logging模块JSON配置文件

JSON配置文件

这个JSON配置文件定义了一个logging模块的配置，其中包含了三个部分：formatters、handlers和loggers。

1. formatters部分定义了一个名为simple的格式化器，格式化日志消息包含时间、记录器名称、日志级别和消息内容。
2. handlers部分定义了三个处理器：console、info_file_handler和error_file_handler。console处理器将日志消息输出到标准输出流，info_file_handler处理器将INFO级别的日志消息写入到名为info.log的文件中，error_file_handler处理器将ERROR级别的日志消息写入到名为errors.log的文件中。
3. loggers部分定义了一个名为my_module的记录器，设置了记录器的级别为ERROR，指定了将ERROR级别的日志消息交给info_file_handler处理器处理，并设置了不传播（propagate）日志消息给父记录器。
4. root部分定义了根记录器的配置，将INFO级别的日志消息同时交给console、info_file_handler和error_file_handler处理器处理。通过这个配置文件，可以实现不同级别的日志消息输出到不同的地方，方便进行日志管理和调试。

这段Python代码用于加载和配置logging模块的设置，通过一个JSON配置文件来定义日志的格式、处理方式和记录器行为。让我们逐步解释这段代码的功能：

加载和配置logging模块

1. **setup_logging函数**：
   - 这个函数用于设置logging模块的配置。
   - 默认情况下，它会尝试加载名为"logging.json"的配置文件。可以通过设置环境变量LOG_CFG来指定其他配置文件的路径。
   - 如果环境变量LOG_CFG被设置了，将会使用其指定的路径作为配置文件的路径。
   - 如果配置文件存在（通过os.path.exists(path)判断），则会打开并读取这个JSON文件。
   - 使用json.load(f)加载配置文件内容，并将其作为参数传递给logging.config.dictConfig(config)函数，这个函数会根据配置来配置logging模块。
   - 如果配置文件不存在，则使用logging.basicConfig进行基本的配置，将日志级别设置为default_level（默认为logging.INFO）。
2. **func函数**：
   - 这个函数用于演示日志的使用。
   - 在func函数中，通过logging.info输出三条日志消息，分别是"start func"、"exec func"和"end func"。
3. **主程序部分**：
   - 在主程序中，首先调用setup_logging函数，指定了默认的配置文件路径为"logging.json"。
   - 然后调用func函数，这样在func函数中的日志输出就会受到配置文件中定义的影响。

   总结功能：
   - 通过这段代码，可以灵活地配置logging模块，使日志的格式化、输出位置以及日志级别能够根据JSON配置文件的定义而变化。
   - 这种方式使得日志配置更加灵活和易于维护，特别适用于大型项目或者需要不同配置的部署环境。
4. 设置环境变量LOG_CFG来指定其他配置文件的路径
   在Python中设置环境变量可以通过 `os.environ` 字典来实现。下面是一个示例代码，演示如何设置环境变量 `LOG_CFG` 来指定其他配置文件的路径：

```python
import os

# 设置环境变量LOG_CFG为指定的配置文件路径
os.environ['LOG_CFG'] = '/path/to/other_logging.json'

# 读取环境变量LOG_CFG的值
log_cfg_path = os.getenv('LOG_CFG')

print(f"LOG_CFG环境变量的值为：{log_cfg_path}")
```

在这个示例中，我们首先使用 `os.environ['LOG_CFG'] = '/path/to/other_logging.json'` 来设置环境变量 `LOG_CFG` 的值为指定的配置文件路径 `/path/to/other_logging.json` 。然后使用 `os.getenv('LOG_CFG')` 来获取环境变量 `LOG_CFG` 的值，并将其打印出来。
通过这种方式，您可以在Python代码中设置环境变量来指定其他配置文件的路径，以便在日志配置中使用。