Analisis Penerapan Higher Order Function Dalam Penentuan Indeks Nilai Akhir Suatu Matakuliah

Rahma Neliyana[1)], Eka Fidiya Putri[2)], Izza Lutfia[3)], Dea Mutia Risani[4)], Dinda Nababan[5)]
Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera
Email : rahma.122450036@student.itera.ac.id[1)], eka.122450045@student.itera.ac.id [2)], izza.122450090@student.itera.ac.id [3)], dea.122450099@student.itera.ac.id [4)], dinda.122450120@student.itera.ac.id [5)]

1. Pendahuluan

Dalam dunia pendidikan tinggi, penentuan indeks nilai akhir suatu mata kuliah merupakan proses penting yang mempengaruhi evaluasi akademik mahasiswa. Penggunaan metode yang efektif dan efisien dalam perhitungan nilai akhir menjadi krusial untuk memberikan gambaran yang akurat tentang pencapaian belajar mahasiswa. Salah satu pendekatan yang dapat digunakan dalam proses ini adalah penerapan *Higher Order Function*. *Higher Order Function* adalah konsep dalam pemrograman fungsional yang memungkinkan fungsi-fungsi untuk diperlakukan sebagai nilai, memungkinkan abstraksi yang lebih tinggi dan kemampuan untuk menulis kode yang lebih ringkas dan ekspresif.

Dalam konteks ini, analisis tentang penerapan *Higher Order Function* dalam penentuan indeks nilai akhir suatu mata kuliah menjadi relevan. Melalui pendekatan ini, kita dapat mengkaji bagaimana konsep *Higher Order Function* dapat diterapkan dalam proses penghitungan nilai akhir, termasuk bagaimana fungsi-fungsi yang lebih tinggi dapat digunakan untuk mengabstraksi logika perhitungan nilai berdasarkan berbagai kriteria evaluasi. Analisis ini bertujuan untuk mengevaluasi keefektifan, keefisienan, dan kebergunaan penggunaan *Higher Order Function* dalam konteks akademik, serta dampaknya terhadap hasil akhir evaluasi mahasiswa. Dengan demikian, studi ini dapat memberikan wawasan yang berharga tentang penerapan konsep pemrograman fungsional dalam konteks evaluasi akademik di perguruan tinggi.

2. Metode

Higher order function merupakan salah satu konsep terpenting pada paradigma pemrograman fungsional. *Higher order function* atau fungsi orde tinggi adalah fungsi yang menerima fungsi lain sebagai parameternya dan/atau mengembalikan fungsi lain sebagai keluarannya[1]. Ada beberapa jenis *higher order function* tertentu yang penting digunakan untuk memahami basis kode modern. Ini terutama berhubungan dengan iterasi atau penjumlahan daftar data. Berikut merupakan beberapa jenis higher order function yang sering digunakan dalam bahasa pemrograman *python*.

2.1. HOF Map

Operasi map memungkinkan Anda menerapkan fungsi ke setiap daftar elemen, lalu mengembalikan daftar baru dengan nilai baru tersebut. Melalui fungsi map, kita dapat menjalankan pemrograman dengan cara yang lebih bersih. Fungsi map akan menerima sebuah fungsi dan daftar sebagai argumen, kemudian mengembalikan daftar baru dengan fungsi tersebut untuk diterapkan pada setiap elemen daftar[2].

2.2. HOF Filter

Contoh lain dari *higher order function* yaitu filter. Filter akan memilah nilai yang ditentukan lalu dikembalikan menjadi sebuah array baru[3].

2.3. HOF Reduce

Operasi *reduce* memungkinkan Anda menghitung nilai tunggal berdasarkan fungsi yang digunakan untuk menggabungkan semua daftar elemen. Cara kerja dari fungsi ini yaitu hasil operasi pada elemen ke-i, akan dijadikan parameter untuk operasi pada elemen berikutnya (i+1), sehingga hasil operasi reduce adalah akumulasi operasi pada semua elemen. Pendeknya, *value* yang tadinya banyak akan menjadi satu saja. Dalam *HOF reduce*, dapat ditambahkan parameter kedua yaitu *initial value* sebagai nilai permulaan sebelum reduce dieksekusi[4].

3. Pembahasan

3.1. Kode dan Hasil Pemrograman

Berikut merupakan kode dan hasil pemrograman yang telah kami buat. Secara keseluruhan kode ini menjelaskan tentang penerapan *closure* menggunakan *decorator* pada pengelolaan akses admin dalam suatu aplikasi.

```
# Fungsi untuk menghitung nilai akhir berdasarkan bobot
def hitung_nilai_akhir(nilai_tugas, nilai_ujian, bobot_tugas=0.4, bobot_ujian=0.6):
    return (nilai_tugas * bobot_tugas) + (nilai_ujian * bobot_ujian)
```

Gambar 1. Menghitung Nilai Akhir Berdasarkan Bobot

Gambar 1 menjelaskan fleksibilitas bagi pengguna untuk menyesuaikan bobot tugas dan ujian sesuai keinginan mereka atau aturan penilaian kursus. Contohnya, jika suatu kursus lebih menekankan pada nilai ujian, pengguna dapat memberikan nilai yang lebih tinggi pada bobot_ujian. Sebaliknya, jika tugas memiliki peranan penting dalam menentukan nilai akhir, pengguna bisa meningkatkan nilai bobot_tugas. Secara keseluruhan, fungsi ini menyediakan metode yang sederhana untuk menghitung nilai akhir dengan memperhitungkan bobot yang dapat diatur sesuai kebutuhan untuk tugas dan ujian.

```
# Fungsi untuk menentukan indeks berdasarkan nilai akhir
def tentukan_indeks(nilai_akhir):
    if nilai_akhir >= 85:
        return 'A'
    elif nilai_akhir >= 70:
        return 'B'
    elif nilai_akhir >= 60:
        return 'C'
    else:
        return 'D'
```

Gambar 2. Menentukan Indeks Berdasarkan Nilai Akhir

Gambar 2 menjelaskan evaluasi singkat terhadap prestasi mahasiswa berdasarkan kisaran nilai tertentu. Misalnya, nilai akhir di atas 85 dapat dianggap sebagai pencapaian yang sangat baik, sedangkan nilai antara 60 dan 69 menunjukkan

pencapaian yang memenuhi syarat, dan nilai di bawah 60 mengindikasikan kebutuhan akan perbaikan atau bantuan tambahan. Fungsi ini juga menyediakan struktur yang jelas dan langsung untuk mengevaluasi kinerja berdasarkan standar yang ditetapkan, yang bermanfaat untuk memberikan umpan balik cepat kepada siswa atau untuk keperluan administratif dalam menentukan status akademik mereka.

```
# Data mahasiswa beserta nilai tugas dan ujian
data_mahasiswa = [
    {"nama": "John", "nilai_tugas": 80, "nilai_ujian": 75},
    {"nama": "Jane", "nilai_tugas": 90, "nilai_ujian": 85},
    {"nama": "Doe", "nilai_tugas": 70, "nilai_ujian": 65},
    {"nama": "Smith", "nilai_tugas": 85, "nilai_ujian": 80},
    {"nama": "Diana", "nilai_tugas": 95, "nilai_ujian": 90}
]
```

Gambar 3. Data Mahasiswa Nilai Tugas dan Nilai Ujian

Gambar 3 menjelaskan proses penggunaan fungsi yang telah didefinisikan sebelumnya untuk menghitung nilai akhir bagi setiap mahasiswa. Selain itu, dengan memanfaatkan fungsi lain yang telah dibuat sebelumnya, seperti fungsi untuk menentukan indeks prestasi berdasarkan nilai akhir, dapat dilakukan evaluasi terhadap kinerja setiap mahasiswa dalam bentuk kelas atau indeks prestasi. Dengan menggunakan data tersebut, dapat diperoleh gambaran yang jelas mengenai prestasi akademik dari masing-masing mahasiswa, dan informasi tersebut dapat digunakan untuk pengambilan keputusan lebih lanjut, seperti memberikan umpan balik kepada mahasiswa atau merancang langkah-langkah pendukung yang diperlukan untuk meningkatkan kinerja akademik mereka.

```
# Menggunakan higher order function (filter) untuk memfilter mahasiswa yang lulus
mahasiswa_lulus = list(filter(lambda mhs: hitung_nilai_akhir(mhs['nilai_tugas'], mhs['nilai_ujian']) >= 60, data_mahasiswa))
```

Gambar 4. Filter Mahasiswa yang Lulus

Gambar 4 menjelaskan bagaimana penggunaan fungsi filter() dengan lambda function memfasilitasi penyaringan data dengan kriteria yang dapat disesuaikan tanpa kebutuhan menulis fungsi terpisah. Dalam konteks ini, baris kode tersebut memberikan gambaran tentang cara yang efisien dalam mengevaluasi dan memproses data mahasiswa untuk mengidentifikasi mereka yang memenuhi persyaratan kelulusan dengan nilai akhir minimal 60.

```
# Menggunakan higher order function (map) untuk menghitung nilai akhir dan indeks semua mahasiswa
nilai_indeks_mahasiswa = list(map(lambda mhs: (hitung_nilai_akhir(mhs['nilai_tugas'], mhs['nilai_ujian']), tentukan_indeks(hitung_nilai_akhir(mhs['nilai_tugas'], mhs['nilai_ujian']))), data_mahasiswa))
```

Gambar 5. Fungsi map untuk menghitung nilai akhir

Gambar 5 menjelaskan bahwa kode tersebut menggunakan fungsi *higher-order map* yang digunakan untuk menerapkan dua operasi di setiap elemen "data_mahasiswa". Setiap iterasi menggunakan fungsi "nilai_hitung_akhir" yang

digunakan untuk menghitung nilai akhir mahasiswa, sedangkan nilai akhir dan indeks untuk setiap mahasiswa menggunakan fungsi "tentukan_indeks".

```
# Menampilkan hasil
print("Mahasiswa yang lulus:")
for mhs in mahasiswa_lulus:
    nilai_akhir = hitung_nilai_akhir(mhs['nilai_tugas'], mhs['nilai_ujian'])
    indeks = tentukan_indeks(nilai_akhir)
    print(f"{mhs['nama']}: {nilai_akhir} (Indeks: {indeks})")

print("\nNilai akhir dan indeks semua mahasiswa:")
for nilai, indeks in nilai_indeks_mahasiswa:
    print(f"Nilai: {nilai} (Indeks: {indeks})")
```

Gambar 6. Kode untuk menampilkan hasil

Gambar 6 menjelaskan bahwa kode tersebut digunakan untuk menampilkan hasil dari perhitungan nilai akhir dan indeks untuk mahasiswa yang lulus dan menampilkan nilai akhir serta indeks seluruh mahasiswa.

```
Mahasiswa yang lulus:
John: 77.0 (Indeks: B)
Jane: 87.0 (Indeks: A)
Doe: 67.0 (Indeks: C)
Smith: 82.0 (Indeks: B)
Diana: 92.0 (Indeks: A)

Nilai akhir dan indeks semua mahasiswa:
Nilai: 77.0 (Indeks: B)
Nilai: 87.0 (Indeks: A)
Nilai: 67.0 (Indeks: C)
Nilai: 82.0 (Indeks: B)
Nilai: 92.0 (Indeks: A)
```

Gambar 7. Output kode

Gambar 7 menjelaskan bahwa output menghasilkan 2 bagian yaitu bagian mahasiswa yang lulus serta bagian nilai akhir dan indeks semua mahasiswa, hal ini memberikan gambaran tentang prestasi akademik semua mahasiswa.

4. Kesimpulan

Penerapan higher order function dalam penentuan indeks nilai akhir suatu mata kuliah dilakukan dengan menggunakan HOF filter dan HOF map agar dapat dilakukannya operasi-operasi berulang namun dengan sintaks yang ringkas. Bobot pada tugas dan ujian ditentukan secara default (0.4 dan 0.6), akan tetapi dapat diubah jika diperlukan. Penentuan kriteria untuk menentukan indeks A,B,C dan D telah ditentukan berdasarkan ketetapan yang diperlukan. Pada kode pemrograman ini dibuat untuk

mengelola data mahasiswa, menghitung nilai akhir, serta menentukan indeks berdasarkan kriteria tertentu.

Referensi

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

[2] D. Soni, "Introduction to Higher-Order Functions," Better Programming, 18 October 2019. [Online]. Available: https://betterprogramming.pub/introduction-to-higher-order-functions-3ff0a05b40eb. [Accessed 2024].

[3] A. Nurdin, "Higher Order Function di Javascript," DEV, 13 Mei 2021. [Online]. Available: https://dev.to/iamalinurdin/higher-order-function-di-javascript-4jbg. [Accessed 2024].

[4] A. E. Pamungkas, "Berkenalan dengan Javascript Higher Order Function untuk Array," Akhdani Reka Solusi, 30 Mei 2018. [Online]. Available: https://blog.akhdani.co.id/2018/05/berkenalan-dengan-javascript-higher-order-function-untuk-array/. [Accessed 2024].