Penerapan High Order Function dalam Transaksi Konversi Mata Uang

Oktavia Nurwinda Puspitasari[1], Mutiara Dian Pitaloka[2], Syadza Puspadari Azhar[3], Kayla Amanda Sukma[4], Muhammad Wafi Raihan[5]

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

Email: oktavia.122450041@student.itera.ac.id mutiara.122450047@student.itera.ac.id syadza.122450072@student.itera.ac.id kayla.122450086@student.itera.ac.id muhammad.122450144@student.itera.ac.id

1. Pendahuluan

Pemrograman fungsional telah menjadi salah satu paradigma yang sangat efektif dalam sistem pengembangan yang kompleks dan berskalabel. salah satu konsep terpenting dalam pemrograman fungsional *Higher-Order Function* (HOF).

Higher-Order Function merupakan salah satu konsep terpenting pada paradigma pemrograman fungsional. Fungsi HOF sendiri dapat digunakan untuk membangun sistem yang lebih mudah beradaptasi dan efektif dalam mengenai transaksi yang melibatkan banyak mata uang. Salah satu penerapan konsep HOF yaitu dengan membuat program yang bertujuan untuk mengkonversi mata uang berdasarkan list atau data yang diberikan. Dalam kasus ini, kami akan menerapkan konsep HOF untuk mengkonversi mata uang USD ke dalam bentuk Indonesia Rupiah.

2. Metode

a. *High Order Function*

High Order Function (HOF) merupakan fungsi yang dapat menerima satu atau lebih sebagai argumen dan dapat mengembalikan suatu fungsi sebagai keluarannya. Dalam HOF terdapat tiga jenis fungsi yaitu menerima sebagai argumen, mengembalikan fungsi, dan menerima fungsi yang akan mengembalikan fungsi. Penggunaan HOF dapat memisahkan logika program menjadi lebih modular dimana hal ini dapat meningkatkan fleksibilitas, modularitas serta pengekspresian kode dalam pemrograman berbasis fungsi sehingga algortimanya lebih dapat dipahami. Pada kasus ini, HOF diterapkan dengan menggunakan fungsi map serta fungsi *lambda* untuk melakukan transaksi konversi mata uang [1].

b. Fungsi *Map*

Fungsi *map* merupakan salah satu fungsi *built-in* pada *high order function* dimana kita dapat menjalankan suatu fungsi dengan banyak nilai yang dimasukan. *Map* digunakan untuk membuat *array* yang baru dengan melakukan operasi pada setiap elemen yang terdapat pada *array*. Dengan fungsi *map* kita dapat membuat suatu koleksi baru dengan pengaplikasian elemen pada setiap objek yang dapat dilakukan iterasi yaitu seperti *range, list, tuple, dictionary, generator*, dll. Secara singkat fungsi

map akan melakukan pemetaan data sehingga jumlah dan panjang data map sama dengan objek *iterable*. Pada kasus ini fungsi map digunakan pada *convert currency* untuk setiap elemen pada *list transactions* [2].

c. *Fungsi Lambda*

Fungsi *lambda* dikenal juga sebagai *anonymous function* merupakan suatu ekspresi untuk pembuatan suatu fungsi tanpa harus memberikan nama pada fungsinya. Fungsi *lambda* bisa mempunyai lebih dari satu argumen dengan hanya satu ekspresi. Hal ini bertujuan supaya kita dapat membuat fungsi yang sederhana yang tidak memerlukan definisi yang terpisah. Dengan menggunakan fungsi *lambda*, kita dapat menyederhanakan pembuatan fungsi dimana kita tidak memerlukan pendefinisian dengan kata kunci "def". Oleh karena itu, fungsi *lambda* dapat menghasilkan fleksibilitas dalam penulisan kodenya. Pada kasus ini, fungsi map digunakan pada fungsi untuk *convert currency* dimana fungsi akan mengambil satu argumen untuk fungsi tersebut [3].

3. Pembahasan

```
# Fungsi untuk mengkonversi jumlah transaksi ke mata uang lain
def convert_currency(transaction, exchange_rate):
    return {'id': transaction['id'], 'amount': transaction['amount'] * exchange_rate, 'currency': 'IDR'}
```

Pada kode di atas, kita mendefinisikan sebuah fungsi convert_currency untuk mengkonversi jumlah transaksi ke mata uang lain. Dalam kasus ini kami akan mengkonversikan mata uang *US Dollar* ke rupiah. Fungsi ini akan mengembalikan nilai yang menunjukkan nilai 'id' pada *dictionary* transaction, nilai 'amount' pada *dictionary* yang akan dikalikan dengan fungsi 'exchange_rate', dan menampilkan nilai 'currency' yang digantikan dengan 'IDR'.

```
# Daftar transaksi
transactions = [
    {'id': 1, 'amount': 100, 'currency': 'USD'},
    {'id': 2, 'amount': 50, 'currency': 'USD'},
    {'id': 3, 'amount': 200, 'currency': 'USD'}
]
```

Kode di atas menunjukkan variabel 'transaction' yang berisi *dictionary* dengan variabel 'id', 'amount', dan 'currency'. Variabel 'id' menujukkan id uang yang akan di convert. Variabel 'amount' menunjukkan menunjukkan besar uang dalam *dollar*. Misalnya 'amount' 100 menunjukkan 100 USD, 'amount' 50 menunjukkan 50 USD, dan 'amount'

200 menunjukkan 200 USD. Dan variabel ‘currency’ untuk menunjukkan mata uang yang dimaksud. Dalam variabel ini merupakan *US Dollar*.

```
# Kurs mata uang: USD ke IDR
exchange_rate_usd_to_idr = 14000
```

Setelah itu, kita membuat variabel exchange_rate_usd_to_idr yang mendefinisikan besar mata uang rupiah dalam *dollar*. Saat ini, 1 US *dollar* bernilai setidaknya Rp 14.000. Variabel ini nantinya akan digunakan untuk mengkonversi list ‘transaction’ yang telah didefinisikan sebelumnya ke dalam bentuk IDR.

```
# Mengkonversi transaksi ke mata uang IDR
Transactions_idr = list(map(lambda transaction: convert_currency(transaction,
exchange_rate_usd_to_idr), transactions))
```

Dengan fungsi di atas, kita membuat list menggunakan fungsi map dengan fungsi *lambda* dengan variabel ‘transaction’. Dalam fungsi tersebut, kita akan menggunakan fungsi ‘convert_currency’ dengan argumen transaction dan exchange_rate_usd_to_idr sehingga fungsi tersebut akan memasukkan variabel *dictionary* ‘transaction’ dan variabel ‘amount’ akan dikalikan dengan 14.000 (nilai dari variabel exchange_rate_usd_to_idr). Fungsi map tersebut akan menggunakan list berbentuk *dictionary* yang ada dalam variabel ‘transaction’. Sehingga fungsi tersebut akan mengeluarkan nilai masing-masing variabel ‘transaction’ yang telah dikonversi ke dalam Indonesia Rupiah.

```
print(transactions_idr)
```

Setelah itu, kita menggunakan perintah print() untuk menjalankan fungsi transactions_idr yang telah dibuat sebelumnya untuk menampilkan hasil dari keluaran fungsi pada terminal. Pada kasus kali ini, keluaran dari variabel ‘transaction’ dapat dilihat dalam gambar berikut.

```
[{'id': 1, 'amount': 1400000, 'currency': 'IDR'},
{'id': 2, 'amount': 700000, 'currency': 'IDR'},
 {'id': 3, 'amount': 2800000, 'currency': 'IDR'}]
```

Gambar 1. Output dari fungsi transactions_idr

4. Kesimpulan

Dengan menggunakan konsep fungsi tingkat tinggi (HOF), fungsi map, dan fungsi *lambda*, pendekatan pemrograman fungsional memberikan fleksibilitas dan modularitas dalam pengembangan sistem yang kompleks. HOF memungkinkan pengembang memecah logika program menjadi komponen yang lebih kecil yang dapat diatur dengan baik, yang meningkatkan ekspresi kode dan memudahkan pemahaman.

Dengan menggunakan fungsi *map*, kita dapat menerapkan suatu fungsi pada setiap elemen dari suatu iterable. Ini mempercepat proses pemetaan data dan menghasilkan koleksi baru dengan nilai yang diubah sesuai dengan fungsi yang diterapkan. Fungsi *lambda*, di sisi lain, meningkatkan fleksibilitas dalam penulisan kode dengan memungkinkan pembuatan fungsi sederhana tanpa harus mendefinisikan nama.

Penerapan pada konversi mata uang dalam daftar transaksi menunjukkan bahwa HOF, fungsi map, dan fungsi *lambda* sangat efektif untuk meningkatkan efisiensi dan kualitas kode. Metode ini memungkinkan pengembang untuk mengelola transaksi yang melibatkan berbagai mata uang, meningkatkan adaptabilitas sistem, dan mempercepat pengembangan aplikasi yang kompleks.

5. Daftar Pustaka

[1] N. Arif, "High Order Function," 21 June 2021. [Online]. Available: https://kotakode.com/blogs/11626. [Accessed 1 May 2024].

[2] B. Priambodo, "Higher-order Function — Paradigma Fungsional Praktis," 20 July 2017. [Online]. Available: https://medium.com/paradigma-fungsional/higher-order-function-paradigma-fungsional-praktis-part-4-c836bd23a82. [Accessed 1 May 2024].

[3] A. Muhardian, "Belajar Python: Membuat Fungsi dengan Lambda Expression," 18 December 2019. [Online]. Available: https://www.petanikode.com/python-lambda/. [Accessed 1 May 2024].