LLM Kod Analiz Dogrulayici

Gercek kod yapisi ile LLM analizini karsilastir

Groq - Llama 3.3 70B

Analiz Et

Python Kodu

```
1  #
   ==================================
   ==================================
   =======
2  # TEST 4: ULTRA ZOR - METACLASS,
   DESCRIPTOR, DYNAMIC DISPATCH
3  # Zorluk: COK ZOR | Metaclass,
   Descriptor, MRO, Magic Methods,
   Monkey Patching
4  #
   ==================================
   ==================================
   =======
5  # Bu kod LLM'yi ciddi sekilde
   zorlayacak cunku:
6  # 1. Metaclass __new__ ve __init__
   SINIF OLUSTURULURKEN cagrilir,
   instance'da DEGIL
7  # 2. Descriptor __get__/__set__
   ATTRIBUTE ERISILDIGINDE cagrilir
8  # 3. __getattr__ sadece BULUNAMAYAN
   attribute icin cagrilir
9  # 4. super() cagrilari MRO'ya gore
   FARKLI metodlari cagirabilir
10 # 5. Monkey patching RUNTIME'da
   davranisi degistirir
11 # 6. Context manager
   __enter__/__exit__ WITH blogu
   icinde cagrilir
12 # 7. Generator'lar LAZY evaluation
   yapar - yield HEMEN cagrilmaz
13 #
   ==================================
   ==================================
   =======
14
15 from functools import wraps,
   partial
16 from contextlib import
   contextmanager
```

Sonuclar | Graf Gorsellestirme | Detaylar

LLM Dogruluk Orani

84.6%

ORTA GUVENILIRLIK

44	8	33
Dogru Tespit	Halusinasyon	Kacirilan

Dogru Tespitler 44

- run_complex_system → _finalize_all
- _on_first_instance → _notify_creat
- run_complex_system → patch_data_pr
- on_attribute_access → log_access

Halusinasyonlar (LLM Yanlislari) 8

- __init__ → _on_descriptor_init ✗
- __init__ → _setup_internals ✗
- __init__ → _on_processor_init_star
- init → super ✗

```
17 import sys
```