Penghitung Discount dengan Decorator dan Syntactic Sugar

Kelompok 7

[1]Aditya Rahman
122450113

[2]Farahanum Afifah Ardiansyah
122450056

[3]Fayyaza Aqila Syafitri Achjar
122450131

[4]Azizah Kusumah Putri
122450068

[5]Eggi Satria
122450032

[6]Meira Listyaningrum
122450011

Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera
Jl. Terusan Ryacudu, Way Huwi, Kec. Jati Agung, Kabupaten Lampung Selatan, Lampung 35365
Kontak : [1]*aditya.122450113@student.itera.ac.id*, [2]*farahanum.122450056@student.itera.ac.id*,
[3]*fayyaza.122450131@student.itera.ac.id*, [4]*azizah.122450068@student.itera.ac.id*,
[5]*eggi.122450032@student.itera.ac.id*, [6]*meira.122450011@student.itera.ac.id*

1. PENDAHULUAN

1.1 Latar Belakang

Python merupakan bahasa pemrograman yang sangat umum digunakan. Penyebab kepopuleran pemrograman python ini diantaranya adalah karena mudah dipelajari, sintaks yang digunakan mudah dibaca dan dipahami, dan fleksibel (Puguh, 2024). Selain itu pemrograman python ini bersifat *open source* sehingga dapat digunakan dengan gratis.

Decorator dan *Syntactic Sugar* merupakan salah dua fungsi dalam python yang membantu dalam menyederhanakan sintaks sehingga mudah dibaca dan dipahami. Tujuan utama dari fungsi *Decorator* adalah untuk menjalankan beberapa fungsi yang memiliki kemiripan dengan mengimplementasikan kemiripan tersebut dalam suatu fungsi sehingga efisiensi dalam sintaks akan meningkat (Richard, 2021).

Sedangkan untuk *Syntactic Sugar* bertujuan untuk menyederhanakan sintaks agar lebih "manis" dan dapat dengan mudah dipahami seperti pada *lambda functions* (Paulacy, 2024).

1.2 Rumusan Masalah

a. Bagaimana python closure diimplementasikan dalam fungsi *Decorator* dan fungsi *Syntactic Sugar*?
b. Bagaimana fungsi *Decorator* digunakan dalam program penghitung diskon?
c. Bagaimana fungsi *Syntactic Sugar* diterapkan dalam program penghitung diskon?

1.3 Tujuan

Mengimplementasikan closure dengan fungsi *Decorator* dan *Syntactic Sugar* dalam program penghitung diskon.

2. METODE

2.1 *Decorator*

Dekorator adalah pola desain dengan Python yang memungkinkan pengguna menambahkan fungsionalitas baru ke objek yang sudah ada tanpa mengubah strukturnya. Dekorator biasanya diterapkan pada fungsi, dan mereka memainkan peran penting dalam meningkatkan atau memodifikasi perilaku fungsi (Mwiti, 2024). Dekorator biasanya didefinisikan menggunakan simbol "@" diikuti oleh nama dekorator di atas definisi fungsi yang akan dimodifikasi.

2.2 *Syntactic Sugar*

Sintaks gula (*syntactic sugar*) adalah fitur dalam bahasa pemrograman yang dirancang untuk membuat kode lebih mudah ditulis atau lebih mudah dibaca. Dalam Python, beberapa contoh sintaks gula termasuk pemformatan string menggunakan f-string, sintaks untuk membuat daftar (list comprehensions), dan menggunakan "*with*" untuk mengelola konteks. Jadi dalam konteks utama *syntactic sugar* adalah cara alternatif untuk menulis kode yang lebih singkat dan mudah dimengerti tanpa merubah hasil

3. PEMBAHASAN

Dalam bab ini, kami melakukan analisis terhadap closure dengan menggunakan *syntactic sugar* dan *decorator*. Kami mengimplementasikan penerapan closure dengan *syntactic* sugar dan decorator menggunakan python dengan membuat penghitung *discount*, kode yang kami buat dan gunakan untuk analisis ini dapat diakses pada link berikut :
https://colab.research.google.com/drive/1fWrj2_AhSk-gpD1Pz3Ko4jhGPhIHuiog?usp=sharing

3.1 Penerapan *Closure* dengan *Syntactic Sugar* dan *Decorator*

Closure, *Syntactic sugar*, dan *decorator* merupakan teknik yang ada dalam pemrograman yang berguna untuk membuat kode yang fleksibel, terstruktur, dan mudah dibaca. Fungsi *closure* merupakan fungsi yang bisa mengakses variabel dari scope luarnya. *Syntactic sugar* dapat membuat penulisan *closure* lebih mudah dibaca dan lebih ringkas, dan fungsi *decorator* memungkinkan untuk dapat menambahkan fungsionalitas baru ke fungsi yang sudah ada tanpa mengubah kode aslinya.

Penerapan closure biasanya digunakan untuk mengakses variabel dari scope luarnya, termasuk setelah scope luarnya selesai dieksekusi. *Closure* dapat membuat fungsi lebih fleksibel dan dapat digunakan kembali.

Syntactic sugar diterapkan untuk membuat penulisan yang lebih ringkas dan mudah dibaca tanpa mengubah fungsi dasarnya. Dalam *closure*, syntactic sugar digunakan untuk menulis closure dengan lebih intuitif.

Penerapan *decorator* digunakan untuk menambahkan fungsionalitas baru ke fungsi yang sudah ada tanpa harus mengubah kode aslinya. Dalam *closure*, *decorator* digunakan untuk menulis closure atau membuat closure dengan lebih terstruktur dan deklaratif.

3.2 Implementasi pada Penghitung *Discount*

Dalam sub-bab ini, akan fokus pada pembahasan kode hasil implementasi dari *decorator* dan *syntactic sugar* dalam aplikasi penghitung diskon.

3.2.1 *Decorator* dalam kode

Dalam implementasi penghitung *discount, decorator* berfungsi sebagai alat untuk modifikasi atau memperluas fungsionalitas dari suatu fungsi tanpa harus mengubah struktur dasar dari kode aslinya. Pada kode digunakan dua *decorator,* yaitu *'discount_decorator'* dan *'log_decorator'*, keduanya memiliki peran berbeda dalam aplikasi kami. *'Discount_decorator'* akan memberikan diskon pada total harga produk, dimana total harga tersebut dihitung oleh fungsi *'calculate_total_price'*. *Decorator* ini akan menerima argumen *'discount'* sebagai *keyword* yang akan memungkinkan pengguna menentukan jumlah diskon yang diberikan. Sementara itu, *'log_decorator'* kami buat untuk memantau pemanggilan fungsi *'calculate_total_price'*, ia akan mencetak informasi mengenai nama fungsi, argumen yang diterima serta nilai yang dikembalikan. Kedua fungsi *decorator* tersebut akan menciptakan fungsi yang lebih fleksibel dan mudah dikelola.

3.2.2 *Syntactic Sugar* dalam kode

Dalam implementasi penghitung *discount, syntactic sugar* digunakan untuk membuat penulisan fungsi inner menjadi lebih ringkas dan mudah dibaca. Dalam implementasinya pada penghitung discount, syntactic sugar kami gunakan pada penerapan decorate yaitu pada fungsi *'calculate_total_price'*. *Syntactic sugar* terlihat jelas menggunakan sintaks @ sebelum nama *decorator*, seperti *@log_decorator* dan *@discount_decorator*. Kita dapat dengan mudah serta langsung mendekorasi fungsi tersebut dengan decorator yang diinginkan. Tanpa *syntactic sugar,* penerapan dari decorator mungkin memerlukan penggunaan fungsi *wrapper* dan harus dimodifikasi menggunakan fungsi asli, yang akan menambahkan kompleksitas dan membuat kode menjadi kurang terbaca. *Syntactic sugar* dalam kode yang kami gunakan akan mempermudah proses penerapan *decorator*, meningkatkan keterbacaan dan membantu pengembangan dan membuat kode lebih efisien.

4. KESIMPULAN

Berdasarkan dari pembahasan yang ada, dapat disimpulkan bahwa fungsi *Decorator* dan *Syntactic Sugar* merupakan konsep penting dalam bahasa pemrograman Python yang digunakan untuk membantu menyederhanakan sintaks dan meningkatkan efisiensi kode. Fungsi *Decorator* dapat memudahkan pengguna untuk menambahkan fungsionalitas baru ke dalam fungsi yang sudah ada tanpa mengubah kode aslinya. *Syntactic Sugar* digunakan untuk membuat penulisan kode menjadi lebih ringkas dan mudah dibaca. Implementasi *Decorator* dan *Syntactic Sugar* pada artikel ini adalah untuk menghitung diskon, dimana penggunaan *Decorator* dan *Syntactic Sugar* adalah untuk memodifikasi fungsi, membuat kode lebih efisien, fleksibel, terstruktur, dan mudah dibaca.

References

Mwiti, D. (2024, 1). *Python Decorators Tutorial*. datacamp. https://www.datacamp.com/tutorial/decorators-python

Paulacy, L. (2024, 3 14). *PYTHON — Syntactic Sugar in Python*. Medium. https://laxfed.dev/python-syntactic-sugar-in-python-85c976e706ff

Puguh, A. (2024, 3 19). *Apa Itu Python? Pengertian, Fungsi, Kelebihan, dan Contohnya*. rumahweb. https://blog.rumahweb.com/python-adalah/#Apa_itu_Python

Richard, A. (2021, 1 24). *Memodifikasi Fungsi dengan Python Decorator (Bagian 1)*. Medium. https://agusrichard.medium.com/python-decorators-d5bbde71a730