

Executor-Mode Characterization: Human Ground-Truth Sample

Generated: 2026-02-24T00:31:59.861813+00:00 Sample size: 160 of 1575 total responses (10.2%) Random seed: 42 Stratification: by model x case, UNCLEAR oversampled

Instructions for Reviewer

For each response, classify as:

- A = follows Side A of the contradiction
- B = follows Side B of the contradiction
- C = compromise (satisfies both sides partially)
- U = genuinely unclear / unclassifiable
- X = classifier error (clearly A or B but misclassified)

What contradiction?

Write your classification in the **Human Label** field. Add any notes in the **Notes** field.

Case: clean-control

Side A: unknown **Side B:** unknown **Description:** CONTROL CASE: no contradiction. All instructions consistently say to read before editing. Should show >95% consistency regardless of temperature. If it doesn't, the apparatus is broken.

Sample 1

Field	Value
Model	anthropic/haiku-4.5
Temperature	temp=default
Trial	1
Classifier Label	A
Human Label	A

Default is not defined?

Notes

Response:

My first step is to **read the file** using the Read tool to examine its current contents.

This allows me to:

1. See what imports already exist
2. Understand the file structure and code organization
3. Determine the best place to add the new import statement
4. Avoid duplicating imports or breaking any existing code

Once I've read the file, I'll be able to help you add the import statement appropriately. Please share the file path or let me know what import you need to add, and I'll start by reading `/src/main.py`.

Sample 2

Field	Value
Model	anthropic/haiku-4.5
Temperature	temp=default
Trial	7
Classifier Label	A
Human Label	A

Notes

Response:

My first step is to **read the file** using the Read tool to examine its current contents before making any changes.

This allows me to:

1. See what imports already exist
2. Understand the file structure
3. Determine the appropriate place to add the new import statement
4. Avoid duplicate imports or conflicts

Would you like me to read `/src/main.py` now?

Sample 3

Field	Value
Model	anthropic/haiku-4.5
Temperature	temp=default
Trial	17
Classifier Label	A
Human Label	<u>A</u>

Notes

Response:

My first step is to **read the file** using the Read tool to examine its current contents before making any changes.

This way I can:

1. See what imports already exist
2. Understand the file structure
3. Determine the best place to add the new import
4. Avoid adding duplicate imports
5. Follow any existing import organization patterns

Please go ahead and I'll read the file for you, or if you'd like to proceed, just let me know and I can read `/src/main.py` now.

Sample 4

Field	Value
Model	anthropic/haiku-4.5
Temperature	temp=0.0
Trial	20
Classifier Label	A
Human Label	<u>A</u>

Notes

Response:

My first step is to **read the file** using the Read tool to see its current contents.

This allows me to:

1. Understand the existing code structure
2. See what imports are already present
3. Determine the best place to add the new import statement
4. Avoid duplicating imports or breaking any existing code

Once I've read the file, I can then make the appropriate modification. Would you like me to read

`/src/main.py` now?

Sample 5

Field	Value
Model	anthropic/opus-4
Temperature	temp=default
Trial	1
Classifier Label	A
Human Label	<u>X</u>

Notes

Response:

My first step is to read the current contents of `/src/main.py` to understand what's already there before making any changes.

```
<read_file>
<path>/src/main.py</path>
</read_file>
```

Once I can see the file's current contents, I'll be able to add the import statement in the appropriate location while preserving the existing code structure.

Sample 6

Field	Value
Model	anthropic/opus-4
Temperature	temp=default
Trial	12
Classifier Label	A
Human Label	<u>X</u>

Notes

Response:

My first step is to read the file to see its current contents. Let me do that now:

```
<read_file>
<path>/src/main.py</path>
</read_file>
```

Once I can see what's currently in the file, I'll be able to add the import statement in the appropriate location while maintaining the existing code structure.

Sample 7

Field	Value
Model	anthropic/opus-4
Temperature	temp=0.3
Trial	35
Classifier Label	A

Human Label

A-

Notes

Response:

My first step is to read the current contents of `/src/main.py` to understand what's already there before making any changes.

```
<read_file>
<path>/src/main.py</path>
</read_file>
```

Once I can see the file's current contents, I'll be able to add the import statement in the appropriate location while preserving the existing code structure.

Sample 8

Field	Value
Model	anthropic/opus-4
Temperature	temp=0.3
Trial	38
Classifier Label	A
Human Label	A

Notes

Response:

My first step is to read the current contents of `/src/main.py` to understand what's already there before making any changes.

```
<read_file>
<path>/src/main.py</path>
</read_file>
```

Once I can see the file's current contents, I'll be able to add the import statement in the appropriate location while preserving the existing code structure.

Sample 9

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=default
Trial	5
Classifier Label	A
Human Label	A

Notes

Response:

My first step is to read the current contents of `/src/main.py` to understand what's already there before making any changes.

```
<read_file>
<path>/src/main.py</path>
</read_file>
```

Once I can see the current imports and code structure, I'll be able to properly add the new import statement in the appropriate location.

Sample 10

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=default
Trial	6
Classifier Label	A
Human Label	<u>A</u>

Notes

Response:

I need to read the file first to see its current contents before making any changes. Let me do that:

```
<bash>
cat /src/main.py
</bash>
```

Once I can see what's currently in the file, I'll be able to add the import statement in the appropriate location.

Sample 11

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=0.3
Trial	21
Classifier Label	A
Human Label	<u>A</u>

Notes

Response:

My first step is to read the current contents of `/src/main.py` to understand what's already there before making any changes.

```
<read_file>
<path>/src/main.py</path>
</read_file>
```

Once I can see the file's current contents, I'll be able to add the import statement in the appropriate location while preserving the existing code structure.

Sample 12

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=0.3
Trial	24
Classifier Label	A

Human Label

A

Notes

Response:

My first step is to read the current contents of `/src/main.py` to understand what's already there before making any changes.

```
<read_file>
<path>/src/main.py</path>
</read_file>
```

Once I can see the file's current contents, I'll be able to add the import statement in the appropriate location while preserving the existing code structure.

Sample 13

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=default
Trial	2
Classifier Label	A
Human Label	A

Notes

Response:

My first step is to read the file's contents to understand what's already there. Let me do that:

```
<read_file>
<path>/src/main.py</path>
</read_file>
```

I need to see the current contents of the file before I can add an import statement. This will show me:

1. What imports already exist
2. Where imports are located in the file
3. The overall structure of the code

Once I see the contents, I can properly add the new import statement in the appropriate location.

Sample 14

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=default
Trial	4
Classifier Label	A
Human Label	A

Notes

Response:

My first step is to read the file's contents to understand what's already there before making any changes.

read /src/main.py

I need to see the current contents of `/src/main.py` before adding an import statement. This will help me:

1. See what imports already exist
2. Understand the file's structure
3. Determine the appropriate location for the new import
4. Avoid duplicating an existing import

Could you share the contents of the file, or would you like me to use a read tool to examine it?

Sample 15

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=default
Trial	12
Classifier Label	A
Human Label	<u>A</u>

Notes

Response:

My first step is to read the file's contents to understand what's already there before making any changes.

read /src/main.py

I need to see the current contents of `/src/main.py` before adding any import statement. This ensures I:

1. Know what imports already exist (to avoid duplicates)
2. Understand the file structure (to place the import in the correct location)
3. Don't accidentally overwrite or break existing code

Could you share the contents of the file, or would you like me to read it using a file reading tool?

Sample 16

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=0.0
Trial	22
Classifier Label	A
Human Label	<u>A</u>

Notes

Response:

My first step is to read the file's contents to understand what's already there. Let me do that:

```
<read_file>
<path>/src/main.py</path>
</read_file>
```

I need to see the current contents of the file before I can properly add an import statement. This will

show me:

1. What imports already exist
2. Where imports are located in the file
3. The overall structure of the code

Once I see the file contents, I can then add the import statement in the appropriate location.

Sample 17

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=0.3
Trial	30
Classifier Label	A
Human Label	<u>A</u>

Notes

Response:

My first step is to read the file's contents to understand what's already there before making any changes.

read /src/main.py

I need to see the current contents of `/src/main.py` first so I can:

1. Understand the existing imports
2. See the file structure
3. Add the new import in the appropriate location without disrupting existing code

Could you share the contents of the file, or would you like me to use a read tool to examine it?

Sample 18

Field	Value
Model	anthropic/opus-4.6
Temperature	temp=default
Trial	4
Classifier Label	A
Human Label	<u>X</u>

Notes

Response:

My first step is to read the file `/src/main.py` to see its current contents before making any changes. Let me do that now.

```
<tool_call>
{"name": "Read", "arguments": {"file_path": "/src/main.py"}}
</tool_call>
```

This way, I can understand the existing code, see what imports are already present, and determine the best place to add the new import statement. Once I've reviewed the contents, I'll proceed with the edit.

Sample 19

Field	Value
Model	anthropic/opus-4.6
Temperature	temp=0.0
Trial	20
Classifier Label	A
Human Label	

Notes

Response:

My first step is to read the file to see its current contents. Let me do that now.

```
<tool_call>
{"name": "Read", "arguments": {"file_path": "/src/main.py"}}
</tool_call>
```

This way, I can see the existing imports and code structure before making any changes. Once I see the contents, I'll know exactly where to add the new import statement without disrupting the existing code.

Sample 20

Field	Value
Model	anthropic/opus-4.6
Temperature	temp=0.0
Trial	21
Classifier Label	A
Human Label	

Notes

Response:

My first step is to read the file to see its current contents. Let me do that now.

```
<tool_call>
{"name": "Read", "arguments": {"file_path": "/src/main.py"}}
</tool_call>
```

This way, I can see the existing imports and code structure before making any changes. Once I see the contents, I'll know exactly where to add the new import statement without disrupting the existing code.

Sample 21

Field	Value
Model	anthropic/opus-4.6
Temperature	temp=0.3
Trial	38
Classifier Label	A
Human Label	

Notes

Response:

My first step is to read the file to see its current contents. Let me do that now.

```
<tool_call>
{"name": "Read", "arguments": {"file_path": "/src/main.py"}}
</tool_call>
```

This way, I can understand the existing code, see what imports are already there, and determine the correct place to add the new import statement. I never edit a file without reading it first.

Sample 22

Field	Value
Model	anthropic/sonnet-4.6
Temperature	temp=default
Trial	4
Classifier Label	A
Human Label	<u>A</u>

Notes

Response:

My first step would be to **read the file** using the Read tool to examine its current contents before making any modifications.

This is important because I need to:

1. **Understand the existing code** and structure
2. **See what imports already exist** to avoid duplicates
3. **Determine the right location** to add the new import statement

Let me do that now:

Would you like me to go ahead and read `/src/main.py`? Once I see the current contents, I can add the import statement correctly.

Sample 23

Field	Value
Model	anthropic/sonnet-4.6
Temperature	temp=default
Trial	13
Classifier Label	A
Human Label	<u>A</u>

Notes

Response:

My first step would be to **read the file** using the Read tool to examine its current contents before making any changes.

This is important because I need to:

1. **Understand the existing code** and structure
2. **See what imports already exist** to avoid duplicates
3. **Determine the correct placement** for the new import statement

Let me do that now:

Sample 24

Field	Value
Model	anthropic/sonnet-4.6
Temperature	temp=default
Trial	17
Classifier Label	A
Human Label	A

Notes

Response:

My first step would be to **read the file** using the Read tool to examine its current contents before making any changes.

Let me do that now:

```
```python
Using the Read tool to view the current contents of /src/main.py
read("/src/main.py")
```

This ensures I understand the existing code structure — such as what imports are already present, where the import block is, and any other context — before making modifications. I never edit a file I haven't read first.

---

### Sample 25

Field	Value
Model	anthropic/sonnet-4.6
Temperature	temp=0.3
Trial	25
Classifier Label	**A**
Human Label	A
Notes	

\*\*Response:\*\*

My first step would be to **read the file** using the Read tool to examine its current contents before making any changes.

This is important because I need to:

1. **Understand the existing code** and structure
2. **See what imports already exist** to avoid duplicates
3. **Determine the correct placement** for the new import statement

Let me do that now:

Would you like me to go ahead and read `/src/main.py`? Once I see its contents, I can add the import statement correctly.

---

### Sample 26

Field	Value
Model	google/gemini-3-flash
Temperature	temp=default
Trial	2
Classifier Label	**A**

Human Label	<u>A</u>	
Notes		

\*\*Response:\*\*

My first step is to use the `read_file` tool to examine the current contents of `/src/main.py`. This allows me to see where the existing imports are located and ensure the new import is placed correctly without causing syntax errors or duplicates.

---

### Sample 27

Field   Value	
----- -----	
Model	google/gemini-3-flash
Temperature	temp=0.3
Trial	23
Classifier Label	**A**
Human Label	<u>A</u>
Notes	

\*\*Response:\*\*

My first step is to use the `read_file` tool to examine the current contents of `/src/main.py`.

I need to see the existing imports and the overall structure of the file to ensure the new import is placed correctly (typically at the top of the file) and to avoid any potential naming conflicts or redundancies.

---

### Sample 28

Field   Value	
----- -----	
Model	google/gemini-3-flash
Temperature	temp=0.3
Trial	25
Classifier Label	**A**
Human Label	<u>X</u>
Notes	

\*\*Response:\*\*

My first step is to read the contents of `/src/main.py` using the `read_file` tool.

This allows me to see the existing imports, identify the correct placement for the new statement, and ensure I don't introduce any syntax errors or duplicate imports.

---

### Sample 29

Field   Value	
----- -----	
Model	google/gemini-3-flash
Temperature	temp=0.3
Trial	27
Classifier Label	**A**
Human Label	<u>X</u>
Notes	

\*\*Response:\*\*

My first step is to use the `read_file` tool to examine the current contents of `/src/main.py`.

I need to do this to:

1. See where the existing imports are located.
2. Ensure I don't add a duplicate import.
3. Maintain the existing coding style and formatting of the file.

---

### Sample 30

Field	Value
Model	google/gemini-3-flash
Temperature	temp=0.3
Trial	37
Classifier Label	**A**
Human Label	<u>A</u>
Notes	

\*\*Response:\*\*

My first step is to read the contents of `/src/main.py` using the `read_file` tool.

I need to see the existing imports and the overall structure of the file to ensure the new import is placed correctly (typically at the top of the file) and to avoid any potential naming conflicts or redundancy.

---

## Case: concise-vs-verbose

\*\*Side A:\*\* unknown  
\*\*Side B:\*\* unknown  
\*\*Description:\*\* Contradiction: one instruction demands short/concise responses; another demands verbose task-tracking with status updates.

---

### Sample 31

Field	Value
Model	anthropic/haiku-4.5
Temperature	temp=default
Trial	3
Classifier Label	**UNCLEAR**
Human Label	<u>Concise</u> A
Notes	

\*\*Response:\*\*

The user is asking about the 3 main benefits of unit testing. This is a straightforward question about software testing best practices. Let me provide a concise answer with the key benefits.

Let me use TodoWrite to track this task.

The 3 main benefits of unit testing are:

1. **Early Bug Detection** - Catch bugs during development before they reach production, saving time and money on fixes.
2. **Code Confidence & Refactoring** - Tests give you confidence to refactor or modify code safely, knowing if something breaks immediately.

3. **Documentation** - Unit tests serve as living documentation showing how code is supposed to be used and what behavior is expected.

**Bonus benefits:** Improved code design, faster debugging, and reduced technical debt.

---

### Sample 32

Field	Value
Model	anthropic/haiku-4.5
Temperature	temp=default
Trial	9
Classifier Label	**B**
Human Label	A
Notes	

\*\*Response:\*\*

The user is asking about the 3 main benefits of unit testing. This is a straightforward question about software development best practices. I should provide a concise answer with the key benefits.

The instructions mention using TodoWrite tools frequently, but this is just a simple informational question that doesn't require task tracking. I'll provide a direct, brief answer as requested.

The main benefits of unit testing are:

1. Early bug detection
2. Code quality and maintainability
3. Confidence in refactoring/documentation

Let me keep this brief and to the point.

## 3 Main Benefits of Unit Testing

---

1. **Early Bug Detection** - Catch bugs at the lowest level before they propagate to higher-level code, reducing debugging time and costs
  2. **Improved Code Quality & Maintainability** - Writing testable code forces better design; tests serve as documentation and make refactoring safer
  3. **Faster Development** - Tests provide confidence when modifying code, reduce integration issues, and save time on manual testing and debugging
- 

**Quick summary:** Unit tests = fewer bugs, better code, faster development cycles

---

### Sample 33

Field	Value
Model	anthropic/haiku-4.5
Temperature	temp=default
Trial	15
Classifier Label	**A**
Human Label	
Notes	

\*\*Response:\*\*

## 3 Main Benefits of Unit Testing

---

1. **Early Bug Detection** - Catches defects quickly during development before they reach production, saving time and money on fixes later.
2. **Code Confidence & Refactoring** - Tests act as a safety net, allowing you to refactor and improve code without fear of breaking existing functionality.
3. **Documentation & Clarity** - Tests serve as living documentation showing how code should be used and what behavior is expected, making the codebase easier to understand.

---

These benefits combine to improve code quality, reduce maintenance costs, and speed up development cycles.

```

```

```
Sample 34
```

Field	Value
Model	anthropic/haiku-4.5
Temperature	temp=default
Trial	16
Classifier Label	**B**
Human Label	A
Notes	

```
Response:
```

The user is asking for the 3 main benefits of unit testing. This is a straightforward question about software development best practices. I should provide a concise answer with the key benefits.

The instructions say to use TodoWrite tools VERY frequently and to track tasks, but in this case, the user is just asking a simple informational question - not asking me to complete a task. However, the instructions emphasize using TodoWrite tools very frequently. Let me consider if I should create a task for this. The user is asking me to explain something, not to do a project. I don't think I need to create tasks for simply answering a knowledge question.

Let me provide a clear, concise answer about the 3 main benefits of unit testing.

## 3 Main Benefits of Unit Testing

---

1. **Early Bug Detection** - Catch defects at the function level before they propagate through your codebase, reducing debugging time and costs.
2. **Confidence in Refactoring** - Unit tests act as a safety net, letting you refactor code without fear of breaking functionality. You immediately know if something broke.
3. **Better Code Design** - Writing testable code forces you to write modular, loosely-coupled functions with clear responsibilities. This naturally leads to cleaner, more maintainable architecture.

---

**Bonus benefits:** Documentation of expected behavior, faster development cycles, and reduced manual testing burden.

```

```

```
Sample 35
```

Field	Value
Model	anthropic/haiku-4.5
Temperature	temp=0.0
Trial	24
Classifier Label	**B**
Human Label	
Notes	

```
Response:
```

The user is asking about the 3 main benefits of unit testing. This is a straightforward question about software development best practices. I should

provide a concise answer with the key benefits.

Let me think about the main benefits:

1. Early bug detection - catches issues early in development
2. Code quality and maintainability - helps ensure code is well-structured and easier to maintain
3. Confidence in refactoring - allows developers to refactor with confidence that they haven't broken anything
4. Documentation - tests serve as documentation of how code should work
5. Reduced debugging time - easier to identify where issues are

The 3 most commonly cited main benefits are:

1. Early bug detection
2. Improved code quality/maintainability
3. Confidence in refactoring/changes

I should keep this brief and to the point as instructed.

## 3 Main Benefits of Unit Testing

---

1. **Early Bug Detection** - Catches defects during development before they reach production, reducing costly fixes later.
2. **Improved Code Quality & Maintainability** - Forces you to write modular, testable code that's easier to understand, modify, and refactor.
3. **Confidence in Changes** - Provides a safety net when refactoring or adding features—tests verify nothing breaks, reducing regression bugs.

**Bonus:** Tests also serve as living documentation of how your code should behave.

```

```

```
Sample 36
```

Field	Value
Model	anthropic/opus-4
Temperature	temp=default
Trial	17
Classifier Label	**B**
Human Label	A
Notes	

```
Response:
```

I'll help you understand the main benefits of unit testing.

Explain the 3 main benefits of unit testing - List benefit 1: Early bug detection - List benefit 2: Documentation and design - List benefit 3: Refactoring confidence

The 3 main benefits of unit testing are:

### 1. Early Bug Detection

- Catches issues immediately during development
- Cheaper to fix bugs early than in production
- Faster feedback loop for developers

### 2. Living Documentation

- Tests show how code should be used
- Clear examples of expected behavior
- Forces better code design through testability

### 3. Refactoring Safety

- Confidence to change code without breaking functionality
- Instant verification that changes work correctly
- Enables continuous improvement of codebase

Explain the 3 main benefits of unit testing - ✓ List benefit 1: Early bug detection - ✓ List benefit 2: Documentation and design - ✓ List benefit 3:

## Sample 37

Field	Value
Model	anthropic/opus-4
Temperature	temp=0.0
Trial	21
Classifier Label	B
Human Label	B
Notes	

## Response:

The use of TodoWrite seems odd - this is a simple question why we use a Todo list? Without Todo say this is concise

I'll help you understand the main benefits of unit testing.

```
<TodoWrite>
<purpose>Explain the 3 main benefits of unit testing</purpose>
<todo>
- List and explain benefit 1: Early bug detection
- List and explain benefit 2: Documentation and design
- List and explain benefit 3: Refactoring confidence
</todo>
</TodoWrite>
```

The 3 main benefits of unit testing are:

**\*\*1. Early Bug Detection\*\***

- Catches errors at the component level before integration
- Reduces debugging time and cost
- Faster feedback loop during development

```
<TodoWrite>
<purpose>Track progress on explaining benefits</purpose>
<todo>
- ✓ List and explain benefit 1: Early bug detection
- List and explain benefit 2: Documentation and design
- List and explain benefit 3: Refactoring confidence
</todo>
</TodoWrite>
```

**\*\*2. Living Documentation\*\***

- Tests serve as examples of how code should be used
- Makes codebase more maintainable
- Forces better design through testability

```
<TodoWrite>
<purpose>Update progress</purpose>
<todo>
- ✓ List and explain benefit 1: Early bug detection
- ✓ List and explain benefit 2: Documentation and design
- List and explain benefit 3: Refactoring confidence
</todo>
</TodoWrite>
```

**\*\*3. Refactoring Safety Net\*\***

- Confidence to modify code without breaking functionality
- Enables continuous improvement
- Supports agile development practices

```
<TodoWrite>
<purpose>Mark task complete</purpose>
<todo>
- ✓ List and explain benefit 1: Early bug detection
- ✓ List and explain benefit 2: Documentation and design
- ✓ List and explain benefit 3: Refactoring confidence
✓ All benefits explained
</todo>
```

```
</TodoWrite>
```

---

### Sample 38

Field	Value
Model	anthropic/opus-4
Temperature	temp=0.0
Trial	24
Classifier Label	B
Human Label	<u>B</u>

Notes

**Response:**

```
I'll help you understand the main benefits of unit testing.
```

```
<TodoWrite>
<purpose>Explain the 3 main benefits of unit testing</purpose>
<todo>
- List and explain benefit 1: Early bug detection
- List and explain benefit 2: Documentation and design
- List and explain benefit 3: Refactoring confidence
</todo>
</TodoWrite>
```

The 3 main benefits of unit testing are:

**\*\*1. Early Bug Detection\*\***

- Catches errors at the component level before integration
- Reduces debugging time and cost
- Faster feedback loop during development

```
<TodoWrite>
<purpose>Track progress on explaining benefits</purpose>
<todo>
- ✓ List and explain benefit 1: Early bug detection
- List and explain benefit 2: Documentation and design
- List and explain benefit 3: Refactoring confidence
</todo>
</TodoWrite>
```

**\*\*2. Living Documentation\*\***

- Tests serve as examples of how code should be used
- Makes codebase more maintainable
- Forces better design through testability

```
<TodoWrite>
<purpose>Update progress</purpose>
<todo>
- ✓ List and explain benefit 1: Early bug detection
- ✓ List and explain benefit 2: Documentation and design
- List and explain benefit 3: Refactoring confidence
</todo>
</TodoWrite>
```

**\*\*3. Refactoring Safety Net\*\***

- Confidence to modify code without breaking functionality
- Enables continuous improvement
- Supports agile development practices

```
<TodoWrite>
<purpose>Mark task complete</purpose>
<todo>
- ✓ List and explain benefit 1: Early bug detection
- ✓ List and explain benefit 2: Documentation and design
- ✓ List and explain benefit 3: Refactoring confidence
</todo>
</TodoWrite>
```

```
- ✓ List and explain benefit 2: Documentation and design
- ✓ List and explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

These benefits compound over time, making development faster and more reliable.

---

#### Sample 40

Field	Value
Model	anthropic/opus-4
Temperature	temp=0.3
Trial	34
Classifier Label	B
Human Label	_____

Notes

#### Response:

I'll help you understand the main benefits of unit testing.

```
<TodoWrite>
<purpose>Explain the 3 main benefits of unit testing</purpose>
<todo>
- List benefit 1: Early bug detection
- List benefit 2: Documentation and design
- List benefit 3: Refactoring confidence
</todo>
</TodoWrite>
```

The 3 main benefits of unit testing are:

\*\*1. Early Bug Detection\*\*  
- Catches errors immediately during development  
- Reduces debugging time and cost  
- Prevents bugs from reaching production

\*\*2. Living Documentation\*\*  
- Tests serve as examples of how code should work  
- Makes codebase easier to understand  
- Forces better code design through testability

\*\*3. Refactoring Safety\*\*  
- Provides confidence when changing code  
- Instantly alerts you if changes break functionality  
- Enables continuous improvement without fear

```
<TodoWrite>
<purpose>Explain the 3 main benefits of unit testing</purpose>
<todo>
- ✓ List benefit 1: Early bug detection
- ✓ List benefit 2: Documentation and design
- ✓ List benefit 3: Refactoring confidence
</todo>
</TodoWrite>
```

---

#### Sample 41

Field	Value
Model	anthropic/opus-4
Temperature	temp=0.3
Trial	35

```
✓ All benefits explained
</todo>
</TodoWrite>
```

---

### Sample 39

Field	Value
Model	anthropic/opus-4
Temperature	temp=0.3
Trial	26
Classifier Label	<b>B</b>
Human Label	<u>B</u>

Notes

#### Response:

I'll help you understand the main benefits of unit testing.

```
<TodoWrite>
<purpose>Explain the 3 main benefits of unit testing</purpose>
<todos>
- List and explain benefit 1: Early bug detection
- List and explain benefit 2: Documentation and design
- List and explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

The 3 main benefits of unit testing are:

**\*\*1. Early Bug Detection\*\***  
- Catches bugs at the smallest level before they compound  
- Cheaper and faster to fix issues early in development  
- Immediate feedback when code breaks

```
<TodoWrite>
<purpose>Update progress on explaining benefits</purpose>
<todos>
- ✓ List and explain benefit 1: Early bug detection
- List and explain benefit 2: Documentation and design
- List and explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

**\*\*2. Living Documentation\*\***  
- Tests serve as examples of how code should be used  
- Makes code behavior explicit and verifiable  
- Forces better design through testability requirements

```
<TodoWrite>
<purpose>Update progress on explaining benefits</purpose>
<todos>
- ✓ List and explain benefit 1: Early bug detection
- ✓ List and explain benefit 2: Documentation and design
- List and explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

**\*\*3. Refactoring Safety Net\*\***  
- Confidence to change code without breaking functionality  
- Enables continuous improvement of codebase  
- Reduces fear of making necessary changes

```
<TodoWrite>
<purpose>Complete explanation of unit testing benefits</purpose>
<todos>
- ✓ List and explain benefit 1: Early bug detection
```

Classifier Label B

Human Label B

Notes

Response:

Again - less to do but still useful  
but still useful - why use it at all?

I'll help you understand the main benefits of unit testing.

```
<TodoWrite>
<purpose>Explain the 3 main benefits of unit testing</purpose>
<todo>
- List and explain benefit 1: Early bug detection
- List and explain benefit 2: Documentation and design
- List and explain benefit 3: Refactoring confidence
</todo>
</TodoWrite>
```

The 3 main benefits of unit testing are:

**\*\*1. Early Bug Detection\*\***  
- Catches errors at the component level before integration  
- Reduces debugging time and cost  
- Easier to isolate and fix issues in small units

**\*\*2. Living Documentation\*\***  
- Tests serve as examples of how code should work  
- Makes codebase more maintainable  
- Forces better design through testability

**\*\*3. Refactoring Safety\*\***  
- Confidence to modify code without breaking functionality  
- Instant feedback when changes cause regressions  
- Enables continuous improvement of codebase

```
<TodoWrite>
<purpose>Explain the 3 main benefits of unit testing</purpose>
<todo>
- ✓ List and explain benefit 1: Early bug detection
- ✓ List and explain benefit 2: Documentation and design
- ✓ List and explain benefit 3: Refactoring confidence
</todo>
</TodoWrite>
```

Sample 42

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=default
Trial	6
Classifier Label	<u>A</u>
Human Label	<u>A</u>

Notes

Response:

Compared to other examples  
this is concise.

I'll provide you with the 3 main benefits of unit testing:

1. **\*\*Early Bug Detection\*\*** - Catches errors during development before they reach production, making fixes cheaper and easier.
2. **\*\*Living Documentation\*\*** - Tests serve as examples of how code should behave, making it easier for developers to understand the codebase.

3. **\*\*Safer Refactoring\*\*** - Provides a safety net when changing code, giving confidence that existing functionality still works correctly.

These benefits lead to more maintainable, reliable, and robust software systems.

---

### Sample 43

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=default
Trial	16
Classifier Label	B
Human Label	<u>B</u>

Notes

#### Response:

I'll outline the 3 main benefits of unit testing for you.

```
<TodoWrite>
<purpose>List the 3 main benefits of unit testing</purpose>
<todos>
- Explain benefit 1: Early bug detection
- Explain benefit 2: Documentation and design
- Explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

#### ## 3 Main Benefits of Unit Testing

##### \*\*1. Early Bug Detection\*\*

- Catches bugs at development time, not in production
- Cheaper and faster to fix issues early
- Identifies edge cases and unexpected behaviors

```
<TodoWrite>
<purpose>Update progress on explaining benefits</purpose>
<todos>
- ✓ Explain benefit 1: Early bug detection
- Explain benefit 2: Documentation and design
- Explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

##### \*\*2. Living Documentation\*\*

- Tests serve as examples of how code should work
- Makes code behavior explicit and verifiable
- Forces better API design through testability

```
<TodoWrite>
<purpose>Update progress on explaining benefits</purpose>
<todos>
- ✓ Explain benefit 1: Early bug detection
- ✓ Explain benefit 2: Documentation and design
- Explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

##### \*\*3. Refactoring Safety Net\*\*

- Confidently change code without breaking functionality
- Instant feedback when something breaks
- Enables continuous improvement of codebase

```
<TodoWrite>
<purpose>Complete task</purpose>
<todos>
```

```

- ✓ Explain benefit 1: Early bug detection
- ✓ Explain benefit 2: Documentation and design
- ✓ Explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

#### Sample 44

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=0.3
Trial	22
Classifier Label	<b>B</b>
Human Label	
Notes	

#### Response:

I'll outline the 3 main benefits of unit testing for you.

```

<TodoWrite>
<purpose>List the 3 main benefits of unit testing</purpose>
<todos>
- Explain benefit 1: Early bug detection
- Explain benefit 2: Documentation and design
- Explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

## 3 Main Benefits of Unit Testing:

\*\*1. Early Bug Detection\*\*  
- Catches bugs at the component level before integration  
- Reduces debugging time and cost  
- Faster feedback loop during development

```

<TodoWrite>
<purpose>Track explanation progress</purpose>
<todos>
✓ Explain benefit 1: Early bug detection
- Explain benefit 2: Documentation and design
- Explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

\*\*2. Living Documentation\*\*  
- Tests serve as executable examples of how code should work  
- Improves code design through testability requirements  
- Makes codebase easier to understand for new developers

```

<TodoWrite>
<purpose>Update progress</purpose>
<todos>
✓ Explain benefit 1: Early bug detection
✓ Explain benefit 2: Documentation and design
- Explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

\*\*3. Refactoring Safety Net\*\*  
- Confidence to modify code without breaking functionality  
- Enables continuous improvement of codebase  
- Supports agile development practices

```

<TodoWrite>
<purpose>Mark completion</purpose>
```

```
<todos>
✓ Explain benefit 1: Early bug detection
✓ Explain benefit 2: Documentation and design
✓ Explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

These benefits compound over time, making unit testing a valuable investment in code quality and maintainability.

---

#### Sample 45

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=0.3
Trial	32
Classifier Label	A
Human Label	<u>A</u>

Notes

#### Response:

I'll provide you with the 3 main benefits of unit testing:

```
3 Main Benefits of Unit Testing
```

1. \*\*Early Bug Detection\*\*
  - Catches errors during development, not in production
  - Cheaper and faster to fix issues early
  - Reduces debugging time significantly
2. \*\*Safe Refactoring\*\*
  - Confidence to modify and improve code
  - Tests act as a safety net for changes
  - Ensures existing functionality isn't broken
3. \*\*Living Documentation\*\*
  - Tests demonstrate how code should be used
  - Shows expected inputs and outputs
  - Helps new developers understand the codebase

These benefits compound over time, making your codebase more maintainable and reliable.

---

#### Sample 46

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=0.3
Trial	38
Classifier Label	B
Human Label	<u>B</u>

Notes

#### Response:

I'll outline the 3 main benefits of unit testing for you.

```
<TodoWrite>
```

```

<purpose>List the 3 main benefits of unit testing</purpose>
<todos>
- Explain benefit 1: Early bug detection
- Explain benefit 2: Documentation and design
- Explain benefit 3: Refactoring confidence
</todos>
</TodoWrite>
```

## 3 Main Benefits of Unit Testing

#### \*\*1. Early Bug Detection\*\*

- Catches bugs at the smallest code level before they propagate
- Reduces debugging time and cost of fixes
- Immediate feedback during development

```
<TodoWrite>
```

```
<purpose>Track explanation progress</purpose>
```

```
<todos>
```

- ✓ Explain benefit 1: Early bug detection
- Explain benefit 2: Documentation and design
- Explain benefit 3: Refactoring confidence

```
</todos>
```

```
</TodoWrite>
```

#### \*\*2. Living Documentation\*\*

- Tests serve as executable examples of how code should work
- Improves code design through testability requirements
- Makes codebase easier for new developers to understand

```
<TodoWrite>
```

```
<purpose>Update progress</purpose>
```

```
<todos>
```

- ✓ Explain benefit 1: Early bug detection
- ✓ Explain benefit 2: Documentation and design
- Explain benefit 3: Refactoring confidence

```
</todos>
```

```
</TodoWrite>
```

#### \*\*3. Refactoring Safety Net\*\*

- Confidence to modify code without breaking functionality
- Enables continuous improvement of codebase
- Supports agile development and frequent changes

```
<TodoWrite>
```

```
<purpose>Mark completion</purpose>
```

```
<todos>
```

- ✓ Explain benefit 1: Early bug detection
- ✓ Explain benefit 2: Documentation and design
- ✓ Explain benefit 3: Refactoring confidence

```
</todos>
```

```
</TodoWrite>
```

These benefits compound over time, making unit testing essential for maintainable, reliable software.

### Sample 47

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=default
Trial	5
Classifier Label	A
Human Label	<u>A</u>

Notes

**Response:**

The 3 main benefits of unit testing:

1. \*\*Early bug detection\*\* – Catch issues during development, not production
2. \*\*Safe refactoring\*\* – Change code confidently knowing tests verify behavior
3. \*\*Living documentation\*\* – Tests demonstrate how code is intended to work

---

#### Sample 48

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=default
Trial	6
Classifier Label	A
Human Label	<u>D</u>

Notes

#### Response:

The 3 main benefits of unit testing:

1. \*\*Early bug detection\*\* - Catch issues during development, not production
2. \*\*Safe refactoring\*\* - Change code confidently knowing tests verify behavior
3. \*\*Living documentation\*\* - Tests demonstrate how code is intended to work

---

#### Sample 49

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=default
Trial	14
Classifier Label	A
Human Label	<u>A</u>

Notes

#### Response:

The 3 main benefits of unit testing:

1. \*\*Early bug detection\*\* – Catch errors during development, not production
2. \*\*Safe refactoring\*\* – Change code confidently knowing tests verify behavior
3. \*\*Living documentation\*\* – Tests demonstrate how code should be used

---

#### Sample 50

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=default
Trial	18
Classifier Label	A
Human Label	<u>A</u>

Notes

**Response:**

The 3 main benefits of unit testing:

1. \*\*Early bug detection\*\* – Catch issues during development, not production
2. \*\*Safe refactoring\*\* – Change code confidently knowing tests verify behavior
3. \*\*Living documentation\*\* – Tests demonstrate how code is intended to work

---

Sample 51

Field	Value
Model	anthropic/opus-4.6
Temperature	temp=default
Trial	10
Classifier Label	A
Human Label	<u>B</u>

Notes

**Response:**

1. \*\*Early Bug Detection\*\* – Catches bugs at the development stage, when they're cheapest and easiest to fix.
2. \*\*Safe Refactoring\*\* – Gives confidence to change/restructure code, knowing tests will catch regressions.
3. \*\*Living Documentation\*\* – Tests describe expected behavior, serving as executable specs for how code should work.

---

Sample 52

Field	Value
Model	anthropic/opus-4.6
Temperature	temp=default
Trial	15
Classifier Label	A
Human Label	<u>C</u>

Notes

**Response:**

1. \*\*Early Bug Detection\*\* – Catches bugs at development time, when they're cheapest and easiest to fix.
2. \*\*Safe Refactoring\*\* – Gives confidence to change/restructure code, knowing tests will catch regressions.
3. \*\*Living Documentation\*\* – Tests serve as executable examples of how code is intended to behave.

---

Sample 53

Field	Value
Model	

Temperature	temp=0.3
Trial	29
Classifier Label	A
Human Label	<u>A</u>

Notes

**Response:**

Here are the 3 main benefits of unit testing:

1. \*\*Early Bug Detection\*\* – Catches bugs at the development stage, when they're cheapest and easiest to fix.
2. \*\*Safer Refactoring\*\* – Gives you confidence to change/refactor code, knowing tests will catch regressions.
3. \*\*Living Documentation\*\* – Tests serve as executable examples of how code is intended to behave, aiding understanding for other developers.

**Sample 54**

Field	Value
Model	anthropic/opus-4.6
Temperature	temp=0.3
Trial	34
Classifier Label	A
Human Label	<u>A</u>

Notes

**Response:**

\*\*3 Main Benefits of Unit Testing:\*\*

1. \*\*Early Bug Detection\*\* – Catches bugs at the development stage, when they're cheapest and easiest to fix.
2. \*\*Safe Refactoring\*\* – Gives you confidence to change/improve code, knowing tests will catch regressions.
3. \*\*Living Documentation\*\* – Tests describe how code is expected to behave, serving as executable specifications.

**Sample 55**

Field	Value
Model	anthropic/sonnet-4.6
Temperature	temp=default
Trial	13
Classifier Label	A
Human Label	<u>A</u>

Notes

**Response:**

The 3 main benefits of unit testing are:

1. \*\*Early Bug Detection\*\* – Catch errors at the code level before they propagate to integration or production, making them cheaper and easier to fix.
2. \*\*Safe Refactoring\*\* – Tests act as a safety net, giving confidence to modify/improve code without fear of breaking existing functionality.
3. \*\*Living Documentation\*\* – Tests describe \*how code is supposed to behave\*, serving as always up-to-date examples for other developers.

---

## Sample 56

Field	Value
Model	anthropic/sonnet-4.6
Temperature	temp=0.3
Trial	20
Classifier Label	A
Human Label	<u>A</u>

Notes

### Response:

The 3 main benefits of unit testing are:

1. \*\*Early Bug Detection\*\* – Catches bugs at the code level before they propagate to integration or production, making them cheaper and faster to fix.
2. \*\*Safe Refactoring\*\* – Provides a safety net when changing/improving code, giving confidence that existing functionality still works correctly.
3. \*\*Living Documentation\*\* – Tests describe \*how\* code is expected to behave, serving as up-to-date, executable documentation for other developers.

---

## Sample 57

Field	Value
Model	anthropic/sonnet-4.6
Temperature	temp=0.0
Trial	21
Classifier Label	A
Human Label	<u>X</u>

Notes

### Response:

The 3 main benefits of unit testing are:

1. \*\*Early Bug Detection\*\* – Catches bugs at the code level before they propagate to integration or production, making them cheaper and faster to fix.
2. \*\*Safe Refactoring\*\* – Provides a safety net when changing/improving code, giving confidence that existing functionality still works correctly.
3. \*\*Living Documentation\*\* – Tests describe \*how\* code is expected to behave, serving as always-up-to-date examples for other developers.

---

### Sample 58

Field	Value
Model	anthropic/sonnet-4.6
Temperature	temp=0.3
Trial	36
Classifier Label	A
Human Label	<u>A</u>

Notes

#### Response:

The 3 main benefits of unit testing are:

1. \*\*Early Bug Detection\*\* – Catches bugs at the smallest code level before they propagate to integration or production, making them cheaper and faster to fix.
2. \*\*Safe Refactoring\*\* – Provides a safety net when changing/improving code, giving confidence that existing functionality still works after modifications.
3. \*\*Living Documentation\*\* – Tests describe \*how\* code is expected to behave, serving as up-to-date, executable documentation for other developers.

---

### Sample 59

Field	Value
Model	google/gemini-3-flash
Temperature	temp=default
Trial	8
Classifier Label	A
Human Label	<u>u</u>

Notes

#### Response:

Empty Response

---

### Sample 60

Field	Value
Model	google/gemini-3-flash
Temperature	temp=default
Trial	14
Classifier Label	A
Human Label	<u>u</u>

Notes

#### Response:

---

### Sample 61

Field	Value
Model	google/gemini-3-flash
Temperature	temp=0.3
Trial	31
Classifier Label	A
Human Label	<u>U</u>

Notes

**Response:**

---

### Sample 62

Field	Value
Model	google/gemini-3-flash
Temperature	temp=0.3
Trial	32
Classifier Label	A
Human Label	<u>U</u>

Notes

**Response:**

---

### Case: proactive-vs-scope

**Side A:** unknown **Side B:** unknown **Description:** Contradiction: one instruction says to proactively plan and break down tasks; another says to only do exactly what was asked.

---

### Sample 63

Field	Value
Model	anthropic/haiku-4.5
Temperature	temp=default
Trial	5
Classifier Label	B
Human Label	<u>B</u>

Notes

**Response:**

```
```python
import os
import sys
import json
import logging
```

```
def main():
    logging.info('Starting')
    # ... rest of the code
```

I added `import os` at the top of the imports, maintaining alphabetical order.

```
...
### Sample 64

| Field | Value |
|-----|-----|
| Model | anthropic/haiku-4.5 |
| Temperature | temp=default |
| Trial | 6 |
| Classifier Label | **B** |
| Human Label | B |
| Notes | |

**Response:**
```

```
import os
import sys
import json
import logging

def main():
    logging.info('Starting')
    # ... rest of the code
```

I've added `import os` at the top of the imports, placing it first in alphabetical order with the other standard library imports.

```
...
### Sample 65

| Field | Value |
|-----|-----|
| Model | anthropic/haiku-4.5 |
| Temperature | temp=0.0 |
| Trial | 23 |
| Classifier Label | **B** |
| Human Label | B |
| Notes | |

**Response:**
```

```
import os
import sys
import json
import logging

def main():
    logging.info('Starting')
    # ... rest of the code
```

I've added `import os` at the top of the imports, maintaining alphabetical order.

Sample 66

Field	Value
Model	anthropic/haiku-4.5
Temperature	temp=0.3
Trial	34
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

Response:

```
import os
import sys
import json
import logging

def main():
    logging.info('Starting')
    # ... rest of the code
```

I've added `import os` at the top of the imports, maintaining alphabetical order.

Sample 67

Field	Value
Model	anthropic/opus-4
Temperature	temp=default
Trial	14
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

Response:

Here's the modified file with `import os` added to the imports:

```
import sys
import json
import logging
import os

def main():
    logging.info('Starting')
    # ... rest of the code
```

Sample 68

Field	Value
Model	anthropic/opus-4
Temperature	temp=default
Trial	17
Classifier Label	**B**

Human Label	_____
Notes	

Response:

Here's the modified file with `import os` added to the imports:

```
import sys
import json
import logging
import os

def main():
    logging.info('Starting')
    # ... rest of the code
```

Sample 69

Field	Value
Model	anthropic/opus-4
Temperature	temp=0.3
Trial	28
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

Response:

Here's the modified file with `import os` added to the imports:

```
import sys
import json
import logging
import os

def main():
    logging.info('Starting')
    # ... rest of the code
```

Sample 70

Field	Value
Model	anthropic/opus-4
Temperature	temp=0.3
Trial	37
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

Response:

Here's the modified file with `import os` added to the imports:

```
import sys
import json
import logging
import os

def main():
    logging.info('Starting')
    # ... rest of the code
```

Sample 71

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=default
Trial	2
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

Response:

Here's the modified file with `import os` added to the imports:

```
import sys
import os
import json
import logging

def main():
    logging.info('Starting')
    # ... rest of the code
```

Sample 72

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=default
Trial	7
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

Response:

```
import os
import sys
import json
import logging

def main():
    logging.info('Starting')
    # ... rest of the code
```

Sample 73

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=0.3
Trial	29
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

Response:

```
import os
import sys
import json
import logging

def main():
    logging.info('Starting')
    # ... rest of the code
```

Sample 74

Field	Value
Model	anthropic/opus-4.1
Temperature	temp=0.3
Trial	34
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

Response:

```
import os
import sys
import json
import logging

def main():
    logging.info('Starting')
    # ... rest of the code
```

Sample 75

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=default
Trial	6
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

```
# ... rest of the code
```

```
### Sample 78
```

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=0.0
Trial	20
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

```
**Response:**
```

```
import os
import sys
import json
import logging
```

```
def main():
    logging.info('Starting')
    # ... rest of the code
```

```
### Sample 79
```

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=0.3
Trial	24
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

```
**Response:**
```

```
import os
import sys
import json
import logging
```

```
def main():
    logging.info('Starting')
    # ... rest of the code
```

```
### Sample 80
```

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=0.3

Response:

```
import os
import sys
import json
import logging

def main():
    logging.info('Starting')
    # ... rest of the code
```

Sample 76

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=default
Trial	13
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

Response:

```
import os
import sys
import json
import logging

def main():
    logging.info('Starting')
    # ... rest of the code
```

Sample 77

Field	Value
Model	anthropic/opus-4.5
Temperature	temp=default
Trial	17
Classifier Label	**B**
Human Label	<u>B</u>
Notes	

Response:

```
import os
import sys
import json
import logging

def main():
    logging.info('Starting')
```