Environment Variables Guide - Docker & Python

Common Environment Variables Explained

1. DEBIAN_FRONTEND=noninteractive

What it does: Prevents interactive prompts during package installations in Debian/Ubuntu systems.

Why use it: Essential for automated scripts, Docker builds, and CI/CD pipelines where no human is available to answer prompts.

Example:

```bash
# Without this - installation might hang waiting for user input
apt-get install nginx

# With this - installation proceeds automatically with defaults
ENV DEBIAN_FRONTEND=noninteractive
apt-get install nginx
```

Real-world usage:

```dockerfile
# In Dockerfile
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update && apt-get install -y \
  python3 \
  python3-pip \
  && rm -rf /var/lib/apt/lists/*
```

---

2. PYTHONUNBUFFERED=1

What it does: Forces Python to send output directly to terminal instead of storing it in memory buffer.

Why use it: Get real-time output visibility, especially important for logs and debugging.

Testing the Difference

Create `test_buffer.py`:

```python
import time
import sys

for i in range(5):
    print(f"Step {i}", end="", flush=False)
    sys.stdout.write(".")
    time.sleep(1)
print("\nDone!")
```

Test commands:

```bash
# Buffered (default) - might see delayed output
python3 test_buffer.py

# Unbuffered - see output immediately
PYTHONUNBUFFERED=1 python3 test_buffer.py

# Test with file redirection to see clearer difference
python3 test_buffer.py > output.txt & tail -f output.txt
PYTHONUNBUFFERED=1 python3 test_buffer.py > output2.txt & tail -f output2.txt
```

When it matters most:

- Docker containers and logging
- Long-running processes
- Progress indicators
- Debugging production applications

---

3. PYTHONDONTWRITEBYTECODE=1

What it does: Prevents Python from creating `.pyc` (bytecode) files.

Why use it: Keeps filesystem clean, prevents permission issues, reduces container size.

Example comparison:

```bash
```

```
# Default behavior - creates .pyc files
python3 -c "import os; print('hello')"
# Check: ls __pycache__/ (you'll see .pyc files)

# With PYTHONDONTWRITEBYTECODE=1 - no .pyc files
PYTHONDONTWRITEBYTECODE=1 python3 -c "import os; print('hello')"
# Check: ls __pycache__/ (directory might not even exist)
```

Common Usage Patterns

Docker Containers

dockerfile

```
ENV DEBIAN_FRONTEND=noninteractive \
    PYTHONUNBUFFERED=1 \
    PYTHONDONTWRITEBYTECODE=1

RUN apt-get update && apt-get install -y python3
COPY app.py .
CMD ["python3", "app.py"]
```

Shell Scripts

bash

```
#!/bin/bash
export DEBIAN_FRONTEND=noninteractive
export PYTHONUNBUFFERED=1
export PYTHONDONTWRITEBYTECODE=1

# Your commands here
python3 long_running_script.py
```

One-time Command

bash

```
DEBIAN_FRONTEND=noninteractive PYTHONUNBUFFERED=1 PYTHONDONTWRITEBYTECODE=1 python3 my_scrip
```

Quick Reference

Variable	Purpose	When to Use
`DEBIAN_FRONTEND=noninteractive`	No installation prompts	Docker, CI/CD, automated scripts
`PYTHONUNBUFFERED=1`	Real-time output	Containers, logging, debugging
`PYTHONDONTWRITEBYTECODE=1`	No .pyc files	Containers, clean environments

Testing Your Understanding

Try these commands to see the differences:

```bash
# Test 1: See buffering in action
echo 'import time; [print(f"Step {i}", end=".", flush=False) or time.sleep(1) for i in range(3)]' > test.py
python3 test.py              # Default
PYTHONUNBUFFERED=1 python3 test.py # Unbuffered

# Test 2: Check .pyc file creation
python3 -c "import json"
ls __pycache__/              # See .pyc files

# Clean and test without .pyc
rm -rf __pycache__
PYTHONDONTWRITEBYTECODE=1 python3 -c "import json"
ls __pycache__/              # No .pyc files

# Test 3: Package installation (Ubuntu/Debian only)
# sudo apt-get install tree      # Might prompt
# sudo DEBIAN_FRONTEND=noninteractive apt-get install tree  # No prompts
```

Pro Tips

1. **Always use these three together** in Docker containers for best results
2. **PYTHONUNBUFFERED** is crucial for Docker logging and monitoring
3. **Use in CI/CD pipelines** to prevent hanging builds
4. **Test your scripts** both with and without these variables to understand the impact