Sure 🙂
I'll explain the code **step by step**, in **simple terms**, so you clearly understand **how this pydash-like get works internally**.

🎯 Goal of the Function

We want a function like:

```
deep_get(
    data,
    "sections.SECTION_SEARCH_RESULT[3].trackingData[0].table_name"
)
```

That safely returns the value **without throwing errors**, even if:

- a key is missing
- an index doesn't exist
- the structure changes

🧠 Core Idea

1. **Split the path string** into steps
2. **Walk through the data** step-by-step
3. Handle:
    - dictionary keys → .key
    - list indexes → [index]
4. If anything goes wrong → return default

📌 The Code (Reference)

```
import re
from typing import Any

_INDEX_PATTERN = re.compile(r"([^[\]]+)|\[(\d+)\]")

def deep_get(data: Any, path: str, default=None):
    if not path:
```

```
        return default

    current = data

    for key, index in _INDEX_PATTERN.findall(path):
        try:
            if key:
                if isinstance(current, dict):
                    current = current.get(key, default)
                else:
                    return default

            elif index:
                if isinstance(current, list):
                    current = current[int(index)]
                else:
                    return default

        except (IndexError, KeyError, TypeError):
            return default

        if current is default:
            return default

    return current
```

🔍 Step-by-Step Explanation

1️⃣ Regex Pattern

```
_INDEX_PATTERN = re.compile(r"([^[\]]+)|\[(\d+)\]")
```

What it does

This regex **extracts keys and list indexes** from the path string.

For this path:

sections.SECTION_SEARCH_RESULT[3].trackingData[0].table_name

It produces:

[

('sections', ''),

('SECTION_SEARCH_RESULT', ''),

('', '3'),

('trackingData', ''),

('', '0'),

('table_name', '')

]

✓ First group → dictionary key
✓ Second group → list index

2️⃣ deep_get() Function Start

def deep_get(data, path, default=None):

- data → your JSON (dict / list)
- path → dot + bracket notation string
- default → returned if anything fails

3️⃣ Handle Empty Path

```
if not path:
    return default
```

If no path is provided → nothing to get → return default

4️⃣ Initialize Pointer

current = data

current holds **where we are** in the JSON while walking through it.

5️⃣ Loop Through Path Parts

for key, index in _INDEX_PATTERN.findall(path):

Each iteration handles **one step** in the path:

- either a dictionary key
- or a list index

6️⃣ Dictionary Access (.key)

```
if key:
    if isinstance(current, dict):
        current = current.get(key, default)
    else:
        return default
```

What happens

- Check if current object is a dictionary
- Use .get() so missing keys don't crash
- If current isn't a dict → path is invalid → return default

7️⃣ List Access ([index])

```
elif index:
    if isinstance(current, list):
        current = current[int(index)]
    else:
        return default
```

What happens

- Ensure current object is a list
- Convert index string → integer
- Access list safely

8️⃣ Error Protection

```
except (IndexError, KeyError, TypeError):
    return default
```

Catches:

- ❌ Index out of range
- ❌ Wrong type access
- ❌ Invalid operations

Prevents crashes ✓

🛑 Early Stop If Missing Value

if current is default:

    return default

If .get() failed → stop immediately

🔚 Final Return

return current

If all steps succeeded → return the final value 🎉

✅ Example Walkthrough

data = {

  "sections": {

    "SECTION_SEARCH_RESULT": [

      {},

      {},

      {},

      {

        "trackingData": [

          {"table_name": "restaurant_table"}

        ]

      }

    ]

  }

}

Path:

sections → SECTION_SEARCH_RESULT → [3] → trackingData → [0] → table_name

Result:

restaurant_table

🚀 Why This Is Powerful

✓ No crashes
✓ Works with deeply nested JSON
✓ One-line access
✓ No external dependency
✓ Same behavior as pydash.get

🧩 Want More?

I can also help you build:

- deep_set() (write values)
- Wildcards ([*])
- Dot-escaped keys ("a.b")
- Faster non-regex version
- JSON → auto path generator

Just tell me 😊