Funcional Programming

Functional Programming atau biasa disingkat dengan FP, termasuk dalam turunan deklaratif programming, dimana membuat kita tidak lagi berpikir bagaimana (how) `step-by-step` langkah penyelesaian, yang penting apa (what) yang dihasilkan.

Beberapa prinsip dasar yang harus terpenuhi:

1. Harus menggunakan pure function, yang mana harus menerima parameter dan me-return suatu hasil.
2. Function yang dipanggil tidak memperbolehkan adanya efek samping (side effect) yang berakibat pada perubahan data di luar scope (pada kenyataannya terkadang efek samping tidak dapat dihindari, namun diminimalisir)

Tentu saja pada funcional programming ini memiliki kelebihan dan kekurangannya,

antara lain :

Kelebihan:

- Karena fungsi, bersifat *Pure Functions*, kode menjadi lebih bampang dibaca, karena selalu menerima parameter dan mereturn hasil yg sesuai ekspektasi.

- Function menjadi lebih reusable

Kekurangan:

- Di beberapa kasus, ketika sistem sudah kompleks, penulisan FP ini menjadikannya tidak readable.

- Untuk pemula, paradigma ini, agak sulit diimplementasikan karena benar-benar berkutat dengan fungsi, dan tidak bisa dinalar ke dunia nyata.

Object Oriented Programming(OOP)

pemrograman berorientasi objek atau object oriented programming adalah paradigma berdasarkan konsep objek yang berisi struktur data dalam bidang yang dikenal sebagai atribut dan kode dalam bentuk prosedur, juga dikenal sebagai metode. Bahasa pemrograman berorientasi objek yang paling populer adalah berbasis kelas, yang berarti objek disajikan sebagai turunan dari kelas. Fitur utama pemrograman berorientasi objek adalah abstraksi, pewarisan, polimorfisme, dan enkapsulasi. Pemrograman berorientasi objek dapat berarti hal yang berbeda bagi orang yang berbeda. Bahasa pemrograman digunakan dengan baik saat Anda memiliki serangkaian operasi tetap, dan saat Anda melakukan beberapa operasi dengan perilaku umum dan varian berbeda. Pada saat yang sama, OOP bagus ketika Anda memiliki serangkaian operasi tetap, dan Anda terutama menambahkan hal-hal baru. Di sisi lain, bahasa fungsional digunakan saat Anda memiliki sekumpulan hal yang tetap, dan Anda terutama menambahkan operasi baru ke hal yang sudah ada.

Pemrograman berorientasi objek adalah cara menjaga data dan perilaku terbungkus dalam satu objek. Ini berarti semua data dan fungsi digabungkan menjadi satu dan diprogram pada panjang gelombang yang sama. Ini berbeda dengan bahasa pemrograman fungsional yang memiliki seperangkat asumsi bahwa fungsi tidak digabungkan dengan data. Kebanyakan orang yang menulis Python hampir pasti lebih baik menulis kode berorientasi objek. Oleh karena itu, jika Anda baru dalam pemrograman, saya akan merekomendasikan memulai dengan bahasa berorientasi objek dengan menulis kode dan algoritme pembelajaran mendalam yang disediakan oleh Python. Namun, Anda dapat menggunakan salah satu paradigma gaya fungsi, seperti pemrograman fungsional, menggantikan program berorientasi objek setelah menguasai konsepnya. Misalnya, Anda dapat menggunakan pemrograman fungsi alih-alih OOP dalam konteks apa pun dengan Python, karena komputer tidak peduli dengan paradigma apa pun.

PERBEDAAN

Dari kedua Program pada Python tersebut pasti terdapat perdaan di dalamnya, sehingga kita dapat menentukan program mana yang akan kita gunakan untuk membuat suatu kode program. Berikut beberapa perbedaanya:

Fungsional :

Menggunakan fungsi PURE

Menghidari ststus bersama dan data yang dapat diubah

Deklaratif

Ringkas dan lebih mudah untuk diuji

Berorientasi pada Objek :

Mengenkapsulasi fungsionalitas menggunakan kelas

Status disimpan di properti kelas

Imperatif

Dirancang untuk memanfaatkan trik pemrograman agar dapat diuji

PENGIMPLEMENTASIAN DARI OOP

- Pembuatan kelas dan objek: Dalam OOP, kita bisa membuat kelas sebagai blueprint atau cetak biru untuk objek. Kemudian, dari kelas tersebut, kita bisa membuat objek yang merupakan instansi dari kelas tersebut. Misalnya, jika kita membuat kelas Mobil, maka kita bisa membuat beberapa objek seperti mobil sedan, mobil pick-up, mobil sport, dan lain sebagainya.
- Encapsulation: OOP memungkinkan kita untuk menyembunyikan detail implementasi dari luar kelas atau objek dengan cara mengenkapsulasi data dan metode ke dalam kelas. Hal ini membuat kode menjadi lebih aman dan memudahkan perawatan atau perbaikan jika terjadi masalah. Contoh pengimplementasian encapsulation adalah dengan menggunakan akses modifier seperti private, protected, atau public.
- Inheritance: OOP memungkinkan kita untuk membuat kelas baru yang mewarisi sifat-sifat dan perilaku dari kelas yang sudah ada. Ini disebut inheritance atau pewarisan. Misalnya, jika kita membuat kelas Kendaraan, kita bisa membuat kelas baru seperti Mobil dan Motor yang mewarisi sifat-sifat dari kelas Kendaraan.
- Polymorphism: OOP memungkinkan kita untuk membuat metode yang sama namun dengan perilaku yang berbeda tergantung pada kelas yang digunakan. Ini disebut polimorfisme atau polymorphism. Misalnya, jika kita membuat metode drive() di kelas Kendaraan, kita bisa mengimplementasikan metode yang berbeda di kelas Mobil dan Motor tergantung pada cara mereka melakukan perjalanan.
- Abstraction: OOP memungkinkan kita untuk membuat kelas atau metode yang tidak perlu mengetahui detail implementasi dari kelas lain. Hal ini disebut abstraksi atau abstraction. Contoh pengimplementasian abstraction adalah dengan menggunakan interface atau abstract class.

PENGIMPLEMENTASIAN OOP DALAM KEHIDUPAN

- Aplikasi mobile: Sebagian besar aplikasi mobile dibangun dengan menggunakan OOP. Setiap halaman atau fitur dalam aplikasi diwakili oleh objek yang dibangun dari kelas tertentu. Misalnya, dalam aplikasi kesehatan, objek kelas Pasien akan digunakan untuk merepresentasikan setiap pasien yang terdaftar dalam aplikasi.
- Permainan video: Banyak permainan video modern dibangun dengan menggunakan OOP. Karakter dalam permainan video direpresentasikan oleh objek yang dibangun dari kelas karakter. Setiap karakter memiliki atribut seperti health point, damage, dan defense yang ditentukan oleh kelasnya.
- Sistem booking online: Sistem booking online seperti booking tiket pesawat, kereta, atau hotel juga dibangun dengan menggunakan OOP. Setiap objek seperti tiket atau pemesanan kamar diwakili oleh kelas yang berbeda. Misalnya, objek kelas TiketPesawat akan merepresentasikan tiket pesawat yang dipesan oleh pelanggan.
- Aplikasi e-commerce: Aplikasi e-commerce juga dibangun dengan menggunakan OOP. Setiap produk atau pesanan diwakili oleh objek yang dibangun dari kelas tertentu. Misalnya, objek kelas Produk akan merepresentasikan setiap produk yang dijual di aplikasi e-commerce.
- Sistem perbankan: Sistem perbankan juga dibangun dengan menggunakan OOP. Setiap akun atau transaksi diwakili oleh objek yang dibangun dari kelas tertentu. Misalnya, objek kelas Akun akan merepresentasikan setiap akun nasabah yang terdaftar dalam sistem perbankan.