Penerapan High Order Function Pada filter bilangan genap

Dearni Monica Manik, Nabila Anilda Zahrah, Abit Ahmad Oktarian,
Allya Nurul Islami Pasha, Yohana Manik, David Boby C.Nainggolan

Program Studi Sains Data Institut Teknologi Sumatera

Jl. Terusan Ryacudu, Way Huwi, Kec. Jatiagung, Kabupaten Lampung Selatan,

Lampung 35365

Email:
nabila.122450063@student.itera.ac.id dearni.122450075@student.itera.ac.id
abit.122450042@student.itera.ac.id allya.122450033@student.itera.ac.id
david.122450048@student.itera.ac.id yohana.122450126@student.itera.ac.id

Pendahuluan

Dalam dunia pemrograman fungsional, penggunaan high order function telah menjadi landasan utama dalam pengembangan perangkat lunak yang efisien dan mudah dipelihara. Salah satu contoh penerapan yang penting adalah penggunaan high order function untuk menyaring atau filter data, yang mana dapat secara signifikan meningkatkan kebersihan dan keterbacaan kode.Laporan ini mencermati penggunaan high order function, khususnya dalam konteks penyaringan bilangan genap dari sebuah kumpulan bilangan. Penerapan ini memiliki relevansi yang luas dalam berbagai kasus pemrograman, mulai dari analisis data hingga pengembangan aplikasi skala besar.

Dalam laporan ini, kami akan membahas bagaimana high order function, khususnya fungsi filter, dapat diterapkan secara efektif untuk menyaring bilangan genap dari sebuah array atau koleksi bilangan. Kami akan menyajikan pemahaman konseptual tentang high order function, dilanjutkan dengan penerapannya dalam bahasa pemrograman tertentu. Dan memberikan contoh penggunaan dalam situasi nyata. Tujuan dari laporan ini adalah untuk memberikan pemahaman yang lebih mendalam tentang penggunaan high order function dalam konteks penyaringan bilangan genap.

Metode

High Order Function merupakan konsep pemrograman yang memungkinkan fungsi berfungsi pada fungsi lain, yang mana fungsi ini memungkinkan pembuatan struktur kode yang lebih kompleks dan fleksibel. Konsep ini sangat berguna dalam bahasa pemrograman fungsional, di mana HOF digunakan untuk menyusun fungsi bersama-sama untuk melakukan tugas tertentu.
HOF biasanya didefinisikan sebagai sebuah fungsi yang menggunakan satu atau lebih fungsi sebagai argumen dan/atau menghasilkan fungsi sebagai hasilnya. Ada banyak abstraksi yang

menarik di sini. Fungsi ini memungkinkan pengguna merangkai dan menggabungkan fungsi dengan cara yang mirip dengan cara kita dapat menggabungkan fungsi di itertools untuk menghasilkan iterable baru.

Modul functools memiliki beberapa fungsi tingkat tinggi yang berguna, dan beberapa lainnya adalah fungsi bawaan. Map(), filter(), dan functools.reduce adalah fungsi dasar untuk High order function, dan sebagian besar bahasa pemrograman fungsional menggunakannya sebagai fungsi primitifnya, terkadang dengan nama lain. Seperti kari, blok penyusun hampir sama dengan map/filter/reduce.

Pembahasan

Berikut pembahasan kami mengenai penerapan High Order Function pada filter bilangan genap dengan kode sebagai berikut:

```python
#Filter Bilangan Genap
def is_even(num):
    return num % 2 == 0

def filter_list(predicate, lst):
    return [x for x in lst if predicate(x)]

# Higher order function untuk map
def square(num):
    return num ** 2

def map_list(func, lst):
    return [func(x) for x in lst]

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# buat Filter bilangan genap
even_numbers = filter_list(is_even, numbers)
print("Bilangan genap:", even_numbers)

# Buat utk fungsi kuadratnya
squared_numbers = map_list(square, numbers)
print("Kuadrat dari setiap bilangan:", squared_numbers)
```

Gambar 1

Kode pada Gambar 1 menunjukkan bagaimana fungsi tingkat tinggi dapat diterapkan dalam Python. Pertama, ada fungsi `is_even` yang menge mbalikan `True` jika bilangan tersebut genap. Kemudian, ada `filter_list`, yang menggunakan list comprehension untuk membuat daftar baru yang hanya berisi elemen-elemen yang memenuhi kondisi yang ditentukan oleh fungsi `predicate`. Selanjutnya, fungsi `square` mengembalikan kuadrat dari bilangan yang diberikan. `map_list` melakukan hal serupa dengan `filter_list`, namun menerapkan fungsi yang diberikan pada setiap elemen dalam daftar input.

Kemudian, daftar `numbers` digunakan sebagai input. Program pertama-tama menggunakan `filter_list` untuk menyaring bilangan genap dari `numbers`, dan hasilnya disimpan dalam variabel `even_numbers`. Selanjutnya, fungsi `square` diterapkan pada setiap elemen dari `numbers` menggunakan `map_list`, dan hasilnya disimpan dalam variabel `squared_numbers`. Akhirnya, dicetak daftar bilangan genap dan hasil kuadrat dari setiap bilangan dalam `numbers`seperti pada Gambar 2. Dengan menggunakan fungsi tingkat tinggi seperti `filter` dan `map`, program ini memanipulasi daftar dengan efisien dan membuat kode lebih mudah dibaca.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# buat Filter bilangan genap
even_numbers = filter_list(is_even, numbers)
print("Bilangan genap:", even_numbers)

# Buat utk fungsi kuadratnya
squared_numbers = map_list(square, numbers)
print("Kuadrat dari setiap bilangan:", squared_numbers)
```

```
Bilangan genap: [2, 4, 6, 8, 10]
Kuadrat dari setiap bilangan: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Gambar 2

Kesimpulan

Dari pembahasan yang telah disampaikan diatas, dapat disimpulkan bahwa penggunaan high order function (HOF) dalam pemrograman fungsional membawa manfaat yang signifikan. Dengan penggunaan HOF dalam penyaringan bilangan genap yang telah dilakukan, kita bisa melihat bahwa konsep ini memungkinkan pembuatan kode yang lebih terstruktur dan mudah dimengerti.

Dengan implementasi HOF pada contoh diatas, fungsi filter digunakan untuk mengekstraksi bilangan bulat yang memenuhi kriteria tertentu, sementara fungsi map digunakan untuk menerapkan operasi kuadrat pada setiap bilangan bulat. Dengan pendekatan ini, proses manipulasi data menjadi lebih terstruktur dan lebih mudah dimengerti yang dapat mempercepat pengembangan serta memperbaiki kualitas kode.

Secara keseluruhan , penggunaan HOF tidak hanya meningkatkan efisiensi dalam pemrosesan data, tetapi juga mendorong pembuatan kode yang lebih abstrak dan mudah dikelola. Ini memberikan pengembang kemampuan untuk menangani kompleksitas dengan lebih baik dan menghasilkan solusi dalam pengembangannya.