PENERAPAN *TIME EXECUTOR* DALAM MENGOPTIMALKAN AKURASI WAKTU DENGAN MENGGUNAKAN *CLOSURE DECORATOR*

Ahmad Sahidin Akbar[1], Nathanael Daniel Santoso[2], Safitri[3], Ferdy Kevin Naibaho[4], Elilya Octaviani[5]

Jurusan Sains Data, Fakultas Sains, Institut Teknologi Sumatera, Lampung Selatan, Indonesia

Email: ahmad.122450044@student.itera.ac.id, nathanael.122450059@student.itera.ac.id, safitri.122450071@student.itera.ac.id, ferdy.122450107@student.itera.ac.id, elilya.122450009@student.itera.ac.id

ABSTRAK

Banyaknya permasalahan di kehidupan seperti tidak bijaknya individu dalam mengatur waktu, telatnya transportasi umum untuk memulai perjalanan menjadi masalah yang harus diperhatikan. Penelitian ini bertujuan dengan dilakukannya penerapan konsep closure yang dikombinasikan dengan *time executor decorator* untuk mengatur waktu eksekusi dalam mengoptimalkan akurasi waktu. Metode yang dilakukan dengan menggunakan pendekatan yang memanfaatkan setiap konsep untuk meningkatkan pemahaman dan efisiensi dari sebuah kode. Studi ini menghasilkan sebuah eksplorasi yang memiliki potensi untuk meningkatkan kualitas dan kinerja perangkat lunak dengan menganalisis konsep-konsep utama seperti prinsip *closure*, penggunaan *decorator*, dan implementasi *time executor*.

Kata kunci : *closure, decorator, time executor,* waktu

PENDAHULUAN

Dalam era teknologi ini sudah tidak asing dengan banyaknya pengembangan yang memudahkan manusia dalam melakukan aktifitas. Salah satunya penggunaan *time executor* yang membantu dalam kehidupan seperti mengembangkan sebuah perangkat lunak, mengoptimalkan kinerja, menganalisis data, mengatur sebuah jadwal kegiatan, pemantauan kinerja aplikasi dan lain sebagainya.

Dengan adanya *time executor* ini, dapat diterapkan pada *closure decorator* yang akan memudahkan untuk mengeksekusi sebuah waktu. *Closure* ini membantu

melakukan pengukuran waktu serta mengidentifikasi bagian-bagian dari sebuah kode. *Closure* juga dapat mengevaluasi serta memperkirakan waktu yang akurat dalam menyelesaikan sebuah tugas-tugas tersebut.

Penerapan *closure* pada *decorator time executor* bertujuan untuk mengetahui tingkat akurasi dari sebuah *time executor* yang dihasilkan dalam sebuah program.

METODE

1. *Closure*

   Dalam pemrograman, tidak asing terdengar kata variabel global. Variabel global seringkali diandalkan dalam berbagi data dan dalam hal ini dapat menyebabkan kebingungan dan kesalahan dalam pengembangan perangkat lunak. *Closure* dapat mengatasi kesalahan ini dengan fungsi yang dapat mengakses variabel di dalamnya tanpa harus mengandalkan variabel global dan menghasilkan kode yang terstruktur (NOSUKE, 2023).

2. *Decorator*

   Decorator merupakan bagian luar dari sebuah fungsi yang akan mengembalikan sebuah fungsi. *Decorator* sendiri memiliki empat input yang digunakan, yaitu *args dan **kwargs (Wake, 2019). Fungsi dari *args adalah memanggil fungsi dan menjadikan argumen list yang akan dimasukkan ke dalam fungsi menjadi list (*unpack* list menjadi *arguments)*. Sedangkan **kwargs adalah *unpack dictionary* untuk fungsi menjadi *keyword argument* (Amin, 2017).

3. *Memoization*

   Memoization merupakan sebuah teknik optimasi di mana sebuah program menyimpan hasil eksekusi dari sebuah fungsi untuk digunakan kembali pada eksekusi-eksekusi berikutnya. *Memoization* juga menerima sebuah fungsi sebagai argumen dan mengembalikan fungsi tersebut (Afif, 2020).

4. *Syntactic Sugar*

   Syntactic sugar merupakan sebuah istilah dalam pemrograman komputer di mana ini digunakan ketika terjadi perubahan sintaks. Jadi dapat dikatakan *syntactic sugar* merupakan perubahan dari sebuah bahasa pemrograman menjadi lebih mudah untuk digunakan dan dapat digambarkan menjadi sebuah jalan pintas dalam pemrograman (Mulyawan, 2017).

HASIL DAN PEMBAHASAN

```
import time
```

```
def time_execution(func):
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result    =    func(*args,
**kwargs)
        end_time = time.time()
        execution_time = end_time
- start_time
        print(f"Execution time of
{func.__name__}: {execution_time}
seconds")
        return result
    return wrapper

@time_execution
def example_function(n):
    # Fungsi contoh
    total = 0
    for i in range(n):
        total += i
    return total

# Contoh penggunaan
result                              =
example_function(1000000)
print("Result:", result)
```

```
Execution time of example_function: 0.13381385803222656 seconds
Result: 499999500000
```

- **Definisi *Decorator* time_execution**
  Decorator time_execution didefinisikan sebagai fungsi yang menerima satu parameter, yaitu fungsi yang akan didekorasi (func).
- **Fungsi Wrapper**
  Di dalam *decorator time_execution*, ada fungsi bernama *wrapper*. Fungsi ini menerima argumen posisional (**args*) dan argumen kata kunci (***kwargs*) yang akan diteruskan ke fungsi yang didekorasi.
- **Mengukur Waktu Mulai Eksekusi**
  Sebelum fungsi yang didekorasi dieksekusi, waktu saat ini diambil menggunakan fungsi *time.time*() dan disimpan dalam variabel *start_time*.
- **Eksekusi Fungsi Asli**
  Fungsi yang didekorasi (func) dieksekusi di dalam fungsi *wrapper* dengan meneruskan argumen yang diterima.
- **Mengukur Waktu Akhir Eksekusi**
  Setelah fungsi yang didekorasi selesai dieksekusi, waktu saat ini diambil lagi menggunakan *time.time*() dan disimpan dalam variabel *end_time*.
- **Menghitung Durasi Waktu Eksekusi**
  Durasi waktu eksekusi dihitung dengan mengurangkan *start_time* dari *end_time*.
- **Menampilkan Waktu Eksekusi**
  Durasi waktu eksekusi dicetak ke konsol bersama dengan nama fungsi yang dieksekusi.
- **Mengembalikan Hasil Eksekusi Fungsi**

Hasil dari eksekusi fungsi yang didekorasi dikembalikan.

- **Penerapan Decorator** Fungsi yang ingin kita ukur waktu eksekusinya didekorasi dengan *@time_execution*. Ini dilakukan dengan menempatkan *@time_execution* di atas definisi fungsi.
- **Contoh Penggunaan** Fungsi yang didekorasi dipanggil dengan argumen yang sesuai. Setelah fungsi selesai dieksekusi, waktu eksekusinya dicetak bersama dengan hasil eksekusi fungsi tersebut. Dengan menggunakan decorator *@time_execution*, kita dapat dengan mudah mengukur waktu eksekusi dari berbagai fungsi tanpa perlu menambahkan kode pengukuran waktu di setiap fungsi yang ingin kita ukur. Ini meningkatkan modularitas dan kejelasan kode kita.

KESIMPULAN

Penerapan konsep *closure* dalam kombinasi dengan *decorator time executor* telah terbukti efektif dalam mengoptimalkan akurasi waktu eksekusi dalam pengembangan perangkat lunak. *Closure* memungkinkan dalam mengatur fungsi secara leksikal agar sesuai dengan lingkungannya, dan *decorator time executor* memungkinkan pengukuran waktu eksekusi secara otomatis. Dengan menggabungkan kedua konsep ini, mampu menghasilkan penelitian yang berpotensi tidak hanya meningkatkan kualitas dan kinerja perangkat lunak, namun juga pemahaman dan efisiensi kode. Dengan menganalisis konsep-konsepnya seperti *closure*, *decorator*, dan implementasi time executor, maka akan berkontribusi secara signifikan terhadap pengembangan perangkat lunak yang mengatur waktu dengan lebih efisien dan akurat.

DAFTAR PUSTAKA

Afif, N. (2020, October 9). *Memoization di Python. di artikel kali ini saya akan… | by Naufal afif | Medium.* Retrieved April 25, 2024, from Naufal afif: https://naufalafif58.medium.com/memoization-di-python-3e1ef419d371

Amin, I. (2017, November 9). **args dan **kwargs Pada Python.* Retrieved April 25, 2024, from YouTube: Home: https://www.codepolitan.com/blog/args-dan-kwargs-pada-python-590c80ef39a84/

Mulyawan, R. (2017, November 9). *Syntactic Sugar*. Retrieved April 25, 2024, from YouTube: Home: https://rifqimulyawan.com/literasi/syntactic-sugar/

NOSUKE. (2023, August 18). *Tutorial Python 44: Python Closures, Pahami Penggunaannya!* Retrieved April 25, 2024, from Codekey: https://codekey.id/python/tutorial-python-44-python-closures-memahami-cara-penggunaannya/

Wake, U. (2019, September 6). *Apa itu Decorators dan Bagaimana Cara Menggunakannya? | by Ulrich Wake.* Retrieved April 25, 2024, from Medium: https://medium.com/@uulwake/apa-itu-decorators-dan-bagaimana-cara-menggunakannya-1f600485b382