## Overview  
  
This file defines the configuration management for the code monitoring tool. It introduces the `MonitorConfig` class, which is responsible for loading settings from a YAML configuration file named `.monitor\_config.yaml`. The settings include patterns for files/directories to ignore and specific file extensions to monitor.  
  
A global `CONFIG` instance is created and exported, providing a single, easily accessible point for configuration data throughout the application.  
  
## Classes  
  
### MonitorConfig  
  
The `MonitorConfig` class encapsulates the configuration settings for the monitoring tool. It initializes with default values and attempts to override them by loading from a specified YAML file.  
  
\*\*Attributes:\*\*  
- `ignore\_patterns` (List[str]): A list of glob patterns for files and directories to be ignored by the monitor. Defaults to `[]`.  
- `file\_extensions` (List[str]): A list of file extensions (e.g., `.py`, `.js`) that the monitor should check. Defaults to `['.py']`.  
  
#### `\_\_init\_\_(config\_path='.monitor\_config.yaml')`  
  
Initializes a `MonitorConfig` object. It sets default configuration values and then attempts to load and apply settings from a YAML configuration file if it exists at the specified path.  
  
- \*\*Parameters:\*\*  
 - `config\_path` (str, optional): The path to the YAML configuration file. Defaults to `'.monitor\_config.yaml'` in the current working directory.  
  
- \*\*Returns:\*\*  
 - None.  
  
#### Code Usage Example  
  
The configuration is typically managed through a `.monitor\_config.yaml` file in the root of your project.  
  
\*\*Example `.monitor\_config.yaml`:\*\*  
  
```yaml  
# Patterns to ignore (supports glob patterns)  
ignore\_patterns:  
 - "\*\*/\_\_pycache\_\_/\*\*"  
 - "\*.tmp"  
 - "venv/"  
 - ".git/"  
  
# File extensions to check  
file\_extensions\_to\_check:  
 - ".py"  
 - ".md"  
 - ".yaml"

**Python Usage:**

The module automatically creates a global CONFIG instance that can be imported and used across the application.

# In another file, e.g., main.py  
from code\_monitor.config import CONFIG  
  
def main():  
 # Access the loaded configuration  
 print("Files to ignore:", CONFIG.ignore\_patterns)  
 print("Extensions to check:", CONFIG.file\_extensions)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
  
# Expected output (if .monitor\_config.yaml exists as above):  
# Files to ignore: ['\*\*/\_\_pycache\_\_/\*\*', '\*.tmp', 'venv/', '.git/']  
# Extensions to check: ['.py', '.md', '.yaml']

If .monitor\_config.yaml does not exist, CONFIG will hold the default values:

# Expected output (if .monitor\_config.yaml does not exist):  
# Files to ignore: []  
# Extensions to check: ['.py']

```