

ELLYPUNK

Applied

WEEKLY DIGEST / T201

21-27 September 2026



This week

21-27 September 2026

Available archived editions, in their original date order.

MONDAY / 21 September 2026

PowerShell JSON Depth: Preserve Nested Data

3

[Open original online](#)

TUESDAY / 22 September 2026

HTTP Status Codes: Decide When a Client Should Retry

8

[Open original online](#)

WEDNESDAY / 23 September 2026

PowerShell CSV Encoding: Keep Text Intact Across Files

13

[Open original online](#)

THURSDAY / 24 September 2026

Python Logging: Leave a Trail You Can Debug

18

[Open original online](#)

FRIDAY / 25 September 2026

Python Exceptions: Catch the Failure You Can Handle

23

[Open original online](#)

Only retained editions are included; gaps have not been filled.

Copyright Ellypunk. Contains AI-generated artwork.

ellypunk.com

POWERSHELL JSON DEPTH PRESERVE NESTED DATA

OBJECTIVE:

Serialize a nested object, read it back, and verify the value that matters.

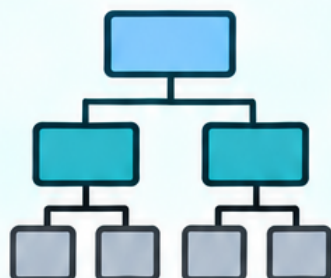
Small
Steps
Real
Progress
♡

Plan
Build
Test
Improve
Together
♡

Data
Works
Better
Together
♡

OBJECT → JSON TEXT → OBJECT

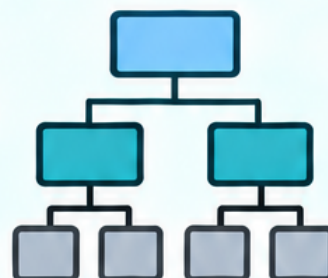
OBJECT



JSON TEXT



OBJECT



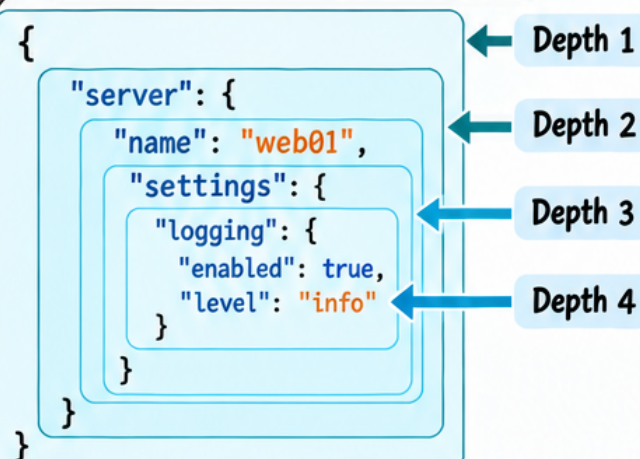
PowerShell 7.6 · Linux / macOS / Windows
Same cmdlets on all three platforms.



AUTOMATE
LEARN
BUILD
GO FURTHER

DEPTH IS A BOUNDARY

Default -Depth: 2



If the input is deeper, convertto-json warns.



\$json = \$config | convertto-json -Depth 4

Small Steps
Real Progress

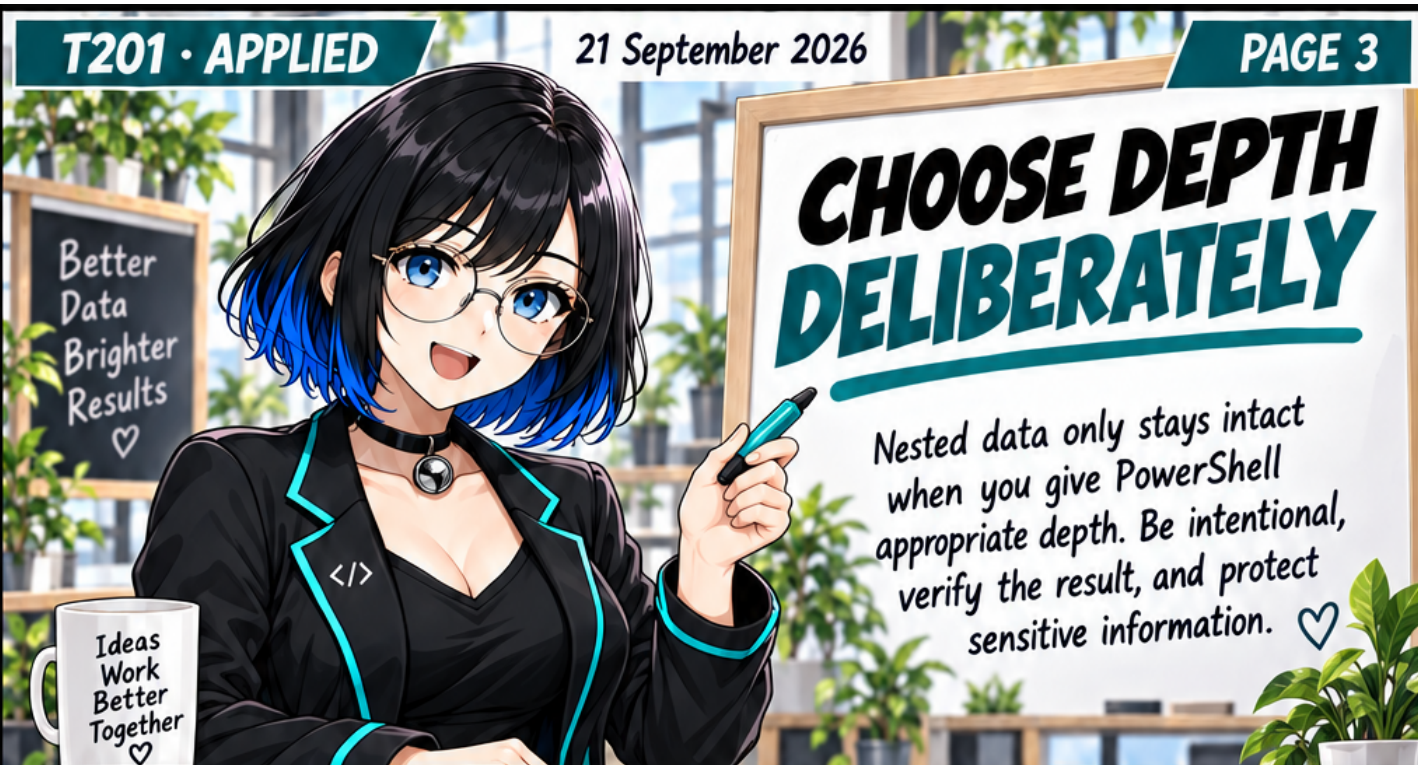
Choose a depth that includes the deepest property you intend to preserve.



Linux / macOS / Windows

CLEAN DATA
BETTER TOOLS
BRIGHTER IDEAS





1 TRACE THE PATH

Follow the nesting path to know how deep your data goes.

```

graph TD
    config --> Database
    Database --> Options
    Options --> Encrypt
    
```

2 VERIFY THE ROUND TRIP

Read the JSON back and assert a deep value.

```

graph LR
    JSON_in[JSON (in)] --> JSON_out[JSON (out)]
    JSON_out --> JSON_in
    
```

Deep value matches

3 PROTECT DATA

Do not serialize secrets into logs or shared files.

API KEYS

PASSWORDS

TOKENS

i Version note: behaviour can differ across PowerShell releases. Record and test your supported version.





1 PROBLEM



2 ACTION



3 CHECK



4 IMPROVEMENT



PowerShell JSON DEPTH: PRESERVE NESTED DATA



RECAP: Export, import, verify.



Keep your structure.
Keep your data.

JSON ROUND-TRIP CHECKLIST

- ☐ Know the nesting path
- ☐ Set explicit -Depth
- ☐ Read warnings
- ☐ Convert back
- ☐ Assert a deep value

Small Checks
Big Confidence

ELLY'S FINAL TIP

Depth is not a magic maximum.
It is a deliberate boundary
you must test.



PowerShell 7.6 · Linux / macOS / Windows



HTTP STATUS CODES

Decide when a client should retry

OBJECTIVE:

Choose whether to fix, wait, or stop.

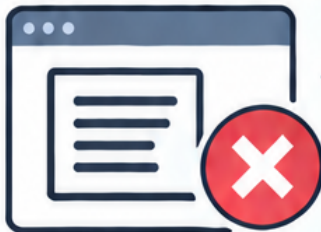
APIs
Make
a Brighter
Tomorrow
♡

Learn
Practice
Build
Improve
Together
♡

Small
Steps
Real
Progress
Together
♡

400 Bad Request

The request is invalid, usually due to client input.



This means **fix** the request. **Do not retry** until the input is corrected.

429 Too Many Requests

The client has sent too many requests (rate limited).



The response may include a **Retry-After** header. **Wait** before retrying.

503 Service Unavailable

The server is temporarily unavailable (e.g. maintenance or overload).



This is often a temporary condition. A **safe retry** may be appropriate, ideally with backoff.

A status is a clue,
not a blanket
retry command.

Better
APIs
Happier
Developers
♡

OBSERVE
REASON
CHOOSE
RETRY SMARTER



READ THE WHOLE RESPONSE

A status is only the first clue. 🚀

HTTP Response

```
1 HTTP/1.1 503 Service Unavailable
2 Date: Tue, 22 Sep 2026 10:15:30 GMT
3 Retry-After: 10
4 Content-Type: text/plain; charset=utf-8
5 maintenance
```

1 · STATUS



```
HTTP/1.1 503 Service Unavailable
```

503 means the service is temporarily unavailable.

2 · HEADER



```
Retry-After: 10
```

Retry-After: 10 suggests waiting 10 seconds.

3 · BODY



```
5 maintenance
```

Read the service message for context.

4 · REQUEST



```
GET /api/data HTTP/1.1
Host: api.example.com
Accept: application/json
```

This GET is documented as read-only. Check safety before retrying writes.

Wait, then retry this GET within a small attempt budget.

Small Steps
Real Progress
Together

OBSERVE
UNDERSTAND
MAKE BETTER DECISIONS
BRIGHTER APIS



RETRY ONLY WHEN SAFE.

Is the operation safe to repeat?

Not every request should be retried. Only repeat operations that are safe to run again.

Plan
Retry
Respect
Be Kind
to APIs
♡

Reliable
Apps
Happier
People
♡

Better
Clients
Happier
Servers
♡

1 CHECK IF IT'S SAFE

Is the operation safe to repeat?



Safe to retry

Examples include read-only requests or idempotent operations.



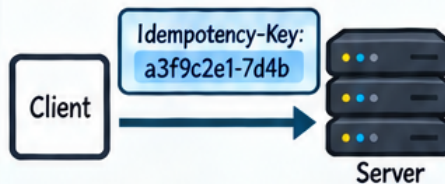
Not safe to retry

Operations that create, charge, or change data may cause duplicate effects.

Only retry when you are confident the operation can be repeated without unwanted side effects.

2 USE AN IDEMPOTENCY KEY

For supported write APIs, use an idempotency key.



An idempotency key helps the server recognise repeated requests and avoid performing the operation more than once.

3 RESPECT RETRY-AFTER

Respect **Retry-After** when present.



If the response includes a **Retry-After** header, wait for the specified time before retrying.

4 LIMIT ATTEMPTS; USE BACKOFF AND JITTER

Limit attempts; use backoff and jitter.



- Use a limited number of retry attempts.
- Use backoff to increase the wait time between attempts.
- Add jitter to vary the wait time and avoid retrying at the same time as many other clients.

5 STOP WHEN THE BUDGET IS SPENT

Stop when the retry budget is spent.



If you have reached your retry budget (for example, your maximum attempts or overall time limit), stop retrying and handle the failure.

A clear stopping point prevents wasted resources and improves reliability.

Be smart about retries. Check, respect, limit, and stop. Safer clients build a healthier web.

APIs
Power
Progress
♡

OBSERVE
REASON
CHOOSE
RETRY SMARTER



WORKED EXAMPLE: 429 IN PRACTICE

A real flow from problem to improvement.

Let's walk through a complete example using a **429** response.

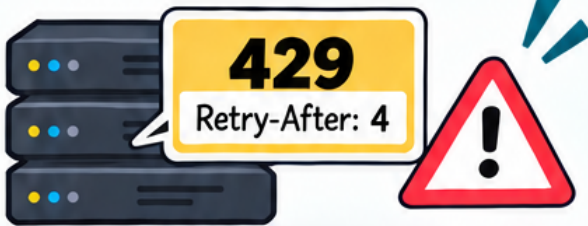
Good Clients Make Healthier APIs

Better Pacing Happier Systems

Smarter Clients Brighter Tomorrow

1 PROBLEM: 429 RESPONSE

The API responds **429** with 'Retry-After: 4' (4 seconds).



The client has sent too many requests. The server asks us to wait **4** seconds before trying again.

2 ACTION: WAIT, THEN RETRY

Wait at least 4 seconds, then retry a request that is safe to repeat.



Only retry the operation if it is safe to repeat (for example, a read-only or idempotent request).

3 CHECK: SUCCESSFUL RETRY

The next response is **200**; no duplicate write occurred.



The request succeeds and the expected result is returned, with no duplicate write occurring.

4 IMPROVE: PACE FUTURE REQUESTS

Pace future requests before the limit is reached.



Use pacing (for example, backoff and jitter) to reduce the chance of hitting the rate limit again.



RECAP: FIX, WAIT, OR STOP.

Here's a quick recap to help you decide when a client should retry.

Learn
Practice
Build
Improve
Together

Same
Concepts
Stronger
Clients

1 400 — FIX INPUT



Fix the input,
don't repeat
unchanged.

2 429 — WAIT AND RESPECT



Wait and respect
Retry-After
when present.

3 503 — BACK OFF



Back off
if a retry
is safe.

4 CAP ATTEMPTS



Cap attempts
and watch
outcomes.

Final tip:
Never let a retry
become a storm.

Practical
Ideas
Real Progress
A Brighter
Tomorrow

OBSERVE
REASON
CHOOSE
RETRY SMARTER



POWERSHELL CSV ENCODING

OBJECTIVE:

Export and import accented text without changing its characters.

Small Steps
Real Progress

ORIGINAL DATA

Name	City
Zoë	München

CSV FILE



IMPORTED DATA

Name	City
Zoë	München

The round trip must preserve the text.

PowerShell 7.5 · Linux / macOS / Windows



AUTOMATE
LEARN
BUILD
GO FURTHER

Ideas
work
better
together.
♡

BYTES NEED AN ENCODING

Same text.
Same bytes.
Brighter futures.

Small
Steps
Real
Progress
♡

1 Text becomes bytes when CSV is saved.

ORIGINAL TEXT

Name	City
Zoë	München



BYTES (UTF-8)

Text is converted to bytes using the chosen encoding (e.g. UTF-8) when the CSV file is saved.

2 Import-Csv decodes those bytes back into text.



BYTES (UTF-8)



IMPORTED TEXT

Name	City
Zoë	München

Import-Csv reads the bytes and decodes them using the same encoding to get the original text back.

3 Use the same UTF-8 encoding for both directions.

ORIGINAL TEXT

Name	City
Zoë	München



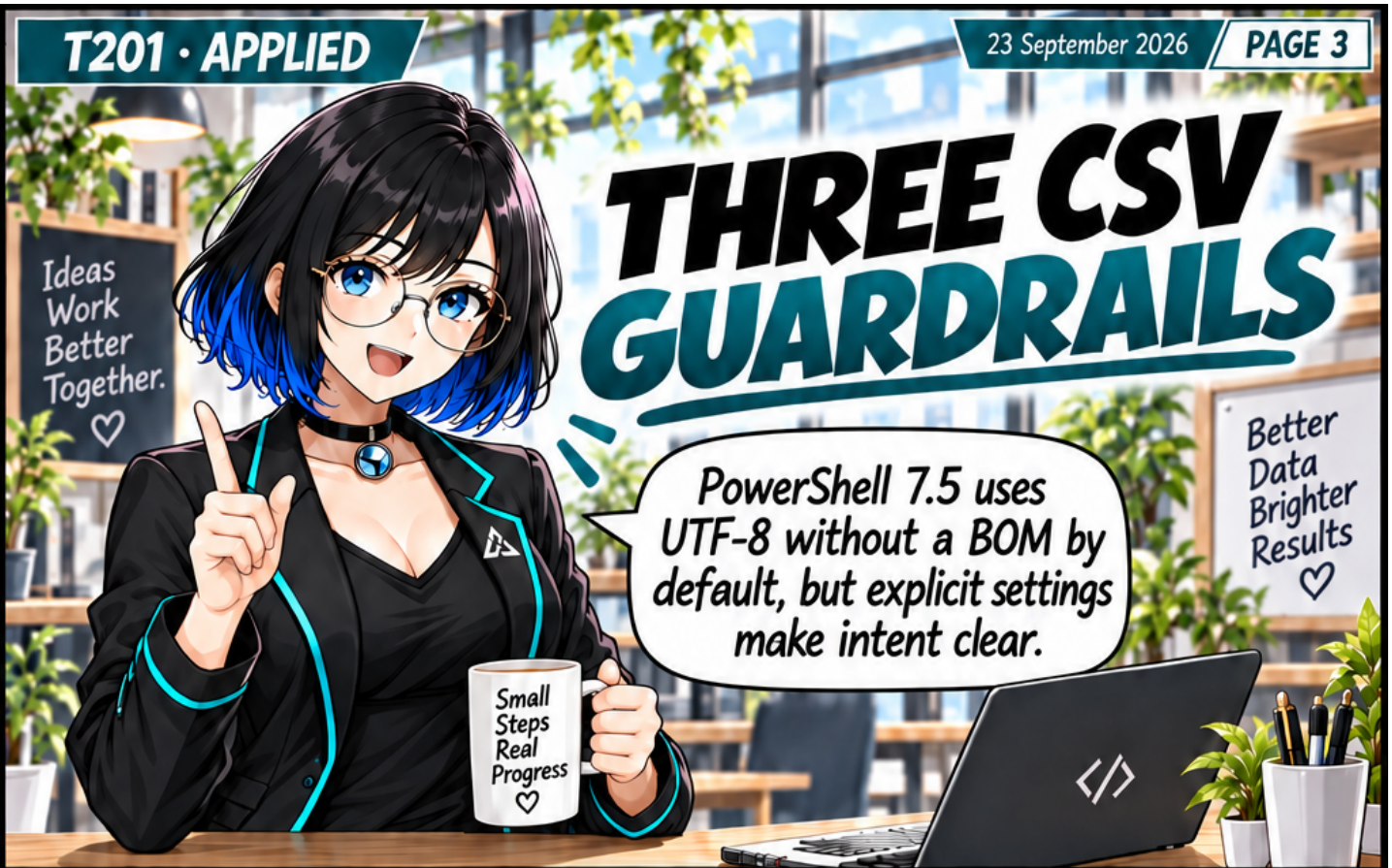
IMPORTED TEXT

Name	City
Zoë	München

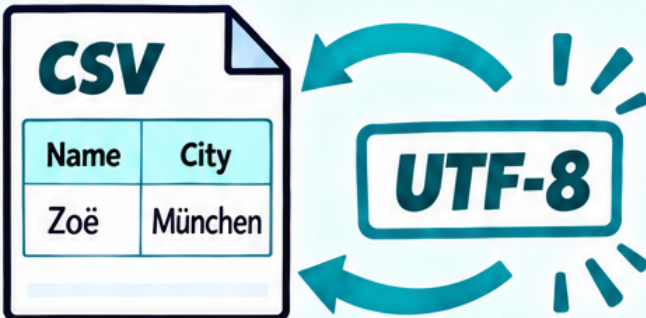
Using the same UTF-8 encoding when saving and importing keeps the text intact.

If the decoder guesses wrong, characters can change.

CLEAN DATA
REAL PROGRESS
BRIGHTER IDEAS ♡



1 Choose UTF-8 explicitly on export and import.



2 Keep the same delimiter on both sides.



3 Check real accented values after the round trip.



WORKED EXAMPLE: CSV ROUND TRIP

Let's keep
text intact
across files!

Real
Data
Stronger
Tomorrow

Small
Steps
Real
Progress

1 PROBLEM

PROBLEM:
Accented names
change after a
CSV export.



2 ACTION

ACTION:
Choose UTF-8
on export
and import.



Good
Data
Happier
People

3 CHECK

CHECK:
Zoë and München
still match.



4 IMPROVE

IMPROVE:
Keep the encoding
explicit in your
script.

Same
Text
More
Possibilities

PowerShell 7.5 · Linux / macOS / Windows

```
[pscustomobject]@{Name='Zoë'; City='München'} | Export-Csv -Path './people.csv' -NoTypeInformation -Encoding utf8NoBOM
$person = Import-Csv -Path './people.csv' -Encoding utf8NoBOM
$person.Name -eq 'Zoë' -and $person.City -eq 'München'
```

Expected result: **True.**

Prerequisite: PowerShell 7.5 in a writable working directory.

*Ideas work
better together.*



CSV ENCODING RECAP

CHECKLIST

- ✓ Save CSV as UTF-8.
- ✓ Import with the same encoding.
- ✓ Keep the delimiter consistent.
- ✓ Compare real accented values.

ORIGINAL DATA

Name	City
Zoë	München



IMPORTED DATA

Name	City
Zoë	München

A successful import is not enough; check the characters.

PYTHON LOGGING: LEAVE A TRAIL YOU CAN DEBUG.

Objective:

Record useful events
without exposing
private data.

START → ERROR → CHECK



START



ERROR



CHECK



Never log
passwords or tokens.

CHOOSE THE RIGHT LEVEL



INFO:

Normal milestones worth knowing.



INFO



Application started



WARNING:

A problem that may need attention.



WARNING



Slow response from database



ERROR:

The operation failed.



ERROR

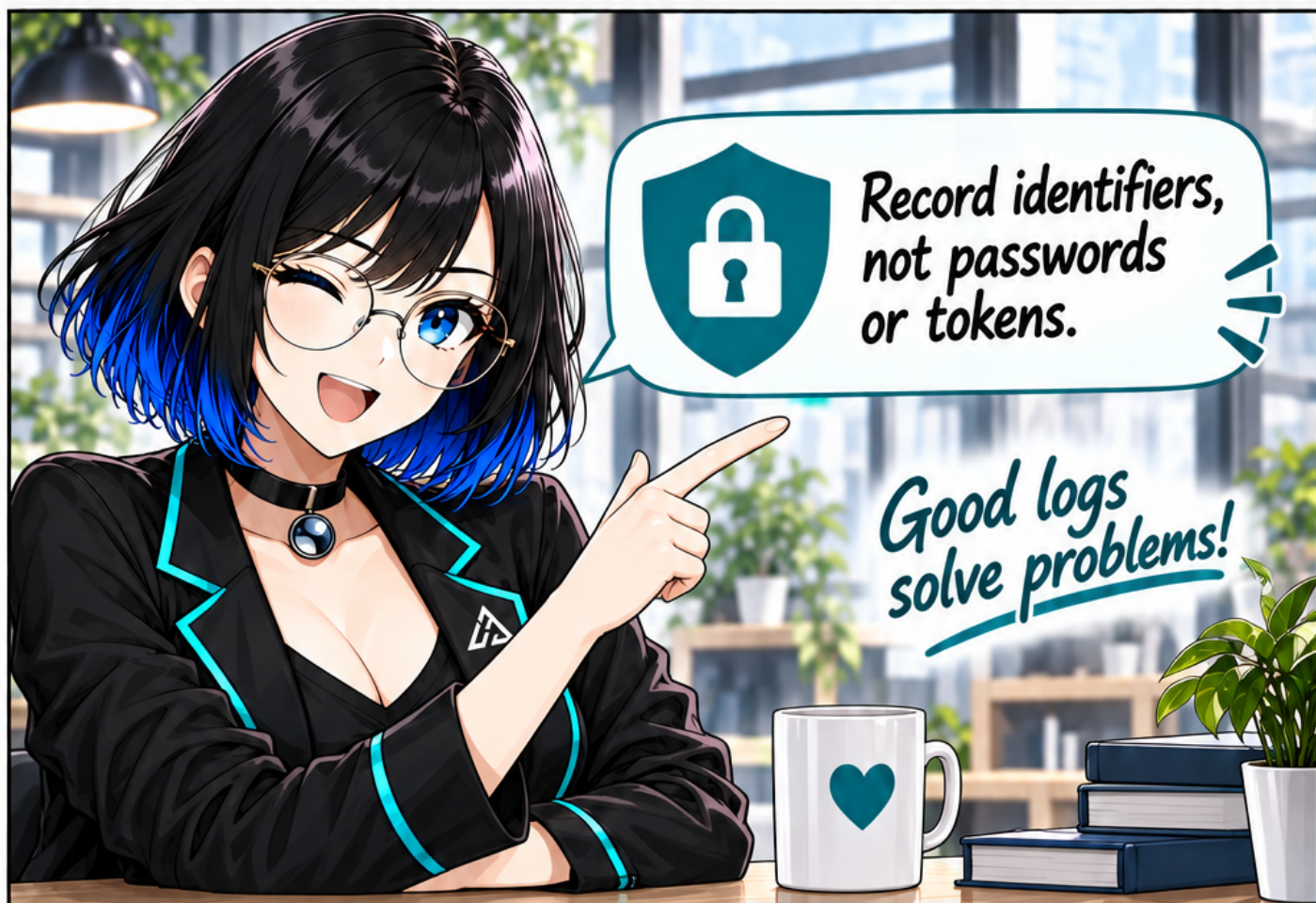


Failed to save user profile



Use **DEBUG** for extra detail while investigating.

WHAT MAKES A USEFUL LOG?



WORKED EXAMPLE: A FAILED IMPORT



1

PROBLEM:

A data import stops with no clear clue.



2

ACTION:

Log the job ID and failure reason.



3

CHECK:

Follow the event trail to the failed step.



4

IMPROVE:

Add a safe, useful message before the next run.



RECAP: LOG WHAT HELPS.



Pick **INFO**,
WARNING, or
ERROR.

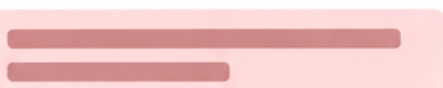


Add a useful job
or request ID.



Check the event
trail after a
failure.

A HELPFUL EVENT TRAIL



Keep passwords
and tokens
out of logs.



A good log helps
you find the
next action.





PYTHON EXCEPTIONS

Catch the Failure You Can Handle

OBJECTIVE:

Handle expected bad input without hiding other failures.

Better Code
Brighter Tomorrow

1 VALID INPUT BECOMES A NUMBER

Input: 12

Convert to number



12

Number
(success)

Valid input is converted and used as a number.

2 BAD INPUT RAISES VALUEERROR

Input: twelve

Convert to number



ValueError

Cannot convert 'twelve' to a number.

Handle this expected error to give a helpful response.

3 UNEXPECTED FAILURES MOVE ONWARD

Input: (something else)

Something unexpected happens



Pass the exception to the caller



Caller

Don't hide unexpected failures. Let them propagate to the caller.





THREE POSSIBLE PATHS

Python chooses one path after TRY.



SUCCESS → ELSE

Conversion worked.
Use the number.



ValueError → EXCEPT

Expected bad input
gets a helpful response.



OTHER ERROR → CALLER

Do not hide a failure
you cannot handle.

CATCH NARROWLY

Handle what you expect.
Let the rest surface.

Better
Code
Brighter
Tomorrow



1 Name the expected error



Use ValueError for invalid number text.

2 Keep TRY small



Protect only the conversion step.

3 Let surprises surface



Do not replace every failure with a default.



HANDLE BAD NUMBER TEXT

Even small mistakes happen. Let's handle them!

Turn mistakes into progress. ♡

Learn
Build
Improve
Together ♡

1 PROBLEM



2 ACTION

Python · Linux / macOS / Windows

```
def parse_count(text):
    try:
        return int(text)
    except ValueError:
        return None
```



3 CHECK

`parse_count("12")` → 12
`parse_count("twelve")` → None



4 IMPROVE

Ask the visitor to enter digits again.

Better
Code
Brighter
Tomorrow ♡



EXCEPTION CHECKLIST

Catch the error you can handle.
Let the rest surface.

1 Catch a specific type.



2 Keep TRY small.



3 Check both success and failure.



Elly's final tip:

A helpful error path
is part of a reliable
program.

Small
Steps
Real
Progress