Py_to_PDF

May 8, 2025

[ ]:
```
#!/usr/bin/env python3
"""
Utilities for setting up and managing logging in the waterline detection cycle.

Provides:
 - setup_logging(): configures console and rotating file handlers
 - flush_log_handlers(): ensures all buffered log records are written out

Usage:
    logger = setup_logging()
    logger.info("Cycle started.")
    # ...
    flush_log_handlers(logger)
"""
import os
import logging
from logging.handlers import RotatingFileHandler

def setup_logging() -> logging.Logger:
    """
    Configure the root logger for the waterline detection cycle.

    - INFO and above: written to a rotating log file (1 MB max, 5 backups).
    - WARNING and above: streamed to console (stderr).
    - Fallback to HOME directory if /var/log is not writable.

    Returns:
        Configured root logger instance.
    """
    logger = logging.getLogger()              # Get root logger
    logger.setLevel(logging.INFO)             # Capture INFO and above

    # Clear existing handlers to prevent duplicate output
    if logger.hasHandlers():
        logger.handlers.clear()

    # Define a consistent log record format
```

```python
    formatter = logging.Formatter(
        "%(asctime)s - %(levelname)s - %(message)s"
    )

    # Console handler for WARNING+ messages
    console_handler = logging.StreamHandler()
    console_handler.setLevel(logging.WARNING)
    console_handler.setFormatter(formatter)
    logger.addHandler(console_handler)

    # Determine log file path; try /var/log first
    default_path = "/var/log/wd_main_cycle.log"
    try:
        os.makedirs(os.path.dirname(default_path), exist_ok=True)
        # Test write permission
        with open(default_path, 'a'):
            pass
        log_file_path = default_path
    except PermissionError:
        # Fall back to user home or /tmp
        fallback_base = os.getenv("HOME", "/tmp")
        log_file_path = os.path.join(fallback_base, "wd_main_cycle.log")
        os.makedirs(os.path.dirname(log_file_path), exist_ok=True)

    # File handler for INFO+ with rotation
    file_handler = RotatingFileHandler(
        log_file_path,
        maxBytes=1_000_000,
        backupCount=5,
        encoding="utf-8"
    )
    file_handler.setLevel(logging.INFO)
    file_handler.setFormatter(formatter)
    logger.addHandler(file_handler)

    # Log the active log file location
    logger.info(f"Logging file set to: {log_file_path}")
    return logger


def flush_log_handlers(logger: logging.Logger) -> None:
    """
    Flush all handlers of the given logger to force-write buffered records.

    Args:
        logger: Logger whose handlers should be flushed.
    """
    for handler in logger.handlers:
        handler.flush()

# -------------------------------------------------------------
# Example standalone usage
# -------------------------------------------------------------
if __name__ == "__main__":
    log = setup_logging()
    log.info("Logging has been initialized.")
    flush_log_handlers(log)
    log.info("Test message flushed to file.")
```