetect.cpython-311.pyc
            __init__.cpython-311.pyc
    metadata/
        __init__.py
        languages.py
        __pycache__/
            languages.cpython-311.pyc
            __init__.cpython-311.pyc
webencodings/
    labels.py
    mklabels.py
    x_user_defined.py
    __init__.py
    tests.py
    __pycache__/
        x_user_defined.cpython-311.pyc
        mklabels.cpython-311.pyc
        tests.cpython-311.pyc
        labels.cpython-311.pyc
        __init__.cpython-311.pyc
pygments/
    modeline.py
    console.py
    scanner.py
    formatter.py
    token.py
    style.py
    util.py
    sphinxext.py
    cmdline.py
    __init__.py
    unistring.py
    lexer.py
    regexopt.py
    plugin.py
    filter.py
    __main__.py
    filters/
        __init__.py
        __pycache__/
            __init__.cpython-311.pyc
    lexers/
        __init__.py
        python.py
        _mapping.py
        __pycache__/
            python.cpython-311.pyc
            _mapping.cpython-311.pyc
            __init__.cpython-311.pyc
    formatters/
        terminal.py
        html.py
        irc.py
        __init__.py
        other.py
        img.py
        terminal256.py
        rtf.py
        svg.py
        bbcode.py
        pangomarkup.py
        _mapping.py
        groff.py
        latex.py
        __pycache__/
            pangomarkup.cpython-311.pyc
            html.cpython-311.pyc
            svg.cpython-311.pyc
            groff.cpython-311.pyc
            img.cpython-311.pyc
            bbcode.cpython-311.pyc
            latex.cpython-311.pyc
            terminal.cpython-311.pyc
            irc.cpython-311.pyc
            terminal256.cpython-311.pyc
            _mapping.cpython-311.pyc
            __init__.cpython-311.pyc
            other.cpython-311.pyc
            rtf.cpython-311.pyc
    __pycache__/
        scanner.cpython-311.pyc
        regexopt.cpython-311.pyc
        formatter.cpython-311.pyc
        plugin.cpython-311.pyc
        lexer.cpython-311.pyc
        style.cpython-311.pyc
        token.cpython-311.pyc
        modeline.cpython-311.pyc
        unistring.cpython-311.pyc
        cmdline.cpython-311.pyc
        util.cpython-311.pyc
        filter.cpython-311.pyc
        sphinxext.cpython-311.pyc
        console.cpython-311.pyc
```

```
        __init__.cpython-311.pyc
        __main__.cpython-311.pyc
    styles/
        __init__.py
        __pycache__/
            __init__.cpython-311.pyc
distlib/
    w64-arm.exe
    w32.exe
    locators.py
    metadata.py
    version.py
    compat.py
    index.py
    manifest.py
    util.py
    database.py
    t32.exe
    __init__.py
    w64.exe
    markers.py
    resources.py
    t64-arm.exe
    scripts.py
    t64.exe
    wheel.py
    __pycache__/
        markers.cpython-311.pyc
        compat.cpython-311.pyc
        resources.cpython-311.pyc
        metadata.cpython-311.pyc
        wheel.cpython-311.pyc
        scripts.cpython-311.pyc
        util.cpython-311.pyc
        locators.cpython-311.pyc
        index.cpython-311.pyc
        manifest.cpython-311.pyc
        database.cpython-311.pyc
        version.cpython-311.pyc
        __init__.cpython-311.pyc
pyparsing/
    results.py
    unicode.py
    util.py
    actions.py
    __init__.py
    core.py
    common.py
    exceptions.py
    testing.py
    helpers.py
    diagram/
        __init__.py
        __pycache__/
```

```
            __init__.cpython-311.pyc
        __pycache__/
            helpers.cpython-311.pyc
            core.cpython-311.pyc
            unicode.cpython-311.pyc
            testing.cpython-311.pyc
            common.cpython-311.pyc
            actions.cpython-311.pyc
            exceptions.cpython-311.pyc
            util.cpython-311.pyc
            results.cpython-311.pyc
            __init__.cpython-311.pyc
    distro/
        __init__.py
        distro.py
        __main__.py
        __pycache__/
            distro.cpython-311.pyc
            __init__.cpython-311.pyc
            __main__.cpython-311.pyc
    colorama/
        __init__.py
        win32.py
        ansitowin32.py
        ansi.py
        winterm.py
        initialise.py
        tests/
            isatty_test.py
            initialise_test.py
            __init__.py
            ansi_test.py
            utils.py
            winterm_test.py
            ansitowin32_test.py
            __pycache__/
                winterm_test.cpython-311.pyc
                initialise_test.cpython-311.pyc
                isatty_test.cpython-311.pyc
                ansi_test.cpython-311.pyc
                utils.cpython-311.pyc
                ansitowin32_test.cpython-311.pyc
                __init__.cpython-311.pyc
        __pycache__/
            ansi.cpython-311.pyc
            ansitowin32.cpython-311.pyc
            initialise.cpython-311.pyc
            win32.cpython-311.pyc
            winterm.cpython-311.pyc
            __init__.cpython-311.pyc
    cachecontrol/
        serialize.py
        wrapper.py
        controller.py
        compat.py
        filewrapper.py
        heuristics.py
        adapter.py
        cache.py
        __init__.py
        _cmd.py
        __pycache__/
            controller.cpython-311.pyc
            serialize.cpython-311.pyc
            filewrapper.cpython-311.pyc
            _cmd.cpython-311.pyc
            compat.cpython-311.pyc
            adapter.cpython-311.pyc
            cache.cpython-311.pyc
            wrapper.cpython-311.pyc
            heuristics.cpython-311.pyc
            __init__.cpython-311.pyc
        caches/
            file_cache.py
            __init__.py
            redis_cache.py
            __pycache__/
                file_cache.cpython-311.pyc
                redis_cache.cpython-311.pyc
                __init__.cpython-311.pyc
    idna/
        intranges.py
        package_data.py
        compat.py
        idnadata.py
        __init__.py
        core.py
        codec.py
        uts46data.py
        __pycache__/
            codec.cpython-311.pyc
            core.cpython-311.pyc
            compat.cpython-311.pyc
            idnadata.cpython-311.pyc
            package_data.cpython-311.pyc
            uts46data.cpython-311.pyc
            intranges.cpython-311.pyc
            __init__.cpython-311.pyc
    tenacity/
        before.py
        before_sleep.py
        _asyncio.py
        stop.py
        wait.py
        __init__.py
        nap.py
        after.py
        retry.py
        tornadoweb.py
        _utils.py
        __pycache__/
            wait.cpython-311.pyc
            before_sleep.cpython-311.pyc
            stop.cpython-311.pyc
            nap.cpython-311.pyc
            before.cpython-311.pyc
            after.cpython-311.pyc
            _utils.cpython-311.pyc
            tornadoweb.cpython-311.pyc
            _asyncio.cpython-311.pyc
            retry.cpython-311.pyc
            __init__.cpython-311.pyc
    __pycache__/
        six.cpython-311.pyc
        typing_extensions.cpython-311.pyc
        __init__.cpython-311.pyc
    requests/
        cookies.py
        auth.py
        sessions.py
        hooks.py
        compat.py
        models.py
        certs.py
        __init__.py
        status_codes.py
        packages.py
        __version__.py
        api.py
        _internal_utils.py
        utils.py
        exceptions.py
        structures.py
        help.py
        adapters.py
        __pycache__/
            api.cpython-311.pyc
            __version__.cpython-311.pyc
            adapters.cpython-311.pyc
            compat.cpython-311.pyc
            _internal_utils.cpython-311.pyc
            cookies.cpython-311.pyc
            status_codes.cpython-311.pyc
            utils.cpython-311.pyc
            hooks.cpython-311.pyc
            sessions.cpython-311.pyc
            help.cpython-311.pyc
            models.cpython-311.pyc
            exceptions.cpython-311.pyc
            auth.cpython-311.pyc
            packages.cpython-311.pyc
            certs.cpython-311.pyc
            structures.cpython-311.pyc
            __init__.cpython-311.pyc
    tomli/
        _types.py
        __init__.py
        _parser.py
        _re.py
        __pycache__/
            _types.cpython-311.pyc
            _re.cpython-311.pyc
            _parser.cpython-311.pyc
            __init__.cpython-311.pyc
    certifi/
        __init__.py
        core.py
        cacert.pem
        __main__.py
        __pycache__/
            core.cpython-311.pyc
            __init__.cpython-311.pyc
            __main__.cpython-311.pyc
    pyproject_hooks/
        _impl.py
        __init__.py
        _compat.py
        __pycache__/
            _impl.cpython-311.pyc
            _compat.cpython-311.pyc
            __init__.cpython-311.pyc
        _in_process/
            _in_process.py
            __init__.py
            __pycache__/
                _in_process.cpython-311.pyc
                __init__.cpython-311.pyc
    rich/
        themes.py
        screen.py
        logging.py
        measure.py
        tree.py
        console.py
        live_render.py
        _emoji_codes.py
        box.py
        color.py
        _timer.py
        _fileno.py
        align.py
        theme.py
        style.py
        default_styles.py
        _wrap.py
        _log_render.py
```

emoji.py
layout.py
containers.py
_emoji_replace.py
traceback.py
region.py
protocol.py
_loop.py
control.py
filesize.py
_null_file.py
_palettes.py
__init__.py
_pick.py
file_proxy.py
palette.py
markup.py
_ratio.py
repr.py
constrain.py
pretty.py
diagnose.py
columns.py
rule.py
_inspect.py
pager.py
text.py
highlighter.py
_spinners.py
terminal_theme.py
bar.py
live.py
syntax.py
table.py
_export_format.py
progress_bar.py
errors.py
prompt.py
segment.py
ansi.py
progress.py
_stack.py
_windows.py
_cell_widths.py
cells.py
_win32_console.py
panel.py
styled.py
spinner.py
_windows_renderer.py
json.py
padding.py
__main__.py
scope.py

```
_extension.py
status.py
abc.py
jupyter.py
color_triplet.py
__pycache__/
    theme.cpython-311.pyc
    color.cpython-311.pyc
    segment.cpython-311.pyc
    _emoji_replace.cpython-311.pyc
    highlighter.cpython-311.pyc
    _emoji_codes.cpython-311.pyc
    layout.cpython-311.pyc
    _ratio.cpython-311.pyc
    ansi.cpython-311.pyc
    file_proxy.cpython-311.pyc
    cells.cpython-311.pyc
    containers.cpython-311.pyc
    rule.cpython-311.pyc
    constrain.cpython-311.pyc
    _windows.cpython-311.pyc
    _log_render.cpython-311.pyc
    style.cpython-311.pyc
    _palettes.cpython-311.pyc
    terminal_theme.cpython-311.pyc
    _cell_widths.cpython-311.pyc
    logging.cpython-311.pyc
    _loop.cpython-311.pyc
    pretty.cpython-311.pyc
    abc.cpython-311.pyc
    scope.cpython-311.pyc
    default_styles.cpython-311.pyc
    _spinners.cpython-311.pyc
    bar.cpython-311.pyc
    errors.cpython-311.pyc
    progress.cpython-311.pyc
    prompt.cpython-311.pyc
    _fileno.cpython-311.pyc
    text.cpython-311.pyc
    live_render.cpython-311.pyc
    filesize.cpython-311.pyc
    _inspect.cpython-311.pyc
    control.cpython-311.pyc
    _stack.cpython-311.pyc
    emoji.cpython-311.pyc
    themes.cpython-311.pyc
    region.cpython-311.pyc
    palette.cpython-311.pyc
    screen.cpython-311.pyc
    measure.cpython-311.pyc
    _extension.cpython-311.pyc
    tree.cpython-311.pyc
    repr.cpython-311.pyc
    status.cpython-311.pyc
```

```
        protocol.cpython-311.pyc
        markup.cpython-311.pyc
        _windows_renderer.cpython-311.pyc
        jupyter.cpython-311.pyc
        align.cpython-311.pyc
        live.cpython-311.pyc
        _wrap.cpython-311.pyc
        table.cpython-311.pyc
        panel.cpython-311.pyc
        syntax.cpython-311.pyc
        progress_bar.cpython-311.pyc
        pager.cpython-311.pyc
        _export_format.cpython-311.pyc
        json.cpython-311.pyc
        styled.cpython-311.pyc
        spinner.cpython-311.pyc
        _timer.cpython-311.pyc
        _pick.cpython-311.pyc
        padding.cpython-311.pyc
        color_triplet.cpython-311.pyc
        console.cpython-311.pyc
        _win32_console.cpython-311.pyc
        _null_file.cpython-311.pyc
        traceback.cpython-311.pyc
        __init__.cpython-311.pyc
        diagnose.cpython-311.pyc
        columns.cpython-311.pyc
        box.cpython-311.pyc
        __main__.cpython-311.pyc
urllib3/
    filepost.py
    fields.py
    _version.py
    request.py
    __init__.py
    poolmanager.py
    response.py
    connection.py
    _collections.py
    exceptions.py
    connectionpool.py
    util/
        queue.py
        ssltransport.py
        proxy.py
        wait.py
        request.py
        timeout.py
        __init__.py
        response.py
        ssl_.py
        retry.py
        url.py
        connection.py
    ssl_match_hostname.py
    __pycache__/
        wait.cpython-311.pyc
        queue.cpython-311.pyc
        ssltransport.cpython-311.pyc
        response.cpython-311.pyc
        ssl_match_hostname.cpython-311.pyc
        proxy.cpython-311.pyc
        timeout.cpython-311.pyc
        connection.cpython-311.pyc
        url.cpython-311.pyc
        retry.cpython-311.pyc
        __init__.cpython-311.pyc
        request.cpython-311.pyc
        ssl_.cpython-311.pyc
__pycache__/
    poolmanager.cpython-311.pyc
    filepost.cpython-311.pyc
    fields.cpython-311.pyc
    connectionpool.cpython-311.pyc
    response.cpython-311.pyc
    _version.cpython-311.pyc
    exceptions.cpython-311.pyc
    connection.cpython-311.pyc
    _collections.cpython-311.pyc
    __init__.cpython-311.pyc
    request.cpython-311.pyc
contrib/
    securetransport.py
    __init__.py
    socks.py
    _appengine_environ.py
    pyopenssl.py
    appengine.py
    ntlmpool.py
    __pycache__/
        securetransport.cpython-311.pyc
        pyopenssl.cpython-311.pyc
        ntlmpool.cpython-311.pyc
        appengine.cpython-311.pyc
        _appengine_environ.cpython-311.pyc
        socks.cpython-311.pyc
        __init__.cpython-311.pyc
    _securetransport/
        __init__.py
        low_level.py
        bindings.py
        __pycache__/
            bindings.cpython-311.pyc
            low_level.cpython-311.pyc
            __init__.cpython-311.pyc
packages/
    __init__.py
    six.py
```

```
            __pycache__/
                six.cpython-311.pyc
                __init__.cpython-311.pyc
            backports/
                __init__.py
                makefile.py
                weakref_finalize.py
                __pycache__/
                    weakref_finalize.cpython-311.pyc
                    __init__.cpython-311.pyc
                    makefile.cpython-311.pyc
    pkg_resources/
        __init__.py
        __pycache__/
            __init__.cpython-311.pyc
    resolvelib/
        resolvers.py
        __init__.py
        providers.py
        structs.py
        reporters.py
        compat/
            __init__.py
            collections_abc.py
            __pycache__/
                collections_abc.cpython-311.pyc
                __init__.cpython-311.pyc
        __pycache__/
            reporters.cpython-311.pyc
            resolvers.cpython-311.pyc
            providers.cpython-311.pyc
            structs.cpython-311.pyc
            __init__.cpython-311.pyc
    platformdirs/
        macos.py
        unix.py
        version.py
        __init__.py
        api.py
        android.py
        windows.py
        __main__.py
        __pycache__/
            api.cpython-311.pyc
            android.cpython-311.pyc
            windows.cpython-311.pyc
            macos.cpython-311.pyc
            unix.cpython-311.pyc
            version.cpython-311.pyc
            __init__.cpython-311.pyc
            __main__.cpython-311.pyc
__pycache__/
    __pip-runner__.cpython-311.pyc
    __init__.cpython-311.pyc
```

```
        __main__.cpython-311.pyc
pip-23.2.1.dist-info/
    RECORD
    WHEEL
    entry_points.txt
    top_level.txt
    LICENSE.txt
    AUTHORS.txt
    REQUESTED
    INSTALLER
    METADATA
setuptools/
    _path.py
    cli-arm64.exe
    logging.py
    windows_support.py
    _normalization.py
    package_index.py
    archive_util.py
    _imp.py
    version.py
    discovery.py
    warnings.py
    py312compat.py
    cli-64.exe
    _reqs.py
    gui-64.exe
    depends.py
    __init__.py
    installer.py
    glob.py
    sandbox.py
    script.tmpl
    launch.py
    extension.py
    unicode_utils.py
    _itertools.py
    monkey.py
    build_meta.py
    cli.exe
    errors.py
    dep_util.py
    msvc.py
    _importlib.py
    _entry_points.py
    cli-32.exe
    gui-32.exe
    gui.exe
    dist.py
    wheel.py
    gui-arm64.exe
    namespaces.py
    script (dev).tmpl
    _vendor/
```

```
__init__.py
zipp.py
ordered_set.py
typing_extensions.py
packaging/
    tags.py
    _musllinux.py
    metadata.py
    version.py
    __init__.py
    _parser.py
    utils.py
    requirements.py
    _structures.py
    markers.py
    _manylinux.py
    _tokenizer.py
    specifiers.py
    _elffile.py
    __pycache__/
        markers.cpython-311.pyc
        requirements.cpython-311.pyc
        _musllinux.cpython-311.pyc
        tags.cpython-311.pyc
        utils.cpython-311.pyc
        metadata.cpython-311.pyc
        _tokenizer.cpython-311.pyc
        _manylinux.cpython-311.pyc
        _parser.cpython-311.pyc
        _elffile.cpython-311.pyc
        version.cpython-311.pyc
        specifiers.cpython-311.pyc
        __init__.cpython-311.pyc
        _structures.cpython-311.pyc
jaraco/
    functools.py
    __init__.py
    context.py
    __pycache__/
        functools.cpython-311.pyc
        context.cpython-311.pyc
        __init__.cpython-311.pyc
    text/
        __init__.py
        __pycache__/
            __init__.cpython-311.pyc
importlib_metadata/
    _meta.py
    _text.py
    __init__.py
    _functools.py
    _py39compat.py
    _collections.py
    _itertools.py
    _adapters.py
    _compat.py
    __pycache__/
        _functools.cpython-311.pyc
        _compat.cpython-311.pyc
        _text.cpython-311.pyc
        _meta.cpython-311.pyc
        _adapters.cpython-311.pyc
        _py39compat.cpython-311.pyc
        _itertools.cpython-311.pyc
        _collections.cpython-311.pyc
        __init__.cpython-311.pyc
__pycache__/
    ordered_set.cpython-311.pyc
    typing_extensions.cpython-311.pyc
    __init__.cpython-311.pyc
    zipp.cpython-311.pyc
more_itertools/
    __init__.py
    more.py
    recipes.py
    __pycache__/
        more.cpython-311.pyc
        recipes.cpython-311.pyc
        __init__.cpython-311.pyc
importlib_resources/
    readers.py
    _common.py
    __init__.py
    _itertools.py
    _adapters.py
    _compat.py
    _legacy.py
    simple.py
    abc.py
    __pycache__/
        simple.cpython-311.pyc
        _legacy.cpython-311.pyc
        _compat.cpython-311.pyc
        _common.cpython-311.pyc
        abc.cpython-311.pyc
        _adapters.cpython-311.pyc
        _itertools.cpython-311.pyc
        __init__.cpython-311.pyc
        readers.cpython-311.pyc
tomli/
    _types.py
    __init__.py
    _parser.py
    _re.py
    __pycache__/
        _types.cpython-311.pyc
        _re.cpython-311.pyc
        _parser.cpython-311.pyc
```

```
            __init__.cpython-311.pyc
    config/
        __init__.py
        setupcfg.py
        _apply_pyprojecttoml.py
        pyprojecttoml.py
        expand.py
        __pycache__/
            expand.cpython-311.pyc
            _apply_pyprojecttoml.cpython-311.pyc
            setupcfg.cpython-311.pyc
            pyprojecttoml.cpython-311.pyc
            __init__.cpython-311.pyc
        _validate_pyproject/
            fastjsonschema_exceptions.py
            extra_validations.py
            error_reporting.py
            __init__.py
            fastjsonschema_validations.py
            formats.py
            __pycache__/
                formats.cpython-311.pyc
                extra_validations.cpython-311.pyc
                fastjsonschema_validations.cpython-311.pyc
                fastjsonschema_exceptions.cpython-311.pyc
                error_reporting.cpython-311.pyc
                __init__.cpython-311.pyc
    __pycache__/
        discovery.cpython-311.pyc
        sandbox.cpython-311.pyc
        dist.cpython-311.pyc
        _normalization.cpython-311.pyc
        _entry_points.cpython-311.pyc
        windows_support.cpython-311.pyc
        msvc.cpython-311.pyc
        namespaces.cpython-311.pyc
        launch.cpython-311.pyc
        unicode_utils.cpython-311.pyc
        glob.cpython-311.pyc
        logging.cpython-311.pyc
        py312compat.cpython-311.pyc
        errors.cpython-311.pyc
        _path.cpython-311.pyc
        warnings.cpython-311.pyc
        _importlib.cpython-311.pyc
        wheel.cpython-311.pyc
        _reqs.cpython-311.pyc
        archive_util.cpython-311.pyc
        dep_util.cpython-311.pyc
        extension.cpython-311.pyc
        _imp.cpython-311.pyc
        installer.cpython-311.pyc
        monkey.cpython-311.pyc
        depends.cpython-311.pyc
```

```
  build_meta.cpython-311.pyc
  _itertools.cpython-311.pyc
  package_index.cpython-311.pyc
  version.cpython-311.pyc
  __init__.cpython-311.pyc
command/
  build.py
  bdist_egg.py
  alias.py
  build_ext.py
  easy_install.py
  editable_wheel.py
  launcher manifest.xml
  install_scripts.py
  upload.py
  register.py
  dist_info.py
  install_lib.py
  upload_docs.py
  build_py.py
  __init__.py
  sdist.py
  test.py
  saveopts.py
  bdist_rpm.py
  build_clib.py
  egg_info.py
  install.py
  develop.py
  rotate.py
  install_egg_info.py
  setopt.py
  __pycache__/
    editable_wheel.cpython-311.pyc
    rotate.cpython-311.pyc
    register.cpython-311.pyc
    setopt.cpython-311.pyc
    develop.cpython-311.pyc
    upload_docs.cpython-311.pyc
    saveopts.cpython-311.pyc
    easy_install.cpython-311.pyc
    build_clib.cpython-311.pyc
    dist_info.cpython-311.pyc
    alias.cpython-311.pyc
    bdist_rpm.cpython-311.pyc
    install_lib.cpython-311.pyc
    build_ext.cpython-311.pyc
    test.cpython-311.pyc
    build.cpython-311.pyc
    bdist_egg.cpython-311.pyc
    install.cpython-311.pyc
    upload.cpython-311.pyc
    install_egg_info.cpython-311.pyc
    install_scripts.cpython-311.pyc
```

```
            egg_info.cpython-311.pyc
            build_py.cpython-311.pyc
            __init__.cpython-311.pyc
            sdist.cpython-311.pyc
    extern/
        __init__.py
        __pycache__/
            __init__.cpython-311.pyc
    _distutils/
        _msvccompiler.py
        unixccompiler.py
        filelist.py
        ccompiler.py
        msvc9compiler.py
        archive_util.py
        cmd.py
        config.py
        version.py
        log.py
        util.py
        fancy_getopt.py
        versionpredicate.py
        __init__.py
        file_util.py
        core.py
        _functools.py
        _collections.py
        cygwinccompiler.py
        extension.py
        debug.py
        spawn.py
        text_file.py
        msvccompiler.py
        errors.py
        dep_util.py
        dir_util.py
        sysconfig.py
        _macos_compat.py
        py39compat.py
        py38compat.py
        dist.py
        _log.py
        bcppcompiler.py
        __pycache__/
            msvccompiler.cpython-311.pyc
            bcppcompiler.cpython-311.pyc
            msvc9compiler.cpython-311.pyc
            fancy_getopt.cpython-311.pyc
            cmd.cpython-311.pyc
            dist.cpython-311.pyc
            _functools.cpython-311.pyc
            core.cpython-311.pyc
            filelist.cpython-311.pyc
            _macos_compat.cpython-311.pyc
            spawn.cpython-311.pyc
            config.cpython-311.pyc
            py38compat.cpython-311.pyc
            dir_util.cpython-311.pyc
            ccompiler.cpython-311.pyc
            text_file.cpython-311.pyc
            versionpredicate.cpython-311.pyc
            debug.cpython-311.pyc
            _msvccompiler.cpython-311.pyc
            errors.cpython-311.pyc
            file_util.cpython-311.pyc
            sysconfig.cpython-311.pyc
            unixccompiler.cpython-311.pyc
            cygwinccompiler.cpython-311.pyc
            archive_util.cpython-311.pyc
            dep_util.cpython-311.pyc
            extension.cpython-311.pyc
            log.cpython-311.pyc
            util.cpython-311.pyc
            py39compat.cpython-311.pyc
            _collections.cpython-311.pyc
            version.cpython-311.pyc
            _log.cpython-311.pyc
            __init__.cpython-311.pyc
        command/
            build.py
            py37compat.py
            build_ext.py
            config.py
            clean.py
            check.py
            install_scripts.py
            upload.py
            register.py
            _framework_compat.py
            install_headers.py
            install_lib.py
            build_py.py
            bdist_dumb.py
            __init__.py
            sdist.py
            bdist.py
            build_scripts.py
            bdist_rpm.py
            build_clib.py
            install.py
            install_egg_info.py
            install_data.py
            __pycache__/
                install_headers.cpython-311.pyc
                register.cpython-311.pyc
                _framework_compat.cpython-311.pyc
                check.cpython-311.pyc
                build_clib.cpython-311.pyc
```

```
                config.cpython-311.pyc
                install_data.cpython-311.pyc
                bdist_rpm.cpython-311.pyc
                install_lib.cpython-311.pyc
                build_ext.cpython-311.pyc
                build.cpython-311.pyc
                install.cpython-311.pyc
                upload.cpython-311.pyc
                install_egg_info.cpython-311.pyc
                py37compat.cpython-311.pyc
                bdist_dumb.cpython-311.pyc
                install_scripts.cpython-311.pyc
                clean.cpython-311.pyc
                build_py.cpython-311.pyc
                bdist.cpython-311.pyc
                __init__.cpython-311.pyc
                build_scripts.cpython-311.pyc
                sdist.cpython-311.pyc
    pkg_resources/
        __init__.py
        _vendor/
            __init__.py
            zipp.py
            typing_extensions.py
            packaging/
                tags.py
                _musllinux.py
                metadata.py
                version.py
                __init__.py
                _parser.py
                utils.py
                requirements.py
                _structures.py
                markers.py
                _manylinux.py
                _tokenizer.py
                specifiers.py
                _elffile.py
                __pycache__/
                    markers.cpython-311.pyc
                    requirements.cpython-311.pyc
                    _musllinux.cpython-311.pyc
                    tags.cpython-311.pyc
                    utils.cpython-311.pyc
                    metadata.cpython-311.pyc
                    _tokenizer.cpython-311.pyc
                    _manylinux.cpython-311.pyc
                    _parser.cpython-311.pyc
                    _elffile.cpython-311.pyc
                    version.cpython-311.pyc
                    specifiers.cpython-311.pyc
                    __init__.cpython-311.pyc
                    _structures.cpython-311.pyc
```

```
jaraco/
    functools.py
    __init__.py
    context.py
    __pycache__/
        functools.cpython-311.pyc
        context.cpython-311.pyc
        __init__.cpython-311.pyc
    text/
        __init__.py
        __pycache__/
            __init__.cpython-311.pyc
__pycache__/
    typing_extensions.cpython-311.pyc
    __init__.cpython-311.pyc
    zipp.cpython-311.pyc
more_itertools/
    __init__.py
    more.py
    recipes.py
    __pycache__/
        more.cpython-311.pyc
        recipes.cpython-311.pyc
        __init__.cpython-311.pyc
importlib_resources/
    readers.py
    _common.py
    __init__.py
    _itertools.py
    _adapters.py
    _compat.py
    _legacy.py
    simple.py
    abc.py
    __pycache__/
        simple.cpython-311.pyc
        _legacy.cpython-311.pyc
        _compat.cpython-311.pyc
        _common.cpython-311.pyc
        abc.cpython-311.pyc
        _adapters.cpython-311.pyc
        _itertools.cpython-311.pyc
        __init__.cpython-311.pyc
        readers.cpython-311.pyc
platformdirs/
    macos.py
    unix.py
    version.py
    __init__.py
    api.py
    android.py
    windows.py
    __main__.py
    __pycache__/
```

```
                api.cpython-311.pyc
                android.cpython-311.pyc
                windows.cpython-311.pyc
                macos.cpython-311.pyc
                unix.cpython-311.pyc
                version.cpython-311.pyc
                __init__.cpython-311.pyc
                __main__.cpython-311.pyc
          __pycache__/
            __init__.cpython-311.pyc
          extern/
            __init__.py
            __pycache__/
              __init__.cpython-311.pyc
        _distutils_hack/
          __init__.py
          override.py
          __pycache__/
            override.cpython-311.pyc
            __init__.cpython-311.pyc
tests/
  nova-test.py
docs/
lib/
.git/
  config
  HEAD
  description
  index
  packed-refs
  COMMIT_EDITMSG
  FETCH_HEAD
  objects/
    61/
      bf74269ade264098bc7b237c7aa17b4b6242c4
      22a020babb9aeb926cbfe94624c1341dcee25a
      a3583834122678a47960547be514370088b131
    0d/
      0c5e6ad9a2ae7f89016134225bded2d6da0937
      9fa24ba245c4638b0b0ca2357a244cdda2dc09
      eb746ac91ed0b526fe8a522ce2a951e2ea5162
      dbf12154b7b1eb70156c4f461c05f29a5e0f81
      8a11797c68cfaec99be817987dc09b842d8d06
      45dff2ec4456529a0e600f8fe081e6c758b812
      5e0aae8edd4cd037fa7f2d46f9a90343579b5c
      a34ec57f9e361c74bf858e941bffb44ce9f18d
      323844a603422db2401a7cb87be09fd9dc648b
    95/
      7538f4752525d1b2cf04ccdce43be0fa4f64fa
      1e69b865f428271542bb5a9d2ff249bcc40a49
      3c272168beadf6c05bcfae9499c78b13bbf5dd
      7235109990e45b411ebb2010c89f72c43b72d5
      1d5817c9e6d81c94a173a0d9fead7f1f143331
      e509c0143e14e6371ec3cd1433ffec50c297fc
      82fa730f121634348a79c1a8b0cc2df99c616f
    59/
      a01d91b87d4282bede38ade7cc78c0f7552d0e
      9154ba2c1d03ca0287b2941cd74356d873742f
      59531e14811aa89b19ea4ccff15c63dcaed2f7
      48570178f3e6e79d1ff574241d09d4d8ed78de
      3bff23edecd3c517c96e119ee777bd4ee1d9d0
      54a79111e2f92266956794982bdef8c094fea2
      72a96d8ded85cc14147ffc1400ec67c3b5a578
      224e71e50c49e5f9f6f925837597c035a8ab7f
      849899db1f4a6c4d585cca1d655b69a183dd53
      91326115fe5026470165b387ba2bc78bceb006
    92/
      746029e25984ed52a4a3954a7681c11c6a5097
      bd93179c5cd3cb377c8b9f1e9d22d13fd7d003
      654beca7008ee8486132892ebf7e792d656ee0
      ae4c7640b7aff00370eda9ab0b3e0ffe506e85
      abd6d86e0eb8a65dd703acf9aa8da359676447
      7eeaddd2fbf75792c1d2479119adeb1ff2892a
      453ec5ca01c0b2bd3a1393be58a2a011e9c065
      67cf312f67a3cae425c162ef77d81a688a2bd2
      c4c6a193873ce09629f6cfaa2dabc4f14ecb03
    0c/
      ec12a03efdbd4dd0e03e79b3fdac6a90f7d0ca
      75f9bd774c4fd11e3ff6473790a18f347531ed
      01d5b08b6b44379b931d54d7fcf5221fdc9fde
      fe5d1612e0cb9bb233ee452a9171bd08b50f2c
      b805fe331b53365cd8a9645a133986cfa67e51
    66/
      443971dfdc53bd58e3aa141d0cf48cda0daf2f
      548003d6ec43293e38ecb319dcc4d927204ea3
      49a71f06ffb35cf2ef7000c044e3c6f51f9f94
      21549b8449130d2d01ebac0a3649d8b70c4f91
      f3a18d5788431a6f4f31dd5b9a19c725371b07
      88da3d3e826d00eed68ec3e688dde2122f0050
      9a3a7074f9a9e1af29cb4bc78b05851df67959
      621fc0c4a5f19be6f3b1b07a8983a6b40833da
      365e6536080bd9372d2a7a58b8ffa3447fec34
      b39b60946a5d6e3a50b268d43895c5c7f28c51
      6307e8fe0608c69f2b6578a49794e1e20a139a
      33ef7ed7c27ba8e228b254d1329bffec675ad6
      a8190a65954aa22f4f085513067e54fbaf43a3
    3e/
      6b954e3f4ce524ef9eafdc439d31dc8e767222
      5e110fb769c94cd19d197d923e215baac0e2d7
      d608b479dbbaa4a0fc92e1f7d9b593188bc0b9
      d1856f6fcc1e5fa319456f42de5cee65132349
      a50eebfe3f0113b231a318cc1ad6e238afd60d
      7f05d6b5003a87f6ee1af9460a2dc7ca60e3dc
      da6a62d5709ff5c860fd85d33cb368a021e033
      bbbc4ccbe47043eb62f8dd770f079745d3b743
    50/
      35e0b25c9d76174bee1066de6ef9d00356f49a
      9477b9cace41371a0ee9c58222c79ab24271d5
      63c3f8ee7980493efcc30c24f7e7582714aa81
      51ad2e498036694a64468e87668afb0a2240f5
    68/
      2b1e5bc36b2f245289447f97e27f5e89486c4c
      bebaf5058c13a17b2fa08d33fd449c979289e1
      c9a00a063e31ccc7c99c0cca7b0163bdbea558
      8b5e10d8608fdb324c5df0ec3d9f4aa720de0e
    57/
      63076d2878093971a0ef9870e1cde7f556b18b
      b6f09bfd83d92cd82e90a87850aae535693eec
      4c9bcea6e222ea8283a3c8dafbda15a2893fe1
      4c4bd78316c7109a346df0bcdb4c1d6ff9bfbc
      85c1694598ecd8aed5bc04ed5317946c8598f2
      605cf0cc59c6a462ab33eaac0e5d5091a1ffcd
      dce9534c203f5621450bbc3ac7235a57596807
      85a6da19475dcaa200f71159f64aab2f27abf7
      0337664835d01904c8ff708626b447edc5640a
      09eb6d8c72fb57f1bc6994dda0110b96e9da45
      5e49fb4b14f9876f4e8fec076b1f5c9b6c72c3
    3b/
      aa1f3c248541e277e2cc8b3cc2527f105502bb
      cdb64eef6c30d92528ab8094af0fa46703dec2
      018dce677d109f5469a6daa45de4dd455bfe07
      765ba7586caee199c0d26844cc631af1f51eef
      7517cd2b339c25433368a953f2fa1323f0a807
      887dc5a41e550047477a66a3b4838d9ef2f515
    6f/
      57d23e71b4caf1209a3096cea6baa0f1fb1f25
      6e2c2c69d25dba4d1038a2d548fbf68017f91b
      b19b30bb53c18f38a9ef02dd7c4478670fb962
      d307f430ca8fcef4808c63969b9f6e4ac0dba8
      05b817e1e9d118b32e6c9b0493e8af619603f5
      fb28aa55afafe035ead5dc17e99dfa50840e75
    03/
      ed925b246dd551ec2ef45095ed6cad00fd2745
      21f6dd895c5a7bb537f3046219db8207a54130
      1cb77c17ba8d8a983448268851d612e05e80d1
      aac775b53f2dd3153a9f44829e7987258950aa
      7a75a77619ff77439e4388e00ddc7e9ba81ee3
      dd10e10a6b58ec1ab4a6ed768fe41f1a3ab46f
      31297b85b89c3387c3868d6254f420ed6a0381
      13049763bb09475051eff9841059fbbfa7d13f
    9b/
      4c8b116636ff5f554594d010846fe636e1fae9
      a31dd4db709ffa03c005874dc3ab7da99f2baf
      e75ecc603c258d6e00cf512489e10786ba99e3
      6d129ed690361770738bec73f44ba7e10a21c5
      87f8682079343f0356adb77cf50b5791330a19
    9e/
      b6922c670630e44403af9e08de7b5199022d0a
      9c0448651861876dc5a0b7c6aa0d2cbc1a110b
      1a4768b2e8016927e0b22d442bf525f92cd1bb
      47d0d7110ce83e217b37bf2bac8d436ec0e192
    04/
```

```
    2dac813e74b8187c3754cb9a937c7f7183e331
    1ca891f9a86d73e2ee1486f7b45c482d3a3924
    476607c8a1c319f6986b29d761c98871f56c06
    445f4483dde7f5f72e4e8467cce402d2b551fe
6a/
    fb5c627ce3db6e61cbf46276f7ddd42552eb28
    8ec15c7ffe86fcc18a3280e2d7466da5340c43
    f1138f260e4eaaa0aa242f7f50b918a283b49f
    8e056875403f0f3777aa9ef64e91323662b95e
    254c1c5e2584dae80f58d38e9a48aae7ec1237
    3b8e6c213a9a8069b38729ab3a0c16a213ce62
    0d6dd12e36092c1497f5390470f85b1afbbb17
    cb05b15553f6c214e8f3a1835c241cbe970888
    43b0475347cb50d0d65ada1000a82eeca9e882
32/
    9d8340a9300e54643f97d681f49f44233a9b4b
    a2f303296d1bffce917413b2e7150a5194e32f
    ff65e783ba2877abf3a6025f8232888b124507
    93576e012a1c931b5e89ebc065c67b65941084
    ba474f1c2de4bef189dd9640c0b75bb66bca3e
35/
    9948748ba316bb2ea51dffdf01e6a552c654bd
    86c783eb7ede4302984cedc2704759f2ff2ec5
    cfcea90282251cd13a3532c9742a215b8478d1
    6463e6d185a2a6b30646ffb31470a5815c27c0
    f161ade5d5ae17b73053de76bfe04c817f2dec
    260092294e586e987a0ba5fddba5269d688c24
    63581e1a03963773cc8cb71a78580bd8530e02
    d985e1d31be1bb1048786ef6707fd9de8ef4a8
    bd974b628c315c5a8090b80017f9548b026a91
    51bc2d29846441299cf57b397b02fc164c99b9
    4456845141eba23dce26482aa6d4196f4804de
    9a34f60187591c26ee46d60e49c136acd6c765
    32061f510e182e510b8124318437ef77255dae
69/
    0b012624178c6b95abe3a4b00cb7daa33ceaa6
    5416c6a57216acaa78746b4ae64498341b3956
    40aaa97951f08d67c80a56d66676076ca83234
    1701b0eefaea52b07d27547db4cbbcfb44b066
    a8f93033e0392468fdf3824047c65ab3b78c52
    3c8308af218a0ee1ac465cd088b393ddc25740
    855439d24ce48864abf985a771c61e2334a921
3c/
    e0937befd3de29bda3f5e58c617d9d87435b86
    50c5dcfeeda2efed282200a5c5cc8c5f7542f7
    3ee1417192979a2f6ef45c562917ce441d6a49
    b981010338fde5a8c448a44d17eb1a8a092a65
    5cd94cf2c353042f5219bdf41b18e19a443a48
    a4892fa31f026428eb5c6de756f1b714656b90
    6de1cfb2e7b8f4ae95100589c4eaa84fb99926
    775e1d9ef1e561c874007a3351c3272d1043af
    e3218d8c5d941ef8b109672ca00eff9cab06e4
    748d33e45bfcdc690ceee490cbb50b516cd2b3
56/
```

```
    2afba66fd669e54529a9bd0a6cb82fb7f005c2
    3d2e9d6068732e79249839e321e77fdb0630b1
    5e9d960f8604c487e063ad9ed3f6f63027f3b4
    c9d8f3862f3f3c3067d468f47a3b0385e69fed
    925b89933e83d0bee83e6ae17b3ff6c0bfd416
    5658678758a3ec6f3b938535d4f4401df4dce8
    d2975877f092ac62ad403803f6456858affcba
    4293c90b30cd03b2ba5c6c4af95ba1fdb84db6
51/
    83934bb755cb4c761c4608229b3e8372dc2c3e
    248cc1461bd61721407704f8d105a1dbaa9686
    8f5ac26024837c1e0d1ad3dbb69694af83412d
3d/
    dd7783b072c18f75da446a3810e1b258c60ff1
    f6c1ae0ef68de5f36b127090ce36875ab966c8
    6ee61d9949112d6a9dd4f15c692976a5fe11b5
58/
    81dc9284d815b83d377ddca00d0a20be52bcc0
    5d1bb501d62cc8f379300bd3901dda9bd7d585
    c023f6b4479c631f382e5062932793d2bee26b
    a76254c28e904419ef6f411ed528201ed01edb
67/
    aeadc353ce372565a86da03913c80f14a2fddf
    c4a4552f898b8ec1b9120731f9792951e3402a
    361df2e49d48dd56c91e291ba92553e9afe344
    70c0f38644bd0ef76f7d513d40d9aec6850ad9
    0db33eab5cf13dad703079023732127f9c1010
    d0c9e4a1da82da9a611a1d79939d67a715a0ce
    db4625829680298b2a5a9032a379d870a00700
    641c95d4f1cd60f9d9b0bef6a3f51fa7ccb7d0
0b/
    7a0247e85c9458624d748dd4f436af4bdc79df
    ba3a86a7e3d5b1acbbc0111288004fd1c1f23d
    1af50a3e85e81e151f51da7aff2b1871e2d208
    8f3203f6dba2c187c312d8b7fdb3bcbc8e4243
    cbe53ef59373c608e62ea285536f8b22b47ecb
    8e9e425642f70c27f9d64f520329abab9310e5
93/
    1d7c3fe294f889690263be26c6b75bc043bb2e
    54f9e3140999702ec8c140636c511d71c340b2
    29e7fa618cba430e77987d8d4482f1ca835645
    581b2fdef28209924c043aa805f54b9db89731
    dbd9f5597b8abcebdcb1b2d6755439d5158574
    6212eb03662f59341e2711db4575a9462b84a9
94/
    0a57e01affa86eb5577550cb70655d016065b6
    6f058df04a562f4b21d8523284cbb12bc71ccb
    1fdb9ec7a9699c1f4de8077eb681751446956e
    8bcc6094d13684a39abd17b9519b2b78ca2813
    a82fa6618270d3a16f521a0fcf710a15a8aebc
    c75e1a05b47922945c5233e90e9f936b108b66
    13f7cfcb40a0eed0af78347f7d60ed367c2738
    0a42aaf61ffd6757b69125c790efb88336b51a
    21fbaf8211b9d71e109d9481662fc15976ba2e
```

```
0e/
    2d6c7b1c76c8307ec6bc25a8c4f015c1f1ae58
    8e5e1608b911e789a3d346ebe48aa7cc54b79e
    cccd1ee062c5177d9ec468cbab1cc30a6d1f8d
    c58cf0aff4c6a207ca4652a08cf9df751c9ec0
    218a6f9f75ea2060a8b08d1f1a043fdad68df8
    b8915732be9e52d7ec90a7b727fe5e4b619e49
60/
    6aa2e8f4c211cebc6fae5830bd05c3f83bf38c
    c521d503a45024a4fe0c3076d0bdbb14cdaef3
    d75dd18effb6e35b216cdfa3e30b8cc5bd620b
    6cce9d898d64cd3e4f78792b89e4ee10575f00
34/
    3dd6e85384577a39aecb9ca8d1b852cbee3441
    35a6fafb4ab67eef13097095f11e86356005e2
    f7a8c21883a03852019a97b66c53b1470cdc79
    e3a9950cc557879af8d797f9382b18a870fb56
5a/
    8483ff60814e24998ad0079b0cb0c54b657a2d
    18b758fe0065416a92b9047dc9c392a3de2c4f
    de356b9c2f3e375bf598635627870f248c0cc3
    b091998de595197887a57588ac4e5ddb94360d
    a9ecbb80cf08255f7e678432313b10b0a5f5ce
    f984b4f4e2d643e03ccf5925d978d8cf2b33d4
    d21cc7c909836931023a6feac195338825a4ea
    cd7687d642f06de84b38f5842c41ae14d5f24a
5f/
    339fad456faa761801cde0859423f64ac1f4d0
    4229b2763603cf7350eb50560c0a5506ed54b4
    bdd6307c5a33e70b625c844f183fc45fbd0434
    b40a0474856e5e3fa1805c989c504c6c46a812
    ce6660c07620911bf44bbfa217288a0ad141b4
33/
    7a9ff63e1cdf2578dbccec0b8d162c330aea29
    0766ef4f3403e05a6ad8ec30f25fe05fdbc199
    c6d24cd85b55a9fb1b1e6ab784f471e2b135f0
    503b6620ca3b2618dce6c0072297979051c6fd
    6b52f1efddbcaeb6716583fc2f043699e278fa
    c613b749a49d6035c0e549389e92c3d68a83ad
05/
    d2971994d36b29c6532e0c99115c1906b8e275
9d/
    048b7c02d838198cb2323f95764c6fe7c6165c
    630f491d9a39644ae65564dac88eb51f0bbe78
    c68410337dcf4619ef66a49d87cea8233bc057
    02945fb311424daf968f79eebab8d784a749b2
    2337e4587d6eaa8bb4067b5ee8aebdfad32a35
    f625cee7d8e48f8be9131b59b76a3352ec866b
    4ead0eb036be85c7681c74ef933969de0a6ceb
    645290ce063d2e83fae5d8b6a9f9ce638a5fda
9c/
    170360b1329a773ac3d3cf326881ace4fb136c
    8eb08d981594784fabe9b2b9a2b7b8dd2e084a
02/
```

```
    cd9e8a5d88f1d83b8b17d8374e8c6ed1ccb906
    5041396b691807a40182ce255741c7e352b11f
    ba60827933d6623cdf6b1417762fee47c1ab6f
    cab328251af9bfa809981aaa44933c407e2cd7
    8dcfa0fc4b3a07307989c40389b2042ceafc03
    08fdf33b640cd9791359d74673bb90cfb87f96
    c62eb92a69a4783fcd593cf682469c1a9d044c
    bbf68e7ad3ce14f191af24260312e817e12df7
    8c2d99b57782ed3bb268ce522ede37c1704d98
a4/
    f8f897b80f2456f044f31de2b8211198b1b588
    0eeafcc914108ca79c5d83d6e81da1b29c6e80
    963aec6388c27c3beb064f0a730af200380aee
    db474379393191c4ca72a14f4bee3981946d3c
    f1efd2c37fc46d2e76150fd01a336442b23808
    0a7231b3003afe17c275c0ad9334335a020ba2
a3/
    7ab18995822ad6b3372d56366becdccf9a4c26
    ffb3e63671b61ba370b7c268b99af4e6d26094
    75a4fe13e0c742282833c227ef3f332d2ca060
    581fe656457648765d3b31d5301677bf6a8f28
    f7aa62af1ee2690e1e17ee41f3c368953625b8
    38c1588fdda63a3dc874c5cfb10b0d340025ab
b5/
    0eb3ee4cef2bd4fcbda62b5c59b00a2c3f4597
    a8909948dc19d89651d60df4a8113f114c9917
    ac1070294b478b7cc2ce677207ee08813bfa37
    a91a85464ff7728856371a49d914bca0776e7b
    2c9c6ea89fc6859fbf3e489072c1b3b0af77fc
    d33fd9fcb83cc419d94ddd362219d77a350d52
    46408968e42d5d06392e2f4564c593937782a3
    1bde91b2e5b4e557ed9b70fc113843cc3d49ae
b2/
    89cc741cf33fa44616783631d131c679cc898b
    06692a0a976d8336e3f5896eadf4765a33fb2c
    9a36c5c03c6106970aae4b0228ca0b3bea8443
    2f7abb93b9d7aeee50829b35746aaa3f9f5feb
    2dc1f46d43c3b680aa87257cc1fe7e102a79c1
    8c525de1a5214d0ccec13a6e82ea3532e36fbb
d9/
    49412e03b29d70592c7721fe747e5085c2e280
    c2e1851591e193028fe88626a49206e69c8f25
    2acc7bedfc5c7c05130986a256e610640582e5
    5fe44b34a936dc178c89d98ee9ef093cb0fccb
    3fbb5159de7059fc19f8defcf8d7bc9f9bcd07
    3ed902dcdbb64c84a65d7761279636ca989104
    f5883f42931526da2ad46c6301960c2e8b14f8
    7c3e395ed89825b2d6ec29abcbf82292bbebab
    6354d97c2195320d0acc1717a5876eafbea2af
    df95922f3e388ef62da386334549d7d07310ff
    c7057fc0fd0e384547cade6b31a616270f3d69
ac/
    489726caef968e7b8d82d44c171862e1af1182
    833db36081cc585ccd903ce8a2c5af536233ff
    d0a6c8f30310d74b2da3e7ec5e1afaa859ed8a
    5f74c5d1e48b433cd4e23a837ebe3c8f77cc38
    10353194f5f17b042c2076b7397b0c12bfe588
ad/
    f35838bd79c096f9d0bac41c8f9ecaa4e4e505
    b7d507c7fd87d534f5e998384a574b03a82efc
    36183898eddb11e33ccb7623c0291ccc0f091d
    2794077b0a0299700fd0e8a0336bd1d6e24677
    93efa62aa3a839bc10de15593f1327ad4b2843
    5178e76ff9245ca515fd826ab51907956f8591
    57a28ad90509df0f2336881fc216975ee18295
    a7a37b2e4043c2206bd3c2d4bc4dec969e8611
bb/
    06d6d75aec9e3bba47c5b2f911d6a23e858f04
    bb1b489f46100bbaf5771fad3353aaa8de5a80
    aa021454eea8f6a1f2d14539116fa6eada9a5d
    23effdf865b007756451f61fcbd7635f15b5d5
    025695d7a63e2e1c82d3a8a36905ed32f1fe1f
    35559069dc8b7c46973d9be937c00e0939a45c
    b8e59cc9afb1180944fccf6e0edff2e6c084e5
    3670f2f21ff1f475f339759dd9e282c2810f3c
    2cafa18011e7115773055338291c366f173d6f
    71e9b48e4a4c7714bd8c0d8bf4942d2e066291
d7/
    4557a48f678ddec39641aab2f350ad5de2987a
    e5b097acf9fac1e394863f0587555a63a59f49
    788d37e3f1e79cc799e8fddde2478866d761fc
    d02725f74fe6d784287cc0506ed4093279cbd4
    e75d58ac9786c7b091e4cec07ff36582c195fd
    09f522070951835a02e775bd80476f0be9d285
    364ba61eca930aa1c868abe3b322cceb995a6b
d0/
    955f9e608377940f0d548576964f2fcf3caf48
    04a944fe97977bed81242ee27d04e390491bcd
    bb1fe751677f0ee83fc6bb876ed72443fdcde7
    9a8f8a65a706842495e7bfda47070a782fac8f
    02be7de332e2505ad372083678c3fe2b097fe8
    75be53df792388d760c5442bd46a9ed5069718
be/
    4c2433212854dd0f5f8cce22b88f74226f4f87
    0585de099263f1629305c8238dde608968cf6d
    be24e6d3ac321523e0442d28b77b6e6df85970
b3/
    4e79ba6fb1fbf0305a5d01439a308ae4c6bf5a
    0e8cbf84f2a441ca87aef2ab1a0fed18caeddc
    2bfc74213d93d434f1f3a47cb5d7d0bf4863d3
    f679b67da7c997478bd9ee8546682106b8be62
    e293ea3a508dc54674349e845f9794118f548b
    e252ca57daf2690e6f67bed78653fcc4226173
df/
    3d9838b6faf2a36eb639649bf455b6789b52b5
    f5ace5ebf65681525cece57e23dd8a7ec1e483
    e455937c86b5b7cc83f5506ae0f7010bece1b1
    9a892f7bdc504a054e6c239fedb8b784e88407
    34eab4ba8c50fab7fe8183d4c400981cb5a62f
da/
    bc26441909d16591d4878b21b0c60b15045d0f
    897690d836d4ff57c6fed76f415d2dbf6e2314
    6f05789019d5c16672d4c7769f4bba55696621
    96455a07a0bad4cde5dc5626544325f82c722b
    a00890c687e599332fe76454ca182bc4cf6dcf
    5978f74944cbb3dfbef83f47fd0037b4129d39
    9857e986d89acac3ba05a6735dc08c249bde1a
    f1660f0d821143e388d37532a39ddfd2ca0347
    4be6d32111c26a2dd57533b23ef5cf81e0f0e7
    8de4ac01d21b6c4e99c25b82dd38eebdeef499
    fa08d15692d56b47225b8ec22a23016c00eee1
b4/
    6017fbb42e9840cbcc79b489c978f43b0e4cf7
    9d77ba44b24fe6d69f6bbe75139b3b5dc23075
    29cbd8469abc2f6a5b716d8963fb41f1c479b1
    bca9076e71ad33dd7cc4b261e5eb9be5b12573
a2/
    d4ba119382a1b662d71eef1aafa0c2902c1980
    c2684c2c0fe260149d324893507940a01d9497
    ca6be03c43054caaa3660998273ebf704345dd
    f2966e5496601787d138e9004fbb3d2ce9b64c
    5d9374ed5f978a128d4abab0e178ef1971c000
    a5ad12f0cc628ad411637187b2baf810e4bb75
a5/
    dc12bdd63163c86f87ce4b5430cdb16d73769d
    08ffa80bd715b47c190ed9d747dbc388fa5b19
bd/
    f2df21e9552e3e1681a3be3d24663c19844f6e
    9b072310a91e8d559116c7eda8d8704370f072
    191c4e14f389d6d0f799dfef9c5c0221a8c568
    ece56bc9d256e895acbd2b01d624e47a4729dd
    2b01e2b59b0d8798aab0df27ee4efbd4f87331
    00ee11c294860ea6b34707fc40d6cf42ab81f3
    06ec60367500e60b076a4364b89d87d02567e4
d1/
    eac334aca5a7fdfabc57030cd6b12dd3e70aca
    6e326024c05a59548619e13258acad781e0a6d
    81ba2ec2e55d274897315887b78fbdca757da8
    305c958aa2d1b846353469f78367d59e97b7f9
    b33e84c16a9e670c8b8efd5f0ed31f3c529d8c
    4d5bdfbeeab59d49a386a3c6fbb7ad95a7b82d
d6/
    5b2479ab27cc893fd79c33459170722d8ac29b
    a481608fb005ce358dc5467a802fc6fd21aca1
    c0c007aad871b9348fea57c9188d0ffd5f10d2
    4ebb9d45c0b74527cd503f53e3758d51200199
    d2615cfdd0b914d064cdf7eecd45761e4bcaf6
    9ca3145704364d2dcb49150e1ad0b4dd761425
bc/
    87e2de22082a23a43c3d3dec5dce09ca3e8000
    1abc96a1bb4ef7d7afe6fdd7e81f0a1e048844
    f704dbdf0e36abf690d3f0c5433d0c8d6cdcae
```

```
   a98a83038fc381cd17b2070f9a669a3bed5ebc
ae/
   f754e10b0c89be36aa16f533df006b2bb376dd
   e64e7498535c6239ab9b0bb5d45fa7d9330aa6
   e37ca22897e4e089b145226b28a26e82faefd6
   f2821b83f6ac1730d063d8ce939134cc2105a7
   353f966724751843f49798da82b2b2d72fcf32
d8/
   d3f414cca94e6988e04878a78916e6b042a48a
   d8137abc677e8d520d677a192b5a15c8d35a27
ab/
   34db74091c8a04ee9004ce9a786de3146ec917
   7fe4c74391eb3f013f5320de6fb3e7a3c24c22
   71d27ed2a0feea1b22b252d526338ed7d73f85
   4f92d013375a7e233205b6f34d8e63baa372cf
   b8770811f6d763433eaa87cf745ee720f1d7c7
   f209e60c7c4a9b1ae57452e36b383969848c2e
   01e615ddac7d0051bf5461cf2e24aaede9aa16
   94ee8d890a5be2b3f5b51b5c53720a3e5efdd2
e5/
   e3f34ed81453ce759c6ade8b2def733e9063e2
   89bb917e23823e25f9fff7e0849c4d6d4a62bc
   59cbb43c18392606d1212cfbde76339719a6a0
   f76777f838f189962eb40536f1521f4bc7710b
   85184e1fd45646efc6958a46b570922c2c78b7
e2/
   19d73849bbbfc556be108fac2ae619042bce1a
   9cf368991ccb083b67cda8133e4635defbfe53
   f7963868c85927b39d587ec7fda436a43c2ded
f4/
   c9ca432f48601b1f1b52b974020db224b8d21e
   bfe35b73b29aea80ca1f1d055da261f75e3d08
f3/
   ab45ed85893c1b87107243c12b4cf8ad0c611c
   30ef12a2c5ea0a4adbecbeea389741479d5eb4
   30d0b3b9c2f11b7dcbac4aab6bc7bc3eb5c7d2
   4b20276336ca0217cef1965c71060f25583827
   e8f3447dc206799a8e124000a81c443adc870f
   d54445a4a9c81f9b5abb73a14fd9da108d9e7f
   4bfa85c8029f514f81f829fc24020e78b3786e
eb/
   57ed1519f82adb79a3d2377e1f286df9d8ef6b
   e0cde06f9fed9fd39ce156d6b57c9e0e5eef5f
   89d652d767292ada81a1f9e9cbcaccf065ad03
   f7a36137f0c6648af4676003527c97836701a6
   4407ff716f67c1fecc81a593bb22d4935d35c2
   e4a96f589474f6f441858de2bb961c5e473c6d
   8e12b2dec992dc38c87510055d6ccb5f66c828
   2c1b46b6928363a1db20306c379b12668c5a47
   40c5f0c8526208d434d762855d23079dc68b36
   39cf63a0911f20d05d89078322446c39954f20
c7/
   c153f7d7f12d607379c5ed399ada68aa4baaa3
   c8bb6ff4f8ed84e466a66cac6b953b901626ea
```

```
    29d8a28439f79e0529af4fb7a61b8bd1d3af72
    6dd83ce73f5a099745f9df40f605cc361cebaa
    62cf2781d460288674314959c727e860aad067
c0/
    16ffdadf13abd48a43c7cd62be742da3f3fcd9
    4823c1590f7bdca9df840cfc17c2500b66d64e
    24272edc85bd9c928707e1011cc98f7c99803a
    e47d06d4ec1114f6630118bab4433dc1221279
    2252bcd13513b7a9e6496d5f3cf0af8c58cb82
ee/
    c1775ba5fcba678f014f8a977259675e9c1854
    8b43ce3b90a20326ec20062be04910a129ff0e
    d25018434d27ae750122c6e5c94212f5628384
    511ff20d73bb245fe7ae0c1fc31a41c33e7d29
    372e983fc606ec01464b46c8e91c84e4e33104
c9/
    08c977ac7a8b647ef5a8b745965fed4be3289a
    d134cc3cedae929e5bef2b5547f7e33dc10a52
    a423e80b1b88ac01c4ecca27e2d5461be4b8f1
fc/
    3a4359f22d707cf7629247dd669ed69ed9cc39
    59de2c66d2fbc1b5070ac84bf5938df2d52a14
    16c84437a8a34231c44d3f0a331459ddcb0f34
    430e825cf47481d06355b3bac03e2c3079a341
fd/
    9373937bf7de01f064bd8a20773bd460f49bbf
    713830d36cabc6a0fb4ab4e8cf426a84decdc6
    80d8c1129722b84771bd6a0f6ccfd57f5cf78e
    adeb597ca6450548e53c8de89d75a13029e74b
    c3665de7488864c811d01380124e48ed40377d
    9d88a8b017d6c1f2600b71812977e80d36d9bd
    00ce6e4cea506f3ab08e6412d2eb6443ef582c
    1a5aca3389bf6b8f25674de25723e2325a551b
    99805f5260a2b85f0ecb5c611bc4034aee1327
f2/
    cf635e2937ee9b123a1498c5c5f723a6e20084
    84bcafa6ab2e1c9ae51be54107836e68cfb0d3
    36910b0e16399845cbbcae181e3044e8fceb15
f5/
    c2b2b91de0a17983ee23c494fe6fef1f15d597
    cf84060dc25374520ba7937f780d6520c062de
    46eb1482f5c4ebc4731ec6671c5d35c5c4b992
    97b5cbc729be301c8c671751c124e0d328282c
    941266722642b6e8778340e59d3c02986e586d
    1190ac60354d90eb2aef4b04c484f8517275c2
    7eefe321c2aeb0ec7e346dc7f3c450c0420916
    a6f59b47c0ca4bea513f7695c1787f78646ded
    5d74879ecb78ff298ad8929f4b553dd66859d7
    ed5f6f6ec0eae90a9f48753622b2b5ee5d4a4f
e3/
    3cba5e44d4f7c62a479db7ea215f7a06ad6efa
    76eca4e5bb63ebe76132b3aa2064326589400a
cf/
    2b976f377c2656afb3d84add8d30b0fc280c03
    d7dc72ee7fe9300948133cfeb660f610b90e4e
    b8639e5602578cb562ee7197d207dbb539cb74
    fa27cb08285d1535e9812858dbad1551fc972f
    b78ce52d226d28835ca38872a51f0671dd0e93
    f676017373bfacb12b937e6bea7266965fc040
    0954e1a30546d781bf25781ec716ef92a77e32
ca/
    2e50fb3f54d519de230edc58dc2e02a4a553b3
    c81c76f41d89db151c003c4b7fd77ac55b14f7
    83d100d57350d2bc849e25abab2df4aaa0ce8e
    fb79fb3dcf43744393e2964056fe32c350bbc1
    539b40f6a9c59ad3937c5490426ca02221afca
    0fe442d9ca499466df9438df16eca405c5f102
    6e4c6afb4e1446bfb2a7bc5126f96e7eea5c6d
    f15f04a603a4d95a52fe0e004a57958054b332
    f05133ce673e4b385ddd897170b4c2d6b2648d
e4/
    a1588f544922442c9250171e2f1b5fd6861100
    63fa04ccafded9d99affe8fdb5b394945c1773
    7a7fe161c72decd5f292e99bb21c1537eae7e4
    99ccc54a5e46d65dd85e5a30b57d95ce107a5e
fe/
    61e8116b71e073351939ed7a499ee752398f1c
    555c339058375f9eab66a419088dd9e04923c9
    f52aa103ea369c96567b9af2a5a0ba14db5cb9
    581623d89d67a49eb43f3c3e88f3f450257707
    4771d4a1a0266e5c2cc7b0f784bb17d42334b3
    5ebe44f7f09bb9230ad27c37d379081d5947f3
    9061caff930ffcccca8522ff801d6d86cff43f
    393e41ff995fe1a60118be667255e9c9c9fbf4
c8/
    ba97e7de3483340c6a4378e75fe28b1b9b6fa8
    47e114b9ddb71332b9d66dfd124f93e88c4da3
    8cfbb2349c6401336bc5ba6623f51afd1eb59d
    31c5a75030b22e0937d050342c4ceef0a99cf6
    1d19d68b40df01cfe94fc513d86434fd2c1bd3
    49c4392a7280871fefcd5ab6a83298b872e0f6
    c9e1cc4703914ee5a6f02bbc4cdac94a024c62
    d0905c634ea015bd733d81825733e7989d8a21
fb/
    7de66d6bc3b47504619617383f905d5a36b28c
    eec342c06e60d8a8893acb30744b58027e6334
    8b871a459d4f43c112d157191c54f69110821f
    be1658881975edce0ba5423a31ea8208151fb7
    35778bb18b57c1d2919573617a6c3f59144006
ed/
    578aa2500d8917d5d3ed1249526b48ad7ee996
    86a552d1ca6baa0cfd48ec73a7a5c952d047c9
    6f3f343fc5718516f8b9da29b94a18b3b0e934
    a80935123cb5db7e18d7fb82fe5f71991d7af8
    e7dc7945c009b11c5ab5eb43d93607a2e909d8
    c19627dba6835339768ccbaf726db21d8ac212
    b38aa1a6c54dcb73e2f74b6bdfff337841d99f
    c0a71d02ce3b4d701eb052e323d5d502fecbee
    2507340aa399762706e4841c47eba85537bf34
c1/
    0e1f4ced6bcc799799b62666695998e095bbaf
    2969990d73388f61a6ab98fb4ee8f0f5cbc44f
    9aabb91ff4595bc7e20e9d6f80a16a9be5d42b
c6/
    b76d1f6fe80cb87c08b13cd9243c3a89b02f36
    eabddc1f1d303b6a581d180c8c7897e847e35b
    19d8bf8b31e78534746186d9a8a50f506d92eb
    50cf595904862a016bde7ed80249310dece92e
    fa38212fb559a9b51fe36b72892839efae63f5
    6ac354deb035405fe0e4040dac539d28570257
    5f2164b23d7c17a82d2ce4d1eec48128d1d1e3
    1f9184f8cb41ddeb3b2a95cf29bde92f98e5e8
ec/
    55b489b0f403c10438a421137496913ebf887f
    253c414474677d3a5977511cfe901bfb786740
    ff31675dc5a1cc8cd5593363356d16597266e2
    0b3a4fe6055b276d5515a4e81d60d921c6f381
    7f81e22772511d668e5ab92f625db33259e803
    7fb3b6c4856708dc6bc3b0c35fd8df73156029
4e/
    e358038c046e70310f57af3ca1c3b5fff7a515
    15675d8b5caa33255fe37271700f587bd26671
    e2d3a31b59e8b50f433ecdf0be9e496e8cc3b8
    9d946294246328c098129a8814e5b9dc48990b
    1c02975fed08bc72d003a1ec11b7979bab348a
20/
    0f3a4070a63b9faf88b08e3c9ab2bb733fdfab
    a17ed09272a09a5b3c0bfbd0e6c43f78db4c1e
    f07aaa2730477b580036214059291d7ad6d06f
18/
    6796c17b25c1e766112ef4d9f16bb2dea4b306
    7342dd3f44f5e64a4aebd70119778cdffb47ad
    8e13e4829591facb23ae0e2eda84b9807cb818
    670aa5e02d7360bfaaecebb878d4c763f5bca4
    b722178618d4912ef38d5a2e2c4fcba34d9be5
    72928e2789840c7b88bc9cd7cd0f3ecf96fddf
    7423e8b88198bcb60e6d059aab830e52dac4bd
    18fce90186134c70e8f7748365e04c88f34f1e
27/
    6aa79bb81356cdca73af0a5851b448707784a4
    c98b7c30c90e077eae29ae932d260ade0f10e1
    bcf9eb5bca0f0eda0c64166110884c37f08a3a
    e5a198a4b93fb98440a7036af19e54452c059f
    c8fa3d5b6999c77dad7aece312a5d6cf12ab48
    c69f0d1eaf3e223d599e91f969d52a821426fe
    0629fd8067bfc20ed4a0b39d9897791ffa93ab
    33c73f926ced3ce763b466baa4629fb20dba9a
4b/
    af1b9ee59c2def446c4ad5dcbc97d720d1c6e9
    0e838a3964ffe11eb69ac0bc3da9d232c767bc
    a5341cbd5582b90926d9860aee23ed4f5015b6
    0b0da6c2a62b2b1468c35ddd69f1bbb9b91aa8
    d072be9769748a852740d037d5c63021472c9d
    bc6a6afb5229048e499bb6983c6f03da7e81bb
    a3821bd1823a155b2bf70b27193e257436d2d9
pack/
    pack-a2f12eef7f8ce01ffaac742884a686ffde963289.idx
    pack-d9f8b523faef57d23cc8b45c8b387b43c0f70051.idx
    pack-d9f8b523faef57d23cc8b45c8b387b43c0f70051.pack
    pack-a2f12eef7f8ce01ffaac742884a686ffde963289.pack
11/
    d4adf771f3f90bb5f1cc11043599b48e955c22
    b0f959ad68143dfc19450c0bb1db98f61ac11a
    b8c900b99b443e5a814562bb3e75d0066d66ad
    23494dee81ecfe4646e4d68888e06339c62518
    ec695ff79627463a0282d25079527562de9e42
7d/
    b5dee35788cf050a576525f162a52397d5adf7
    36e64c467ca8d9cadc88ab03da71faf1aa8abb
    c3b10387d1c3d2da8b4e27e917ee2a85086e0c
    9969ea6977cdc2906fb0c927c7ee887741f16d
    f468d855f22f0a6d8e27d707d614afa35cecfa
    cada2f5c7937431b4df83e1fe849fe6d8d2bc6
    577d66b200877778e32fcfe737d152a874f8b4
29/
    1ab4ea5e915643b02bf6ee2e63b0b928f36ed7
    1857c25c83f91a151c1d7760e8e5e09c1ee238
    5dc928ba71fc00caa52708ac70097abe6dc3e4
    960ac48cbe5ecaf83795cf240ddc6ae82763db
    cbf91ef79b89971e51db9ddfc3720d8b4db82a
    be4309bf0f846223f737a09b30a16a2b256a07
    4b813b8ec67ef67c16da2c3d1097d2890ba75a
7c/
    c0b3b9d7675cab80504c3ea6393f2bec86dd0d
    0f0e5b78c0451711d7225481e1d3c9160e37fe
    69906646d667f82109e88b96316210e88fd02f
    0e99324854d8dd6d55250b028bcb8852332393
    a29f46bb1ffed7ebeb6f771caf4bdedb494403
    a97febef6a273333a66a35093d68999ee68047
16/
    89f5bf8d5f17b68ba7954f2e401895367d7130
    de903a44cbfdf2f4dc40ee581059155fa1a9b3
    8d07390dfc366102b8197e4b271e493bd94d11
    5dd31f53b6aed554ecb3d9c4cf8688ea5f5c39
    9b2f675aeb442ae3011d40c8218db3a10da6b3
    933bf8afedcbe3e9d4fcc04e5f7246228c56fc
    fd4427058819fb24559728d02cef4d9118b1c1
    d93a67b7b6feed66f2cc432f6250ca3ad34914
42/
    dade18c1ec2b825f756dad4aaa89f2d9e6ce21
    cc2e863c1953482fd9c4f89640c67839939be5
    17ec47b71e2ad5b6a7427c7a127ea18676c6cd
    66b5ee92a24b5e0ef65689a1b94a98bb4a9b56
    8db41c9b5ed04f4a388081d37b5f780d0e4fbb
    c1a95006db1697452b2423500ae82a245d7d36
    e01f4ce78e5b949649688465243cec5b4f22ec
    5ac456750652b8cfb2e02c12db3e0bdafe7c2f
    4f2144855b12ce20f2c246ebd39f2430961821
    9606f5d1675eed462da30f80640d37b945bb68
89/
    f9b07511c8fee74686d9cc434bf66345a46d6d
    210e0c79dcc4f1bb541db730430eeb1df61600
    09f8454e94752d188ed13cf36c35f93fc6c3f2
    e1868047225bbcdfe04bdc4bea3281bf91bc20
    0ae8465c5b0ad2a5f99464fe5f5c0be49809f1
45/
    5516a3ad3e0e8eac8cd53955fe21d230f846c2
    fe9495f1e7f058b488ceaddbc40be3802855e5
    95960b5bfff671449235d51a0b9312e7d6c5d1
    30ca6a45bee0d4f1c9ceae437819dc0b95191e
    9ca30964b32194913779c369a0bc67dd207998
    223eccc10ed35a7cade624cba9878690b88661
    47fc522b690ba2697843edd044f2039a4123a9
1f/
    e3d225acb9bf37acffafc2198dc96c7c7fd313
    c5de0462cd9a09472cece4087cafe699da4fa7
    479713a94495ca72ceeef4806d1c07223284fe
    ac29b6b913afce18211b17c149c3c7f8457b9b
    2877bb2bd520253502b1c05bb811bb0d7ef64c
    5630aab8426bd3e57e5207f3cf58e6598472c2
    ec30e737644d12cbc097fe40d063f83dafd811
73/
    4e7578329d24aa3e462eec7784d07c8090cf16
    fa94ecf82e0ddf691fa13ed16c4fd892f38073
    f58d7740813264d20047ffe918c82e1acc84eb
    9acaee2bcc07ed84a813fb2a344ae52ce6e557
    95cb6a620b4d6d4cf5f026d85639cebb137a9d
    8cb4ce90afe2278241d9786d5a3dfe8b155ae6
87/
    2af332d6f838c224a15c64e7e715e4b9e0e433
    d9f972edde20d1f8e391b8010703242a8de977
    2dc07f04044bfb237a26e966ce619f7de007bb
    5e4fe56af351716a38ae542cb2967263c01ec8
    65b907d70c4a530bc90dc88f24b3df73473b01
80/
    b7efd5e8a823e8fee5e6d5cf6471653bc3867c
    f93e3d7025a92afdd1bc9ecc0d3ae67a805449
74/
    01cf5d3a372da67d241dafe83ba756e015eafa
    7a69067e3b1a5762536cff972b773ec3cf573b
    2c539622ce30c838b2696eab40ecaeabed6c5b
    dfa25e7b7764210d9c60076fe9dbb42cf13a02
    c40d7bfd5fc63826864345837f1ce66ac613c7
    e8afca3e1562722b0c685d4c7f5a444bfda304
    fe95b3eb5b571839eab42201bee1321600a998
    4497fa3aafe9e96b549a8e8470aae5f68753c8
    4bd7ef58b4870406fcef8cb3b3667548a0ccea
    41c476ef218011601d86b0bbec4d28f2abbf3e
1a/
    c76bea7648dc73f9da1eb4b484b0f56a808e38
    9426fb7741d5d59d4974baff6c5e891517b621
    188c35cb6a82cfb7dfb6d8a813fed35bed0cc4
28/
    175b1f750445011d74e14027745a4b23556a29
    2d3c9f68adbc515b73af45cf4e093d5a7e1b5e
    4b832e2d7f06e3cdadb35429062c26a1a78c0d
17/
    409f2ee8df322a5ac115d1d0ff0c2d2aa11c4e
    0263d9fd19caafdab1440e860236b9e685a691
    1d5ca3eb2608642925f88ff45490ea650e884f
    59b18661b272f6a2df1068038c2285d972b1a7
    17ee22cdf77849e2e273566c877f95311e691b
    6cb996408e6681a88722783919efc0e9dafb29
    769e9154bd9cc3f3c00dc10718e4377828cb5e
7b/
    6f6a324bad46c7d78fe6ab4ad9630ba674f0a6
    0923baa876b60dc0231de92882a2e14705f0ea
    be97e6665356327814e2b797ffcc5724974a46
    722d58db0f35c3f6621d02876cefc74e64384a
    c00d797bfc81c8373178304abbf89a2b05cd43
    a35eef8270c34f183090bfa189358565526899
    64501b8bc962c535bb7c2d9f38fbdde1bb2370
8f/
    9c661769ea87d650dcfe542bf6cd8b67dc8e71
    88c48feed6e3523bc28fbbf6c495d8c235309a
8a/
    59eea28e1b8286fe3e4ab58d3ac2909f00025a
    b78b5fe9635b4f04c2a78db733b73deea1aabd
    ca526a0f1b61a507ede4baa6be0f4aba0840ac
    c717a36f926e82b295fa60a8a9259f5dfcf79f
    c3059ba3c246b9a5a6fb8d14936bb07777191e
7e/
    8a69ee38d6604f7e07ce902c208b2b1eb9fb75
    1eb37c69e7fd0a244a69e5418c8899dd077e8c
    6bd5c3e5bc012322fc1e927b3df10efac7ed95
    688737d490be3643d705bc16b5a77f7bd567b7
    6aa726450081f6c59ba24e5e89844cab40f2ef
    061f5b39081f39e9f4fa2a0e88aec0e0a3da79
    723db93e64975208a883f42727ac40c077104a
    f47176e2782518043e25b8b305aa94bc30edd5
    3545a78edad14656d6731a202b82db3e631654
10/
    c176790b622465538788d73a9e3afee99b3875
    ff67ff4d2bca253a91e4e6461ad096b41da03a
    67292b69f348d27fa25fd4a88f3e2329f9cd10
    fc0d7e9f398dd550a42c6b8c0637684882ee60
    9c137b68c60b3640e6142a8597bc875ea98cd7
    82b3da420a2f509dc67f4b22c6c5000bc5e62b
19/
    a169fc30183db91f931ad6ad04fbc0e16559b3
    2066d5442541237c99fe0fed5b61b9b855bfd2
    5b119a1858833a2117ec72dc2f26e9b1458232
    e4aa97cc138e4bd39bebf6c49ff1955cb00437
    6a5f8e12fa0f60dd002b50914c2c7be45f09c8
    4564e761ddae165b39ef6598877e2e3820af0a
4c/
    11c05a7f114d8ce81e45b19b62e11031483d21
    6b83f946c6fb73f1ba62315efea93533e67b94
    379aa6f69ff56c8f19612002c6e3e939ea6012
    25647930c6557d10e8a3ee92b68cfe3a07f7d7
    6ec97ec6961bcf184b6e0b2437b9924db0b9de
26/
    0c8b43f299a9b3088901afbf6df64b5488503a
    10459228fe87836fd9311d45b3f77b676c0f11
    251a51d2f819708b1383d2e6dd59ffc15bac7c
    4d564dbda676b52f446c0d25433a15939a78a3
    cc05633fe6c86b6873936d1358b48ec100b6db
    b723c1fd3e25740e0268b8c9b50905c58c3d4a
21/
    44d439e0f15e77ed8193487db6eddbe8f5dfb6
    b0c5d1a82d1b408546ad1e01f1e5f7d52f29f7
    99cc7b7f004009493d032720c36d6568f9d89e
    6f9f9f22ea6dbcda419049916eb081fcbf58b9
    b4590b3dc9b58902b0d47164b9023e54a85ef8
    5811184c44aa0ec6e7adbe725a42bae4d04bbc
    b2aefdead012215b6ed3187157994c6b3c3650
4d/
    8f89842e53cd88bd3562b10f70f247c96fdd53
    a07c2e03b5439a9ca085ef5fac0fc8577e9e87
    7f3ec62980d837a7bf57db3ad220e0a74b70c5
    d8645c978fb9efb9d92c6258e6ea8de4afae9f
    242b5409dc6a49af38896216969917fdbd0574
75/
    b3631c3879294549f1f27418859aefb63925a7
    ce2dc9057a20a957abe2fbd4ef094dc4196684
    8e97d66380d954b2ae5d4a049b56b93daae6d7
    6513257e5faf21b49ccadc9ad5aa687a02848a
    cfd2eb59c80cf029e8a68bba57c2e3374ae0cc
    7d9a1057d5244e37208980197785e0ad20a4b8
81/
    342afa447746dbb8f060da2d454c0175f12e30
    617b3903e5c4650be8cc082b277202548ceb85
    0d3481800e5044a21cb290b0d9bdf514d2a809
    5daaa9217ea38ef80cfae36137461d2cb381f6
    3a0c4e2a7e4851450554f32af9cb32db31f58f
    b1082905338a74b72b9de432ece50a456687bc
    50d93f33bdbcc6fb05eca6c703259ab36ff40f
86/
    68b3b0ec1deec2aeb7ff6bd94265d6705e05bf
    57cf735927f49565e1580a9b6173841a8139c9
    49b74095a880d8acb924d008da1ecfdb211efa
    964dee474baa41927e32024107d2d6eb640bd6
    63097b447cdd80c52e2b2abde33a4736ddb9c2
    070f10c14b14dbfac004d11ba3234d36b70276
72/
    ca84040b626183e3328679db600c13472021be
    c88544e3e0e79debda0f7bbc4ec0cb88266f04
    3ef35150d326e8dacc178dfb60386b3a1b4bf3
    b2e45cbcfb2ec6eaed30c04198920195239780
    bd6f25a554b303d0bf5028145cf3a5c71b3e06
    71f1d66b02095228cc24c43b607bd0da7f6a3e
    aa5bfd4b60d8e6ef6ed0cf2ae4f763d12195cc
    c40a7099772399e65585b9a68c5de0a17657f2
    67effed2413ba315d0a1af8490ec677c227662
    81dd47120788f999a3c76ef18fb72f76a28ab8
    a595f337f6206354fc778bea96ca7a39a9efef
    22982537c7f9bb01b5fa8e50bcada3deb6f3fb
44/
    53519ad0202281cfa53b3ca2a0282a9b0a1799
    9a2073ca39ea293acd79fe86e2377012444dff
    ab70fa72050a731748a457014918ee6dff81af
    529b2123642a723da1f8ceb5860d7ebf0fe831
    9c655be65a948f7b2476302a46c35d9e7605ac
2a/
    153d1a8824ff466af8b9c316ef8fc044b93eeb
    965f595ff0756002e2a2c79da551fa8c8fff25
    f3fcbd490cf9615222d8ef1b85bdb6fb51da7f
    3e7d298f393ed8532e4f11913635efc94cb329
    cb345146f5a1b4583fa1ca43623985bd6ea914
    aa4d546da08abff08e9d75659a5e394b229e6b
2f/
    165532e5ac7c4e30b5ecad0573b43c86ecf57e
    53bdda09e92da38e31cac1a6d415f4670137f7
    f13866054ec6fe15990dbc38d66d124ca8e44a
    6e6641cea72163b9047153adaf42d4c33823ed
    7f8cbad05d3955be8fbe68ac8ba6c13ef974e6
    0fa04f35f0cbab0ac7a2c3eb9813d9706b35a1
    efd7b6bbf4515165ee02c43921f014d94acac7
43/
    11f6b7ffea6f3dd7309ae5a0bfd402218ad678
    f6e144f677a113b5362dcbdfb75db4f41c2b2f
    402d18f0187c1438973edb730d0e161fdc2137
    2726ee0ef577edd2a467953ea473045003f811
    29db09ee37d9eceeadab9164b308ec65c3bf7f
    66e217356ea08e17bcb60a70d5fb475e8c325d
88/
    b3cd019c62c10177ef401df476381e585a81d8
    a17574d41c5107785d0b1bdd9d59ea704bcb59
    2e36f5c1de19a8200000c216cf80119b37c96d
    cdc237878d652df54ae53c619e4d8a9510c41b
    7dc14e796cad0257e5ccfd51ed3a21b7908821
    bc10ac18a6af79f962fec16091d3494adc9e66
    fcb9295164f4e18827ef61fff6723e94ef7381
    e77e157279ac398c9bedcfcb1400996518492d
    6421437557e5d898f5e608ea7e9f23662f01bb
9f/
    e19e5c7bf67a5f9d0053f351dd760dda02b690
    73ca7105ff0bf11d74dd16ffb0653059466f70
    c66b12d16f30ce701597df8927f44132f61cbf
    1c7aa31e20a7d0ef2e6877ea325c068d50e406
    6eb98e8f0ef41d6fab05af6e9c24c5ef8a04b8
6b/
```

```
    8575de2949cd0519ee5f26b6eb00df417e2113
    5837ec6deaebe72027ba5689a94fb11d3a7c7b
    a5c51504bdf6c869a10ccf3f776009869bcdac
    8460bd92c361bdddbb3fc45edf482db4e8929a
    acca092fcd894cd52ae92db93d045bbdc0f6aa
    b325d0788e0ef98b9f3300533c1da2d8c05aa3
    9e601c56c1b0624ec329611f0434d654eb4a52
    dec63d6867928bf73a7e513f60cee8f49ca050
    a2e04f350792e2c0021cf7ba7f40b25dc6cd51
    7d2214ac885841284c145112b1e0c8f30f905f
07/
    e1073be4f84efceb97d7ffcc7a7a32b3d219c9
38/
    988739d6406aeb5e3be903c0ea6fb82752f328
    58d5ae2d5fa678b0da245b2807ee9c20fa44b3
    3101cdb38706c305449674044e9288b92b7d75
    cf869dc4054ec0ba93b209a2b2166565b769ef
    696a1fb3419dd810004d5aec9654e5224042ed
    50ddaf412022ac00ffa515518962a8a4c4de5e
    a0da1534a5ddc735c1a56461edfd0f3f790135
    60c3ff1e0095c27fcfa36501c8c3fcbb1f4674
00/
    514494012c4c7dbf4a0204f9af075e60d6bfb7
    341531698d1a6ff59b9b7e8410a2b53a17ab7e
    319c6d762fcda83601a88f2cc54aa489d85526
    376349e69ad8b9dbf401cddc34055951e4b02e
    8f06a79bf598b149bdccb73e572d13331a1631
6e/
    1cbc1762aeeb44a787aa6fc5ac9ee5ccb63547
    ade3709094417fd871014debed6cbfe7f6c730
    6490105bbb251f055fb6650ae0c5452129a3b1
9a/
    f81d9821c3148d099a2f66fbc2b0ffb880c984
    8463fc9aa49093b5762acfa5b22506dadcdf72
    fabcd52cb5224fa4780dac4fd6fbad656c71b8
    b2bb48656520a95ec9ac87d090f2e741f0e544
    80781cf4e2c7e01a4658938140390145d5e097
    4044adaf876f57befa8cf37c5c23f8840a99f4
    3d25a71c75c975291cf987001ecd6882d6417d
    89a838b9a5cb264e9ae9d269fbedca6e2d6333
    37422c4c43b8ef85996d8b4c4666847edef3f4
    295666341ab1bd216a54082b7409e9c4b4e813
    ca5a030545d3ba26fa96fbfd7cae1c31dcdd15
36/
    fdb579ccc488f5c3e0ef1836b3e423b8575f60
    f947e51c8a5436b636097a3e9e2626f66bcabd
    a66a623c4562e5af0a2e49a9247d3961a894c7
    6ac89d15f3b13f4edd1e2129ed197f82fde36a
    a1f75b42e9b8d1f8469eaf24be356ae13fc410
    607eda2ec5b8bdc4fe87e256cc8f3b1a79f707
    c9252c647e67bc7353c523152568b993c1331f
    286df379e28ea997bea3ee1fd62cadebebbba9
5c/
    185310982d06956b10c7970a8b089893939d26
```

```
  d9731da320ac5e355207d1a8ed2e0a058dd08b
  bb341efa2c001a22b3234a32fa849193f64f92
  01d412ae1bbdc88454684f132360ea1a1558aa
  4722c1c9e948d2267b55ee903105919168c2b0
09/
  4d2dc226dde3122f09e4de5de0ef05599978bd
  7ca9bf10efdfe1da77573ec66cb1fd26a7c8df
  eff405ec194ee2884f203cb48c5df54ff0b9c7
  a0d326b983b59b58f84b00e55fbe6909a23793
5d/
  cfe87a8c2d6a95ae2210a3f6ef75ec8ae616c0
  15bc257bed61e8ac187955e94670c4cd8588f9
  5b927fd8728592fa8eb11891b1a6b1379c4199
  9a42bcb531a2b56b6cd93351335884b1e9b456
  f4493b97206fc8ae986dbe4d76e2f821e60904
  693627d22d2a3b27caa7ab6266754b454f8256
  435997e9ec0492a125da9aa7b4dfe6261619f5
  6292108b1dc815b61be01972315643ffb03491
  03ddb3adbf8636a3f6b78b0f44cd6bd46708de
  b5d7f507c1d150e6b36f236df7ee61c0f65581
31/
  188df448ffacb381ee86253c84b4e25502afda
  88895c991accb92aa18d67fd9969ee224ba8d4
  c5a7abe75e2548a85b93c1c376fb733941dc68
  5fb9c8902c5e3f4dd8419ccdf7d85c6718096e
  3c889496d90cef94d5537c122e5c5e898e3bb4
  899f7ab1d62098a861c82992c980c1e2b1b58f
  0a38f9f7924c8363ebbfa5c45dfd531083ae5d
info/
91/
  ca551f97b4576c680711e826a1855fb944c872
  cd0db31c14e30d4c1e2e9f36382b7a5e022870
  4f025bc185d28231a8537e12978b0de4c4e109
  ea630e10f893bf5d6b17fcd9a1fedcecee6f02
  df077961b6310b8e1c708b74003d5343bff6a8
  7fa065b3c7feccdef5bc666a5109c855217260
  886f8d58b35e37e4d7e5d82708f5a06807b886
  abb11fdf507883caeeb2d2958e1c65fb6cbdc1
  368dda78aad590837aa12023dee67e224709ba
  e935b5457bb77f0aa5c3a32de7086d509d30ff
65/
  fdf56342e8b5b8e181914881025231684e1871
  01e6429cf24ca199316e061fdc9f34470284a8
  ab6593fc5734c7c6dc94c852cc86deea77158d
  c3cd99cc7433f271a5b9387abdd1ddb949d1a6
  e5bf7810b8e07f57056346cf11c674265e5b96
  c043c87eff27e9405316fdbc0c695f2b347441
  2cbb7f2e351f21aa600a9f6451f39211f84265
62/
  76a76df83307a28e8de973e306d42119a1fb2c
  6254c321fb31033c54fed7ff57a0df5eaaa608
  066318b74dcc5c32bcd24b9493fb34d1ce52d7
  ff2c2fd52d87c347a32ecd00c9a86b407ee614
96/
```

```
    6ebc0e37d6104a8e0e1fefe9dc526f39409ce2
    844d933745d81d1e29e33c75f2ed24e518c999
    22332bc96b1da5d6c8877c3d305d9dd18d66f7
    8c9da25c3bc6b3905685014cbed3a8934f1d8b
    c0292043788c9b32349c182584a7fbde6f1ba3
    2173c8d0a6906b59f2910c9cae759010534786
    f824955bf098d86e54cd8bce3bf0015f976ec2
    d1b2460670e20ac92a5ade7a74b7ab1cba71d8
    45d46d2535f9dfcc385cc99f5ce71ba7125d5b
    e7597ad3ad1d75011bf36f83915a2590f43bdf
3a/
    aa3c898706c3e854a99f3d41fcf82e68a25e11
    614b798e2db323ee985985bcf5551ebf4354ed
    1f9c593a86851be15bc08136bda907ab856ba0
54/
    02f120b811e3c4302c4d6473a2139d3f140c86
    2d83f66818c0be9a22b26b85673b33eb29c2de
    e2ade153e7460b242505af19b2383acf5f1a54
    247a78a654187206cd17a403913c6257ffcc7d
    0e7a4dc79d02a820e291b57c43335d5aa25a41
    b96b19154ccaa138af6bc0a4ac2b8f763017ce
98/
    6b0f6dcc8d1424534f48267bcf8d24ee7be281
    ba0304f5fb8388c12afb035b4a46c7dc55d6dd
    b6a7fdb31bf5d00f368275e77f0af490e298a5
    169c720937924cf5aa527960eaa3e14df76642
53/
    80fbcaea42480b3fca9ca3fe4fc470a905345f
3f/
    1dcb3faae56ef26e36c808536c52322f8f33ea
    c1446fbbee5d333f98a54aab8feb8d3a369624
    bee89e29068c17997af7cad5e069771600f3d1
    5f9a0d996e2897d5755d55d41232568dfe8295
    4d300cef077e698989245562375a9444d983fa
    0594054498c222bdce9d7db2822dfffaaaa1c2
    018c0872f7fb5b08fe565d4f6d51c71b63420c
    acc2e3a67be6aff35c826b05661365e7dc02c3
    da98727b690bfd29a6637572b04a28f613b901
    9f896e632e929a63e9724ab80ecdfc9761b795
    10701f6b28c72b62c9904fec37b96bdd199dcc
    06c8b0bddfbc4090d776d8446b015fdf1b37f7
30/
    d70eaa6da50ca6b89aebc8cb64602e5040a6ee
    a528e668f8e8bcbde9b466c95a2a34bffbef8f
    51209e6d75858a9ab7f3ea655fc20b79ceb6d6
    77c3d5dd40683e267e5557c2998fb3a9b6dc69
    446ceb3f0235721e435f5fbd53f2e306f078cd
    26e328dd2cde1b77007530106c248129f9eaa3
    b7c2338533ec805cbfd82875c8ef311175e098
    c441dc28ee327076a850b1d3c88a9a2c8f04f0
    9a5c34c429698c8dac53c2d1cb763e4cade3ab
    233fc7ad2c07c42e7c2d384312f1f4373155f6
5e/
    fd0a3416041e3afdf32a2d346db01d99e8f7d9
    a609ccedf18eb4ab70f8fc6990448eb6407237
    36df6bcf447d5be357f6d20f6681f95737e12c
    bf5957b46598f5d6a922edcf1c0bc162af4bab
    082684d778bb013215cb7ceb948d5417b297de
    ce05649e7268a75c82de6ced552619ffc093ab
    1f7ef7832d065df76ddde7713906c7a44de730
    3e198233698f2b007489dd299cecb87d971067
    29502cddfa9a9887a93399ab4193fb75dfe605
    356b14ff54f840dc71cc64c574fff0759ab5bb
    7af9fe521bd529dd2c1878b0a6e9ea7c57752d
5b/
    836c4df37123eac4f559ecb54692daa19f775a
    5ecec89fa63df274de10facecf3538781e644b
    d138bd7e4a501ddc2a58456fd5a2e3c0b94d14
    84025e6b6a7b997a23870ef66636f7c5418b6f
    4d15d9c58b8a56be542003fa90852d99b9a1f8
    be430c8f7801a51218a83590698af240ac3973
    3c6021c004dd26d0b5da999c43dfacb605b581
37/
    d55f89e2e0adf09fb3e81963c728ac29edaece
    b0e6531f1544e1ba9b5895c48939fc97441ce7
    c86cbb044609e188d629593502a99d4e0ebff0
    a6b613d935deda090ba549ced417eadfe5e198
    58d89e1b84b31fb879b01a124166a5b4dced1d
    f4dcf713e4ad9bf0a9d555166598d7f7fcf505
    827291fb5a76d7ef9a7a3a695710b2074ca09a
    e3907e45a3b357a51bb72ff61c42a1d4c1d877
    ca46216a59fd6e759b145e65d5132242a88c18
08/
    87c3021cbf797b5eab1e94e6b3ef184c274661
    8fdf35ee9454a71b4eff8b471580110def021f
    5272c1a2274471d6cdc9022b8a273966086bcf
    c99174055278bae45037895f87782bb924bb2e
    df6d7af42f35a8300fff7dd8b9fb00adbd9f2b
    42f0d680fdf20823b4481f4df85c068046e031
    f87ce1f39107d7f110f213d4930530e71c8289
6d/
    c5bf8398ffabc6fe4030bce93610f38cb46139
    0ff6ed4e2aa6cb803b458d5fdb09c362bc9e17
    adf923d71b29a07c93331c7913baccd8c01947
    e5c71d3f5bd42fc876423e064ba7ca2707ef2b
    bb2010d6e577e359c6e20e6f9962900acdaaba
    6aaa071512c23a3abb68d7d0bbb58305ee5f49
    17cec84e79be9681623fb2cfbe31fe78376223
    ef56b4a75f67000ed8181ae2d2c40eefb645fb
    cae9fba8b508319d0c840291e20b79ccdf4653
01/
    b39951cc105ba0dc612507d53934b4b855160d
    dd79079b04b6743295ef224592b49e6d9d2cb8
    0db7bd73344c32483ae7344938ac3708ac1d4b
    7ae57265113be696253278fa38e288e4cef015
    c6cafbe53f1fcb12f7b382b2b35e2fd2c69933
    7c897b86a3fb68b7a8a5b6626458733339d0f7
    cb9083498813e80572722735a78152a0f216fd
    5302f3acab80d61549f3e9af3bee1804aad2a4
    8f0d6ac863f2e4a27636c721669061887ae554
    ecf0400b81f1c90d3d675f1397722eb9fdb2e4
06/
    22a407f1796fae4d3a1b4f0b2c2b3461b00265
    6640118235ec4a353a71b3348dee1c2cde3175
    a829bd7d34d6687e138aa9e169ea2ac0186595
    8ebadfa42030edfd2e2f49047eb96d2db188ea
    985585eae3625771dc2abbb0c04e8048a67401
    addc0ddce8d1fd1df15b26f8b45221a44737b6
    4811ad11bb07b2b7bc8e30ec6c03f21997d6b2
6c/
    907ef61e8a8a0f035c3d56fb668b1d58f635f8
    a2332ae16a575a850fe97e5bc1e42d33b7b2f2
    c3bce448297a4fd3dd56ebdb8adb3138d7a193
    ad6b7b509ce8eda2a8e177236790c317a348c0
    54cd037aabcd5d5c8bc7223927c65ead1256b0
    f0aa3c8caa24d69810cb9734b50697d938dad5
    9ec6342a9e6e61a4e1797be836ec535efe7ed7
    9681e9a17a9e7dbf9f20579d389ebc03a18cd8
    b9cc7b3bc751fbb5a54ba06eaaf953bf14ed8d
    624482748f87b6d38aa020846c897f5f44ffb0
39/
    c84aae5d8e1f4701b0b04fb9fcb8d4ca219de4
    cf150a1dcea8e5738dcb66f26af49e186bdf84
    6b55fb80c66c22a4294a2939f5ae25001d260c
    06f5de4fbae2e8362bebabfd47a142db975232
    487f4098d7c2068b67d7d3dd85b61848974a23
    f6baeedfb8ec129e0076cc3eb94dd5bef92ed0
    a24b04888e79df51e2237577b303a2f901be63
    8386a5b9f61c13be314e256e671a37d28e3623
    e5e925211dcbd1aa93691acd949681c4f20d62
    db84262d8647010774faa696334516fcede11b
    8b7a03e74169f4dd37c4c97d38e5eb19509a3e
    a5388948ef12b69b65fbfa89a84c6ef4a4bfd6
    4a67cbe116ee7d5b82932854445df49de17784
99/
    3e86e42407327d10b6247f07794475a0a3800c
    0ead480218fdc7ca01ed6d146e47205987b72e
    c79f15dd8ca10939680db49fe3d3cb9367ab44
    8cb87dab758332ecc17f8acddbd0378beef160
    4392ab9f8f175e075df9e40273e75613ce2a55
    4668219dd4def6404e0afd3f538b29a0e50f8b
    4bb7493fd92865e6ab87c277ba5741b44c31a9
    6f95e2ddb4b2ed6dc7e1ba6db3dca7f4b64df6
    0dcf3bb254c802c417547a3391ce0432bc02ab
    f118e20103174993b865cfb43ac6b6e00296a4
52/
    154f0be32cc2bdbf98af131d477900667d0abd
    d424bdf0e2ec9ff5c77804d380a31b2fba4d89
    f0d930dd1d59e5e5fab840cd99013c067b17d2
    1abd7c2ca633f90a5ba13a8060c5c3d0c32205
55/
    939ce06674d63e1deb1b54d76ce587edd17848
    001c2ae0b9229233ee867ea8f5eae15874df22
    2bae17798696c1c334d7f10fc52898000e5ab8
97/
    33686ddb36b826ead4f4666d42311397fa6fec
    7bc4caa75c1e76156fa97e2841a01332f6fa47
    69daf8d19989b62567082729edbb34d572c99b
    ff3b1bce2d2671bfe7cbcaf3323209bac7d654
    d2a94445770e195b9fc73e904b920d5ff04104
    aef1f1ac237e6ef97b1a1d026818d7b8ab9be9
    6daac014a3845427d604e12bd16da32a08530f
63/
    29039ce466e0adb786324c7b98140271fe2fe9
    d46963f49680add13e9c14c9534cb83689e2c5
0f/
    79398ed12b3ddd875b503f6f639d8a84586315
    c1edd59cf0fa832eb99c22b5b2a4248a26b24e
    fbcdd2c3e21b68566c88a3f05239447489df84
    84e4befe550d4386d24264648abf1323e682ff
    1d688e1797bb506139def5c6833afae8a62bf3
0a/
    4ec5be79b7fe6b03679fd5c5a424289a26054d
    6b3fa2104a0f7216dd008ea2361dd4dde441d9
    314f810b8e36d0d32e9bf2cfa58529a0a47b2e
    d14031ca50c2c348dc0daa8fe7b38af532c0f5
    6bd75362c926ae335086028cf1e24694e5f1e7
    20c80f882066e0e1323b0c7f61e22913c32e35
    b00cf43599e13e7694bea9946f379ca7d0dc72
    cabc6a0f1af1a3d3742f859b4f4ae93ff582cf
64/
    db819c3bbefa284510f566770a7084aede09c4
    129a79bcb21f84b4575e54fa569cdf47da3054
    a3afcd7a1b54d2c072e4b8d2e56a9d7593506e
    04d4dc1bbdd6a843b64aa23be8c976fa5e25bc
    cc4af50e8b5473735345010f610f660b8718c7
    bb880d2fb52931898c838fa7716eac4fb548fe
    7b9bc6ed2946ef57f87227b8dcd638199ae0c7
    1dc9864657a26b4cf7bd544cf15854f0a5523e
    ef3d8936de2d558966069ad66744e2123d4b72
90/
    d609b226595a56661e9a979fc6c623c798a208
    a6465f9682c886363eea5327dac64bf623a6ff
    cfb8f4999fb49b94ac71256aa885bd559cb1f7
    1be43d48b2d05d5be333f13941dc32d1a111c3
bf/
    6db104a2c4fd4f3dc699e85f2b262c3d31e9a0
    54ab237e410603061b8cec8fd195912d3cfb08
d3/
    f1adbef3ea01ca0ae74755cb35f517aaa0381c
    1640d120519fdc40c6011fdf0e6518888ba566
    0568d944e7a03a36175d77c0c92352b0b6f0b7
    38368007e38befe0fc44ef0670a5e7045ee636
    9aa6d0669a55b0d33e7d4bfe6292ffbf358024
    293585671a3b1c47ed268f553c2893b0f10fb2
d4/
```

```
    0686526e7178515b9693320509f783a6d0d148
    e37af7fb624ce986ad6f37d0e5d2b382ab5a26
    957bfd9e03b224393c32b852ab86af85107c0f
    aa61ea50a79850e6d83876a0d0be1cd7a15b8a
    23e7311e2fbd9a014de808c107e96ad11c66e5
    ca9b9140e3f085b36609bb8dfdaea79c78e144
    39a183f590b8c05aa2110af989765ef88cb5e8
    45c4c888e1bb90a93dd87cf8ead52af36be75c
ba/
    b724553f0f1a8d8ae136bcd0200db7eb3120d2
    45ea2b9500e62b8cf6786432336f5b1ddddec1
    3a18b661bccfb35947f5e9fef38744029976ec
    b11b80c60f10a4f3bccb12eb5b17c48a449767
    44f9929850ce60da6f70fce9c7dc4d9e614b87
    c8c263e0ef787691eb475ecba9fe2f756dd68b
    8fe37b7f7fd0f1e46666e3644b6394dcaff644
a0/
    ede8ae72d312b67eafb826ccfff5a9ae652e81
    306d5ff5cc4a2eb76458c127c462efe59a566d
    e91bd74f2861846945b6b03e9095cc280e7fab
    1337c7764e1e1aba1f3ba378a1c9241f31806d
    ce209e5b4be4e6a5786e6b87ee1ceb24abf27f
    ea32d11a4faec034f19b4affedf30a7b0e686e
a7/
    95edf14edfc55f6330a78f6edeade333469dd5
    53e2a3aa24383ec6ac8fd125a0120c1d6f9029
    438d6350d8d0fda749c7cb8b019e8094346815
    1e1dd893361f0d7d442b51895f20c674f3bb4f
    ac4e6077b065fe11015bb8c9b11547bf5e7798
b8/
    85d6a8eb00a66d2b09dc1a058d0ff1cb37860d
    fb2154b6d0618b62281578e5e947bca487cee4
    ece669ff236e0761528024ee01eda9c1fbacab
    7dad4cac9cf8928a9c11041926c558aadf7500
    bf6d210aec669b6b948942eda1db953e8725fa
    1d13c119032ea5c6488fcb24a5870de9c5b269
    266b9a60f8c363ba35f7b73befd7c9c7cb4abc
    299457c201dee1d30b0465130ac596e3b37205
b1/
    4eb1a12cda74af5488287ad6f1334142a0dc97
    7b7e4530b185a4011f4dc3211ddedd6d6587aa
    7ee6511742d7a8d5950bf0ee57ced4d5fd45c2
    9b8b05218193c866f9dcf82cb943165f2afb44
    eefee7c893bff24a8dff3e009b587015b22728
    ea8105dad6e27eefd5a34f64dfee974a5c4f71
    ad630091381c6f48cab9cd3f4469eea6ce0654
    1614b470828e956597f7f6831df24278447eed
    799adf2569d6a9523f36256bc7a70894a2d28c
    84304be87845a0d574bbcf6c7f179c452972ea
dd/
    d37fda8fd3d94256f107b1b6a5bf761ff34342
    fcf7f72f31658d75c8128de0732fbbf0e12b15
    01849d997e5ae9dc9809295e29ceb871b14216
    0d648d49a7c1a62d25ce5c9107aa448a8a22d1
    d0823c9049c06adb2edb7cb162555ebb4ba33c
dc/
    5e36e17288a096ba34d88a3390b9dc6ebcfe3d
    a37193abffab8b5b388018f895f197316ab652
    2b0dbee35b3fbf4705cd1987196257d0638e3c
    8c44cf7b267cc122b491566af0b54c85c19c92
    f08a528718804f2959754abd6f2687d370d82a
    1b4ee6944fefea471bdf3361020f350470ff7b
    9ccf62c20afffdedce9e90ada0e01a9e0705bf
    17b0d7cf4ad7cbb92f828caa330ff132bd02a6
b6/
    6d31cf1e2948b226471d49a1f7c50ae168a602
    714a5cd92facea3fca739d077bfaa5f1e79a52
    ed9a78e552806cb23d8ac48ada6d41db5b4de5
    5e73628f9fcbeb7a0e615291b918347e821733
    c795e4b4efb39224f7b6d33f43154df8150963
    59673ef3c1d5431e6699898ae4d073b4be764b
    f8d57e854b77f60c04f59a7f3ff74476a5f5d6
    adefe330a1a02073d5fc4f36f2d205b32a4e89
    38b28b9480ff7cc84bc890fb7a562869a47229
    ee7f2039801c9792dfe6e473843fb0a4bc4a5b
    bb21a8b26680b38c3af8278ed139b6628356c5
    beddbe6d24d2949dc89ed07abfebd59d8b63b9
a9/
    5956ee65cc88f8eea8accb8d0f87c5a85843c8
    f79e9961f0c8b6785d069b0574837c16607cb6
    4030346897a055de8a2ef52fa27dbe320f4dcb
    4059026a26b8c883ce5353cf18e49f55cc190e
    146335caee664ba1dc3d89bb7311aa149bd85f
    60b2f3c5f3d11fc9ae43638da9877d635e8d91
    9fd0c475cb63468599d0a2b6560f4470e44046
d5/
    4bc63eba364bda3f869a0f3b1863b872f9682a
    f74998e3e863b4e9c4541300fd48e6a34edc65
    22d80b5189554d1acf9b46d5db1981b946d712
    7ffdbefd65fcf72818a037ec4e0a4bc18bb482
    fd4b71fed1bb4871717f978f0c470280f099c1
    8aa9db9aae701c41f48ddee14af171a4f63e06
    b238608b2af459e3db803edbe1b23a7955df7b
    d505f8e5f0c2b3ec3f21862bed8576080a962e
    c27b341403fa03283562373b2a0b05daba075e
d2/
    1d697c887bed1f8ab7f36d10185e986d9f1e54
    706242b8aac125a66450d5ce8dcd3395336182
    52feb96f151f73546fd690f1e46b3123d20d58
    dddd6a106f021a4723c1e8f5953ccc09e55e1f
    10c317b710642b6ff3d760b5a9e91ac191ea04
    1177e5ea6c5fded52399abaefa6eb48053afa7
aa/
    bb8d001a5e72c70ab424ac3f28006bfab9d7dc
    89706296864c9093e4e5c00d2a3de106d3aed1
    d248fba92329de664923dcdafd4817d8f44471
    0c0a7fcd100886e3cd27b3076b6b30c4de1718
    d867e8c80b826bf6a060116f17fa08a8eb0765
```

```
af/
    3879d0107568af28006b12df48968fd99f7b71
    c21d8f6e328b8a46e566faab496c82ec21ebe0
    3fe7f21cfdf2c0f92e34b108018c99fe28a32d
    efe525ef13ac82b24b356ccce10e6f5141e4cf
    bf597a8d7ab23de3ddbd0cca691c25f3c40402
    e8da1a4a30daf6e48ffba514656e7c86c9abaa
    1d80cbdd3ca58903f9516e3e4954b58a32c905
    6f206c2d78d7b150526bb59e29e5550b50c479
    c3265ef8625926e81f1ab983303967c2153535
b7/
    2a9a15dbdd03ec586ce08cde32e7afe3b59bf0
    a2632afab1f2d3bbbd40376ec601222942a760
db/
    744ca15211a5fb026a544bae27870c171bae2f
    3995eac9f9ec2450e0e2d4a18e666c0b178681
    8860f7ebdb360661bf81acd34695170c2a1c36
    e6cb4ca471f146b431d2fbb558d47317a103f0
    1ee08676b761ce267309457d89944afaeeed3e
    cf2a7b0ee2898b72714b756e4b27fbbad4beab
a8/
    a5990629ca73f828878b389700e6a592938c5f
    4c83e48b9bef06aa9919fd5c0a959d6f1f1e49
    a28320a17e7d71b47d5bf02002e1a6873ce61a
    1a23985198d2eaa3c25ad1f77924f0fcdb037b
    17b6d8a795e10ca7facf92620117cccfc5dbc1
    cd1330f0f73ac76832bdbd6b455b10bd91ba83
    745871f8a3f43c8918eccdc243b6141c5670b0
    4a1fa28a2da86555e02ac467c0294c59a20468
    727ed8592533a009b6202be92f438d4152e793
    d626a5f34667bfb665269ecabf2c4d0d8cfabe
    1a0aeaf0923a9ef6b3f5254b2ed683e84a84b8
    d3992f622157cbf5532054221d076d76280456
de/
    a53fc62ac7bfe8c4161b267b357b398ab2b8b9
    0774eb97f947f32fb417c941b36c8f474e605f
    5aff6d01e3de1338b37fa37e78e5b9c06cfc79
    04e1d73f2d86fb3ac094c82b9109ab71a0f917
    6a0153b777f255a754c1ca9f8e4dc55cd3934b
    9a09a4ed3b078b37e7490a6686f660ae935aca
    ffd3ec3dc55dd6b23f36142ac5fb912f438123
    b4937f74f9a1ccc5fe4cc7761ff5c9d4f5c3d4
b0/
    d1ef37cbccbf20c0606fd1132bf58c26d91da0
    ac90a5287492c31a7a596a4820288b79638ea2
    66ad8ceb9f5623079522a3e7dcdb7b2b71a40a
    1fdaf0893557c0375fb133adbe08b1ec7349ca
a6/
    7f49c80b1d829a577ee2f17413c02f2399c6e9
    b94ddf347b5f05f91b4d518636731e9eea161c
    32e198e8a3785c021c6974e847c8f28e2acb9a
b9/
    b94f43c683369fe88a48e25beed265d9ac34e8
    51c2defd0b447a6974de79cd7353255b613f6a
    e1d28e358359e1a8ad7d30e1543389be6fd648
    9d9dadcfc3789629aa803d3256cd22d2873c29
    54956f61cf5bc971b344937b2270370076d6bb
    f6af4d17410ce7e1d573c41a1f04dd18ae275e
    4c32511f0cda2363bfc4f29c9c8bfcc7101f9b
    4e462c4bc5fa8a9f14dc3561578712f8f48144
    906feec2f9cd07de6d0a8a21c794179bb9aa79
    0fbf7f35097694f727e201b0b378942d70a443
a1/
    d2e8184fcc4103c5e0014c64e69c75b25dba60
    6fb66629496a9e73179854f9efc901d20c3ed3
    64428ec2835a8c32981f3d29a49f308ef27966
    10ed0d5b4a748bfe338f36ef820e199d2389c5
    03ca11356606402c03b320a4fcdb8635051623
    1f6b4d692b9701c64fa76c70adfd191b1bd428
    b589e38a32041e49332e5e81c2d363dc418d68
ef/
    42417c208e93c55d704728d3e88dfe46250d92
    09c60e327a0122e32f95f2f10a826a033c573c
    d75adb279a5be0e4c8ba364b482d97f516cb10
    fbf49457c90e592648a8711afcb8aa6bb650b1
c3/
    29e1977fd1ed403bb65529296d5c803a6b289f
    26e80dd117458ff6e71741ca57359629b05ae4
    8ab3666a9f721623afa88f60daec487eb53518
    e546604c85678dd72db35893c46ffe2d79c052
    10b66e783820e5596bee9e4d92e531d59d6dc9
    823e257ef1de3a79d7f297f38f3bf0bf06f24b
    70d95f6111ac5405a0edbf47eabf4795991430
    5a215e8a25c8bde60da93e6dbaec7781fcefa8
c4/
    058fa0bf481ccf434bcd3df9f1019514725677
    ffe1f99e6dc9c0509459196cb68fa95e79048d
    cc4b7d690f3abba19a9ea8c4bf76dddf859c3d
    5417da435f58d022c59511bae82716d5bf43a6
    3d8a3496e0b30ab001e991480eb91ca71f292c
    5282fb5366a8dc380c819b62532d9645d8d620
    db8f4ef21a1893593182b46f3fd7f2043c33a7
    5f193f74ad7385c84f3b935663198415cfaa4b
    406f148f113d0e3024d0a22ecfdfeeb95bd712
    3e5f10fdecb6606a1b75af3e149cb6a0a55e42
    66378ceba69a335d2beb4d3af92703d52b3831
ea/
    363d86a564b5450666aa00aecd46353326a75a
    0447482ffae43e4a7f8b5e55cf4665e265732c
    1fcbf91fad8e07daad0295625ee92f5f34175c
    c1516261d631ac59d9e145703eb27f3a5d5819
    94493f21e6f5583469d882d08203381ee31117
e1/
    615e482df38eb3d69dbacf2d23f33395539ad7
    13a9fcc8e33722aceb5f4b0360d2d5aeb57395
    924ffc9ddcbd6bedd09612f0bf16d6092711ac
    25798463512ce4322a2cc139b4e5c1515e5c05
    ab8f8f589eadabaf3efa068dce3ff620a01898
    9c30b18905a39466ab6b51403438605e706caf
    2c10d033026f09cf97b81d29555e12aae8c762
    a3f3f2859322fe1b8fb0e7deba486691b47182
    2c52b538b37da441f4431f620a943130dfc0d1
    d1787681834b0c10199d2e7211ff9c901d7d08
cd/
    e4558fbbeb938de8a5aa8ee165465c23180ffb
    9cb8d40f135d1da7d2517630816605a0805fe7
    d727f3c5ecb9eee7df6d7dd93a7fbcd574eb6f
    a5affa7bc16505c73035fccd9415e1f0df6f69
    0b3eeac3ebca7fe4a627ba5a96c1bbaf827d4f
cc/
    e05582ffc6fe6d72027194f4ccc44ee42f1fcd
    9b367ef922e94fa6db2b7fb9fbbd23fa0c8963
    ec9379dba2b03015ce123dd04a042f32431235
    e44dc1e7973f6253ff4c2936608eaf47bfeb5c
e6/
    e2be29f57bcb74bd4c277e98bdb748728c8b82
    e498efabfab0dcf31cd7731f8f821cc423bc4f
    0988d643e007801f79e8718354e7d00c7acf18
    fca4d47f661ff16fdc8c2bb7ae5b86c7f347b2
    ca31f52649160e97869c80c6433a3de484c531
    51861b21cced6904b34842f84d498a97adec2a
    3829ee32d78d6b5a3c3be9eae5fd302cafc6be
f9/
    94f178d2009794638030246e04f96fbafd40cf
    9c00b715e05884fd84655fe813c5497dd52490
    2d14b9336fc4c6128114450979bc248a41c702
    349c028360d541c56962d6a09bd9c2a00e3a37
    4fe01b71d9aea3c2a61c24549e4a93d2ab4daf
    a60751285785eecf91921ae9148930ebb427f8
    f9f2d844e3208b9133d9e775ac0775d8ff47ab
    46ce87848c966be17e0ed752e5b2ea2f50765b
f0/
    f9c74fc39ea6ca4505ecf7cc469a487974e3dd
    1dc043c1e416e7818c71dd4cf70fb233c24cd5
    99a3dcd28d2fec21457c9b6c01ded4e3e9ddee
    28cf505c481f30959f6c252251f973f72b8e7a
    e8acfa2880ea34a262c0b1e610e71916d732cd
    a853e4f32033a3b8d6ed1e45d4bf6467ee679c
    e1896cf51c6dbe8b98efc16633783f8c41f8d9
f7/
    5b89cc8e15da43a5bb5da05e096b6f53892af7
    8e4838fb3a364fde4eddaf5d5b6b1557fdbe0b
    17c1ccc79f7581f1293b3fcf1a0764def7a84a
e8/
    280a98062a8774cb027b7f7722d358203343ec
    ca24f73c9b71454667a08f362f24102e962cbc
    629796ebc490b709c00146234c25d5812b6440
    d9803e4e3c97cdda54fc054932747a38c881d9
    385fea84032763ffbab6862ab58d3dc7eb6400
    cebc1bef7870a5e772ea066c485eddf5c1c57c
    94aa1d9c515d4d1aba9afeb7d3781fa8e56efd
    dc26907bf4b64a4d6056c5e87bf78b44986d85
    4e65e3e14152a2ba6e6e05d914f0e1bbef187b
    3f959a1b7880a4bd2b4e500e884c41edfbece5
    bebdba6d8f242244bf397ab067965d47c5093e
    a3a674e0070159b956c29c5092b0f72abc969d
fa/
    7307ed8985ad7e318660da0066440f890d1624
    d1a1a04e53860d4077119921a10ea476a62deb
    4ecb154b501a6adcc6d730725ec111c3f924b9
    0c4dd40381addf5b42fae4228b6d8fef03abd9
    6fd3bc9c41d2c10c8c07cb6470e128c5609f2a
    0b245d279e96724d5610f93bc3b3c8c22ca032
    1a87469581f31bfccb62da40aeb8c9550b10de
ff/
    5efbcab3b58063dd84787181c26a95fb663d94
    2fe2e72f5521a8c858d73a98fd6e238d1e1f4e
    ea394436b9d4f387d9d0ff83d5d1a1c3c7629f
    cc624066bca91480f6ff9ed36d42cf3a35aa42
c5/
    fd1187b6ae99f9a69ac57b9d9f4d0ddc68f9cc
    903ea264fe99f73fd888ae40fe8e5091ed8aab
    f0492ccbe9c727c835c12c84a1d8340366fa1e
    ca2d85d5176c65a2e90000b0d67390573120a6
    e9d85cd75884b129d4ab8d0453c0e50d0c1f68
    a55d3ae814e60bfef1f122358087340c027420
c2/
    646794a98578bdb735f5047dbc6b1d50b90230
    1e109ed6db376a83ea01d19daceeb2fc99745e
    2d9f97684fc939c33f592dea0fac23bd90c2a1
    a68b3418c7faee44ad718f79f36437e7128b26
f6/
    a300804f4a99eb79b4f3a1ee676251c30e629f
    e9c2047e7ae02c5ec5a0647a2c1edbffbfc5bc
    720cd678de3413a2c8607534b134183b6f52c6
    31ae6df4747b808cac7c03b38e3e1d48bea00b
e9/
    bf2d3372797eedd6abe967e2e260b4c63a61dc
    6d2b4924c468c666f3ad6dab902f217ee43c39
    92cf4f1556072a69f9d4dbb57217ded54852af
    9d87ee75f6f665989a109828e07ef81cb3410c
    c57e33320c247bcda368bdbf540d3342f4cf16
f1/
    98fc313ff57929d95d36216e3e6ecec3877673
    a0eac1ede2ff3662d574bca693f369cbae8dab
    ddb2ebdf9eb702718fd31e09ff92b592da519f
    a89a6966491d24379b6ef87bc5a3b7eebbb170
    75de1015c2c65a7e67e38af59b0aa9d5b71d91
    2a69a180a91b41885e68816f13298e686f7a3b
    8803a32de4455ebe84ed5ffd8a038d84a2f8f5
    4ff32096eab20a8cc1a5d3c2b8c8cfe1fcb2d2
    ea0cdd8a754b4d0ffb530d6820e74f59deedb3
    744d85709d31eb502c467bcab02fa23230a316
    d3d9f7fc750feeadd6e05cc27ea871437b019d
    637e458d0fee5e05488d01a4c48632f2835728
    bb0aa19a556725aa2ae2b8cea95489c99a9078
```

```
e7/
   053bac12fdb7b2cc50448f88318cd93f62cc0e
   b4d5456a7fa18d31d8ab4e9ef120b42e61ef13
   ab538b701f9d8f01ea08f8dec8eb7bdd1509e1
   6a60c395eb62d5f05d7248cf67210cdd10740d
   3961661d17e939b0cc3aa52fdaa2251d9f94af
cb/
   d6da9be4956ce8558304ed72ffbe88ccd22ba5
   d2ae5e53bf30a0024f2f74fc3365744e21e7aa
   a6f3f560f71b3b15ab6aaf21dde4f1bba1bd00
   121bcb25245ac039e8325413c4f2be3ad71d42
f8/
   9f132884c4698844e98ba2617d9aa7db4a0a9a
   863bed5f40df8f087f21876e19b843894be001
ce/
   75338f7703565a71195dc5467f339eaa57ef4f
   222f1e52d3e46840602cb9a8b09574a8b07cd8
   2c4bcdb6bd2d931eba07dd70c1dfdc99722e26
   e3b5b56664ebb03268e03c34123447bff983a8
   ca99ca76206390e75303787ac118a4e8f17ff3
   3a3c308b1da208eba68674670d462af07a3791
e0/
   c7c429b85d66ca58b184630a825a0f87e76f53
   e2d3de92d1c4acd6e47943d57a7c57ca675afe
   6947c051a7d2273260343eab37d9437f91e781
   76afecd741f3f8e59a6825f5dabe36d223c77f
   a556d1d9b9e0ab60ad0fff2ff64e87a92d5d87
46/
   8b0ab4957f7cc06070c284c4913b4fac94e198
   5ecbfeb0dda2d2f97963ab15a84a762ea9ac26
   5f9c777e359341f7592cc7059e835545415f22
2c/
   341af81187625f0877e6e9b31e49db0cf5ecbb
   8fa7999c2c3441ba2f3aff98c91794596ee654
   bc8c6de0cb3650fdf13d039993258eb0b17a30
   ba4b0708032d62b4c1278f99e5db87ed8d90fe
   82f95fd84d1f128eb78d5bd65ad37f773ecce9
   fc9d60168213802e24685694385888af58c123
   7facde830998f629d7abcdc0ea9ff93a96b9c9
   9b145b58ddf148b6723748d847e499da788df8
   3d0e306f91f9dfac1843b40babd223766bbf50
   c433a4a55e3b41fa31089918fb62096092f89f
79/
   e5b4c025afffa24dd81d18f4f277dda7987691
   1f0465de136088e33cdc6ef5696590df1e4f86
   08356092071fedd87df47d0a66838d8727d4a6
   b82a570e5be5ce4f8e4dcc4906da8c18f08ef6
   1c874c0ed70ec34f2dbca9afe822d0be4e2d61
   19bcc88c95085e65eb33f15ad49e07dcf6b276
   e10458a5a89c500e3110323c89b16145f57524
2d/
   292c2f062cd80cd108aac503eae7b635ceec8d
   5a3a51ac233d08a4ea71c6a6dc0b3558e58787
   64860b04e018560106819ea19724bb63e8bb86
```

```
    787397f98c0ee74d43b35d9842cd69adccd125
    db8effed79053d2ce9832a0d61d5bb0dddc43a
    0b80306eebb60dcbddc6e897d8c3f0920b4b96
41/
    f553ff03969692dce490af58e32a7493e3f2ac
    784104ee4bd5796006d1052536325d52db1e8c
    cc42c5677ddf0709d9eeb894eb8dbe4fd16f91
    58910f313209833af630131e259df7099aeb2e
    2680c065ddd0aae23414ea42eb497483150b45
    9b74d4ca5925eb9099354ad24b8ee1e8e93639
    313c0c36125f88aac95483560d48ff93ab848f
    30a421cfd7260d323b13cbd9d75ab8146e6030
    f30466f1fd658341bab5cecd650187ca9e4c6c
    7379f7ee6abef120b8a225a6dba43e2c7458ce
83/
    afdf208b883e377a2d16ae39662ee810c6dcf6
    c2df75b963e5866b63aaf0f4446a8ca61aebce
    f9018ee9357bd193e91abbc66fe9f0e7075f2c
    1659b3f202528deec8c59f22290ce1cfe24730
    f6a2dccf09247151aaf5c44e81e62cc9b6adcf
1b/
    ecc5093c5ab8e196bb9fee415e2381e7158fc3
    2204f59f2ce4d9c8f2cca85326e4d81f8805bb
    1b97806b65a8463fe49b36759117d970cfc6e1
    b5a44356f00884a71ceeefd24ded6caaba2418
77/
    c54013a3f8d2f028f13831c32972a213b4d722
    70c922c84fabe0031333a4de305dd6d6852911
    eb39a427d30152d82400f0a9409586ad03a9d8
    a160fc56cbb0119904bf7d5c54517563014d73
    72cde48accfad09d32de145f092d0f68827c13
    8b14e3d6821e386cb8a07af59f27ec76104468
48/
    8738a21b57fbc5f6d01bb09402eb95ce0cd332
    da8641c636edede90d6200076eaa0a41c88fbc
    0aa5b6ed239cd672b2b2456b68410ff9962d91
    9cad930e0029fc2f8e5111df1bad38151a07a9
    7087351b066e2ce96d0036fa1469ee8a004a55
70/
    2bd94566bb99056a3de4cc7098d0c947953bed
    98a1e2e9b5d12ebe7e95d1f1096c8f8ba44d18
    5f416d6b06ce5f51b3ff47c49d078e93c6f034
    6ba600a93c1b72594d96d3026daaa1998935b6
    95585008e712328fc5536ec7f72210b878ef36
    e7f8359c6b724c751e7bb46fc1bb546a041e05
1e/
    8ff50edfb8059799b334325e65eea9bb9b1ab3
    6dec4606347a4db03c36092c4fc4fb2c6d9d91
    00ffacb182c2af206e5dd9d9fbc41d236da0d1
    b430c6d614a5daea4139badc09c222a4b0e72a
    c07758d24583d289a7dcd0945f36dc649b248c
    ab7dd66d9bfdefea1a0e159303f1c09fa16d67
    af3b48ecc802b9cdbbf4704028f164546d777f
    e5091a3a4e18ee8d80c9b3d674c53ae15b8aba
```

```
84/
    9356ea9a03a031abce367b955a30fce26c9845
    3cffc6b3ddd6eb01483bcf1b5c33c717e027b6
    b134e490b081d661daf69f98e0b9b1fdddd36f
    a3ae874a60ece9bbc5da5f4db0b066fc576e2f
    1b0e270a381cdfaca544a9be976d7276d83b1e
    f9eea4f3c4e588f5358c586275f2ce8a647630
4a/
    7d55d0e50cb8b892caa021695522e5ddd54a17
    31dc568214e72f0bb3e2e6d6b2640ae272389f
    b6049be017bd9f39afcdb7cdaf0de16993cc2f
    2ae0acb865a3f1c36d00730cd076a12e5d031d
    06bc69d5c850fa9f7c4861bc6b3acca3905056
    e90f0e581a4b1ae8f7a998543af1769be65f0b
    1c73203ab9841821753995dadd022a17f2b193
    32ace4a25ca366c2901160baf6035a0a6abd30
    7105d17916a7237f3df6e59d65ca82375f8803
    eb29ba713d2a468d47bcf298c75eeff7dfade2
    936d0263d3446c3265c18ff61175788db1324f
    b2915fb80caa0eb282c675c0763060c639448c
24/
    66bc04e4dfe6aff46fdd14578cddbbe169f0db
    90d5e5b63359a7f826922dc69c0015cb9a5b2e
    d6a5dd31fe33b03f90ed0f9ee465253686900c
    35ee165ce4d719a887c39b3e6e7ea9246e7a3a
    9f8543ff8304075324cdd187a7d11ad3990846
23/
    9f31585068ca6f1b03c8bc7e7dd7e447037c89
    37806bf9349edfe96c299af7898afeeebde503
    e0d6b41ce6a36a2bc1a9657ff68aeb99d8b32f
    eff2777843f121e2e497801204ec5f4a715de9
    3f5916dc270b686e199b6d230a416fc252eb36
    3523e5ec3fc6413007fdee776f7647a53413e6
    801503b648aef769cb3297b30ac67fe555af72
    a87e68388fa93ac66704cf1b9cd9ef999f5b80
    e8e184f26b6f5b0d1330652b44a0d0e1ab9c0f
    dc3392a2c9b11f93bc8d9e1381848b7c8fd7b3
    b6aeafe4f43d097734e186907232513ad27a3c
4f/
    1603adeb6fcf9bc1c4a16a9b6e16223c6534f3
    704a3547da02f913d6cfdbd4e0ed77c81caabe
    6d8b2d79406012c5f8bae9c289ed5bf4d179cc
    bc9dc6fe90a3f67e6dff8722b224826f33778b
    2985af1de4a97b356bfa4f7748e6f5cf49f9cd
    3003711020eac05ef5a19ab29ba5670d89f642
    93acffbdc1d8ba9555114c190e44140c34c291
    081c7e9253f5f693a0f5506f5a0560386f0356
    e802bac91d522f7efa785968b7a1b7a79df07e
    a9173df6e996e8aa54739b8961264614825627
8d/
    8564b6fd84e272074ef1cac149f5d3f4fd8396
    e485e5a3675890010de119e232db347fec6e53
    aa25f513d02f190fa48c1e5690b052ca421c6f
    1190975f9649db55fc874d44ba57231d4d9648
```

```
   3a664c7d1be57579608a7c1e1da4570b439a19
   1d499376744954308bdf96f80e5b5a39a24195
   5a856ecd6810561790df0eaf24f6b61bed6f55
15/
   d6b92a59071d54e605cf690641df6f3139a0ad
   0136938548af6aa5ae1f716b330d0eb2d3e013
   124f1b4cc371dc60222ee68d2574a14413495a
   14095bd7544000b38abf178b8ef7e7a73d7544
12/
   e22336e7935a3bc1697ece8f182f3c9197f538
   5189c6fa57d61bea0012080eba12f90faf740c
   1801ac7467e446bd725bf6afe4e839eb787e6e
   adeff7b6eacafc9c8c655c8f6633622b646992
   ab23713a70dda46edd300bd975b02bfb2be031
   f0dcc7064a1413cbd55477655133b97004e9f9
   219f124aeca6d3d7edd2621071f100c7ecd90a
   9e35ee871cc4364c014a485473fdf956037285
   bfca005a2161575c01744f062aab6c56ef5ed0
8c/
   cc31937f22304b9175009d39879f2db231f96f
   7359137a73b93f2df24451391c63ec91052b18
   6167fb3a6b390c5c0a3ba455f76cedf34695f2
   3df0b982578bd1757fe9d5de5726a67182ae54
   5661e93a205bf4fb22404d4fc50f902cc31369
   4ff122b7ea6a8463651f4a96a80bd3279eee1e
85/
   afbe3f46c3c4389b44c70416ffa540b13dfa72
   952ac18b377b742e736f34a8b2eb194918c29a
   51096496a8a73b614fe87b4895f7ae037aa299
   db9e328dc82599673eb6c551f630d9c6daf36d
   8a41014169b8f0eb1b905fa3bb69c753a1bda5
   01893bd153b7216524084cad23e90aeac0b1f8
1d/
   b26b06e3dd0ceb757cf63c4fc7fdd964470ac7
   c45ac13680332bc368aa383fec751aed9e6220
   8f8e992878341dd4c1e6b4482d33144fa17407
   3601c78b0d5cb5d88e586a06830b62ea3f4c7b
   e0c431276bcea456c1d692a8fe339d72d51489
   d950c489607d06ecc5218292a1b55558b47be8
71/
   a13db84ffaf69cf467eaafb1118e940da9a77c
   9d69dd801b78b360c6c2234080eee638b8de82
   63193f1edfb401945293b7819dd3113e40e126
   f66bd03cb713a2190853bdf7170c4ea80d2425
76/
   e6f199c0042cec6500f53c062ff9ea1033e79d
   527dda41f578f1caf3a0ef3256cd71b8e8d67a
   d243414d00f54a8973359cf553123e9bd1760e
   452ecd778d7c0c6f67bc7f8f40036b8a101226
   cc7f027015bc36960e4ab365ce7f9b4f50ff4e
   56989be6507ec66487d1533c5610c86afcb11e
   86fe85a7cc94188da76bfb1c10ad2a10821256
   1b64715e77a8869ca577edfca3ba9048bc88d7
1c/
```

```
    5883f8b2298313d05450f6cca60cf3751b6d8f
    80e46879ecec5e51b1a5cecd13ea09748b94b5
    f87fc527d1705ac9984dc7aa94ba1127d4045c
    c8a39ff0d800c414cabf90556d24868de4d253
    d676e2d2a3736af16ebcfdea58f98830fec2d4
    a9ba62c208527b796b49306f4b8c95eb868a51
82/
    eab36c800a30e09bc354ab30e2daa9ce4c4abc
    ec50d5106ff0ac41dd1c03c2a789dbc468c401
    95e80df91c2fd5e4fff05a058c8b550a99e819
    2816f964a4e992eb2b5670caf299a5375e7b7c
    67092a5bd04cb7131698966cee47c654a83456
    647104c9097ef3d6996d8a67b0b28d3d2b4786
49/
    a148a097e9cc06c165571e0bffaf7cae17dc5b
    f269f63fbc4a97cf42acd02055c414a3de48fa
    00ccc160a1dbf4de3a01c234735c21dd4417d6
    4d97d16f622346de60669b9bc93f2d6c181b67
    3b53e4e7a3984ddd49780313bf3bd9901dc1e0
40/
    57b76483c71d05223ed3bb01f07604b8b38648
    5657d509388cb784537280ac6e397b72a8c2a2
    190733d6097c761295ed5a9f56bc949d08dfd3
    3fdc81484ef581136a27b1a93a42a05469d8f5
2e/
    aa9ec173b085d1cb29b83db96e54b6f84a061f
    ab011b6f2513563ce50adfc9fdc0658d201b20
    50cd7b40ef18e7f7ee56c0f528bf0ef88b167a
    3e607a82fe5cd0fdcc1607752b4bec2be90559
2b/
    45d391d4d7398e4769f45f9dd25eb55daef437
    d98ff2504bd5d1b76e226b42d64f0d3ae8afff
    d0a7724f4375a5272cb7c101be5b6a582dc031
    89553cf6c08441b1bab1913af6c0b41de39e63
    d9eb0073d3e0a6c56311b42097ff322f75dcdd
    548beaf4cb15abaac086438a52653a660ec93b
47/
    1dfb2f9271c073f0713ca98f8db2f89c975071
    cd1fc477c375c360fd644264039383ece6b3df
    1665754e9f199f07f90107ebb350c38b378100
    c7a2319e86f2d5c24e3b1722c9a84c55d4b61e
    efa377c5be57510abe5c36bb35402f199b1782
    41998a2834ea3b0eb2dccd34d7930dbad649a4
    2fafb4403efb9673d5cc724dafd9cf764aac5b
78/
    55226e4b500142deef8fb247cd33a9a991d122
    e18a6272482e3946de83c0274badc4a5cfcdfa
    0e362af9466ca04789603309249fe221711d88
    6e6bda63699b72d588ba91dd73df017570aee5
    33eed9d5ea3de01d772d5d4b9eb5accee2909a
    69d9f850da916ae86302c8f47f28997bc14b84
    49f21587d66cad83610614bf6334c3edd9c0b9
    5d0057bcc0ea74a4b8d65ab7a0de78474bf892
    b5c13ced3d0a429b6d292e2b0b985d50909942
```

```
8b/
    98fca7233be6dd9324cd2b6d71b6a8ac91a6cb
    5b1d280f3c7b45cee54338f60d5271a7510c2e
    7e633dea271a610c8a00ef41e8aed5f59cf6a4
    5c8d205c24d508a0dd62183212c6360842bdd2
    0a315f32466ac03a205898394f958f221818a7
    a4f094de0200f29efa131336241322a613e7d0
    f278419be7cecf4bd562620c52f50fb582a57b
    6f5abd9d4ae92d35b7bcb9225c4a043bddb53c
13/
    520e3178013dfc73f32292f7cdc5d8581d5666
    ce1f7ab78ce1990b27a1e308d4bc133355fb97
    9995ac3f109a82664e4913f7ebc32ecf7617e1
    00b866043e22e3b318ba791d31333ca8fe8514
    e8ac7c9bfb0668efb21eb05a50c6cf2f5a592e
    63be0cce6230b5be3e339d9927d593c0dca5e2
    cadc7f04d4c24afb829c9fc44367ac06a3c61d
    4848ae526e54e2b18738f83088c4a17efcce96
7f/
    809efe9222cd67c6c7ae72ac4fe108b308846e
    9e1e00ccdb0e67a5601db4707a1cfa46cbc96f
    09607afe6d3766c14677982c1d1f7fa7c96848
    66f7e3e98dd9937db4c572d708db5c4428e7dc
    3b07a02b1b7338d430c1fde0d1052700660da1
    23529f1155cd3bbfde335ccdb7fc483b9d2d19
    a28e77a43e640d3757afce7b5eeb129d647d04
    723b21977fd8eac35858c55366d1d61a38858d
    a567f22a8feadbe53b7d9db3df39e9126feb2f
    001f35ef20b63f6b6a5954864b69ec5f37efc6
    e1a3e33a3adbfd9ad1126a22d7175154ebc200
    009fe9bb04116f3f68e0df3d90c0d84599803e
    416e1e799abfbf62382456020cc8e59e5cf01f
7a/
    554795416de7711d32c721101c3c65d808d90d
    ed91e000a9eed5c9a330f43899d2a3db62c8e0
    3c4c7e3fe16e91225a87cbc58b8bbd798f9cc1
    666b276df018333e3243984d4182451dd7af3c
    0554581a96accde7690580b25f3bdef1971ef9
    17b7b3b6ad49157ee41f3da304fec3d32342d3
    be989c47a074f65c50e765d1a908330c596494
    916ec1d75b96e61afd2aff66bdab729bcb1d67
14/
    da132ea6e732f60e7cda5ba9510b3829a9fac2
    876000de895a609d5b9f3de39c3c8fc44ef1fc
    b7a3a7316dc9960807e260eb3a9d7e98b28087
    c0e4823c5cde93b407a3ba19a6326165a88950
    b10daf3a96229be87eed34c643e962a0d30450
    25d10ecaa59a9e49b73cea2b8b4747de73f6b5
8e/
    1212a6e7a8e6ea7df28f5ac312129c20c1b151
    e8a1cb18017880cd0bebd66bc2cec5702118c6
    22e1252200489c9060bf212a34f7ed80509c21
    9d91ff7d235921be0a60b69b756aeeb456c0a0
    053ba06c5b1be7c140ba2b7903fda838ba5f91
```

```
        7b65eaf628360e6f32f4140fcdd7ec7c2b7077
        d4a8773b8404c2705aa8728e5fd692362ba168
    22/
        6fe84dc0d0c4eb78f9b3c603df20cef0fdfda4
        f4d716ac9764ee18005b9b852946d614152375
        bc84bea0bedfe2403a6764fa80dab1ee1a8231
    25/
        962863b9375665c916b3d386d41424c579785d
        9b15ba194db7c02fbcbf170d522230b4418933
info/
    exclude
logs/
    HEAD
    refs/
        heads/
            NovaGPT
            main
        remotes/
            origin/
                Nova-Tribunal
                HEAD
                NovaGPT
                dev
                main
                dependabot/
                    pip/
                        ruamel-yaml-0.18.5
                        urllib3-2.0.7
                        werkzeug-3.0.1
                        pbr-6.0.0
                        werkzeug-2.3.8
                        blinker-1.7.0
                        urllib3-2.1.0
                        Archive/
                            urllib3-2.0.7
                            werkzeug-2.3.8
hooks/
    commit-msg.sample
    pre-rebase.sample
    pre-commit.sample
    applypatch-msg.sample
    fsmonitor-watchman.sample
    pre-receive.sample
    prepare-commit-msg.sample
    post-update.sample
    pre-merge-commit.sample
    pre-applypatch.sample
    pre-push.sample
    update.sample
    push-to-checkout.sample
refs/
    heads/
        NovaGPT
        main
        tags/
        remotes/
            origin/
                Nova-Tribunal
                HEAD
                NovaGPT
                dev
                main
                dependabot/
                    pip/
                        ruamel-yaml-0.18.5
                        urllib3-2.0.7
                        werkzeug-3.0.1
                        pbr-6.0.0
                        werkzeug-2.3.8
                        blinker-1.7.0
                        urllib3-2.1.0
                        Archive/
                            urllib3-2.0.7
                            werkzeug-2.3.8
src/
    main.py
    utils/
        your_file_structure.txt
        generate_file_structure.py
```