LAPORAN DESAIN ANALISIS ALGORITMA

BRUTE FORCE

DISUSUN OLEH

MUHAMMAD DZAKY PUTRA UTOMO **L0123092**

MUH MISBAH ULHUDA **L0123085**

DOSEN DESAIN ANALISIS ALGORITMA:

FAJAR MUSLIM

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2024

```python
from flask import Flask, request, jsonify, render_template
from operation import calc

app = Flask(__name__)
```

- Flask digunakan sebagai tools dari python atau lebih tepatnya kita menggunakan template flask atau menginstal flask pada program

```python
@app.route('/')
def index():
    return render_template('index.html')
```

- Route yang digunakan untuk menjalankan "/" dimana "/" sendiri adalah home pada program
- Pada home di mana dia merender template pada index html atau sebagai tampilan dari web

```python
@app.route('/calculate', methods=['POST'])
def calculate():
    data = request.json
    input_data = data.get('input')
```

- Dimana code ini akan dipanggil setelah menggunakan memasukan input dan memencet kalkulator maka kode ini akan memanggil code pada program yang akan melakukan proses kalkulator

```python
if not input_data or len(input_data) != 4:
    return jsonify({"error": "Input must be a list of four numbers."}), 400

try:
    input_data = list(map(int, input_data))
except ValueError:
    return jsonify({"error": "All input values must be numbers."}), 400
```

- Dimana program memberi tahu apabila user tidak menginput 4 inputan maka program akan eror dan memberi tahu user kalau user harus menginput 4 angka

- Setelah terinput 4 angka maka program akan memberitahu apabila user tidak menginput angka maka program akan mengeluarkan statem kalau nomor yang dimasukkan haruslah berupa angka

```
    operation_result = calc(input_data)
    if operation_result:
        result = []
        count = 0
        for item in operation_result:
            count += 1
            item = item[1:-1]
            result.append(f"{item} = 20")
        return jsonify({"result": result, "total_solutions": count})
    else:
        return jsonify({"result": "Solution not found!"}), 200

if __name__ == '__main__':
    app.run(debug=True)
```

- Dimana code dimulai dengan menjalankan fungsi kalkulator, dimana user menginput Hasil dari fungsi ini disimpan dalam operation result. Fungsi kalkulator berperan untuk melakukan eksekusi berdasarkan inputan dari user tersebut
- program memeriksa apakah ada hasil yang dihasilkan dari kalkulator. Jika hasilnya tidak kosong, maka program akan melanjutkan memproses hasil tersebut. Jika kosong atau tidak ada hasil.
- Jika hasil ditemukan, kode membuat list kosong result untuk menyimpan hasil yang diproses dan menggunakan count yang digunakan untuk menghitung jumlah. Setiap item dalam hasil diproses dengan membuang karakter pertama dan terakhir dari string, lalu ditambahkan ke list result dalam format "<item> = 20". Setelah semua item diproses, hasil tersebut dikembalikan dalam format JSON bersama dengan total jumlah solusi.
- Jika tidak ada hasil kode akan mengembalikan respons JSON dengan pesan "Solution not found!" atau hasil tidak ditemukan, dengan begitu user bisa tau kalau program telah berjalan tetapi tidak ada hasil dari inputan.

```
class combination_1():
    def __init__(self, operand_1, operand_2):

        self._operand_1_ = operand_1
        self._operand_2_ = operand_2

        if (isinstance(operand_1, combination_1)):
            self.operand_1 = operand_1.dat
        else:
            self.operand_1 = operand_1
        if (isinstance(operand_2, combination_1)):
            self.operand_2 = operand_2.dat
        else:
            self.operand_2 = operand_2

    def __str__(self):
        return self.fmt.format(self._operand_1_, self._operand_2_)

    def __eq__(self, other):
        return not self.dat < other and not other < self.dat
```

- pembuatan class untuk kombinasi pertama dari 2 operand kelas combination 1 menerima dua operand melalui metode init. Operand tersebut disimpan sebagai atribut operand 1 dan operand 2. Jika operand merupakan instance dari kelas combination 1, .dat dari operand tersebut digunakan sebagai nilainya. Namun, jika operand bukan instance dari kelas yang sama, operand disimpan langsung ke dalam atribut operand 1 dan operand 2. Ini membuat kelas untuk rekursi operand.
- Metode eq digunakan untuk mengatur jalanya logika perbandingan kesetaraan antara dua objek dari kelas combination 1. Dua objek dianggap setara jika tidak ada yang lebih kecil dari satu sama lain .tetapi,apabila atribut dat juga tidak dijelaskan dalam kode ini, yang membuat bagaimana objek dibandingkan bergantung pada implementasi lebih lanjut dari atribut tersebut.
- Dalam kode ini terdapat dua atribut, fmt dan dat, Atribut fmt digunakan untuk pemformatan string di metode str, sedangkan atribut dat digunakan dalam logika perbandingan di metode eq.

```
class combination_1_sub(combination_1):
    def __init__(self, operand_1, operand_2):
        super().__init__(operand_1, operand_2)
        self.dat = self.operand_1 - self.operand_2
        self.fmt = '( {0} - {1} )'

# pengecekan kemungkinan kombinasi pertama dengan penjumlahan (2 operand)
class combination_1_mul(combination_1):
    def __init__(self, operand_1, operand_2):
        super().__init__(operand_1, operand_2)
        self.dat = self.operand_1 * self.operand_2
        self.fmt = '( {0} * {1} )'
```

- combination 1 pada kelas combination 1 add Pada kelas combination 1 add, konsep inheritance diterapkan dengan menurunkan properti dan metode dari kelas induknya,

combination 1. Dengan menurunkan dari kelas induk, kelas combination 1 add dengan begitu combination 1 dapat menggunakan apapun di kelas combination 1 tanpa harus mendefinisikan ulang.

- Konstruktor pada kelas combination 1 add menerima dua argumen, operand 1 dan operand 2, yang digunakan untuk operasi penjumlahan. Melalui init, konstruktor dari kelas induk dipanggil, sehingga inisialisasi di kelas induk tetap berjalan. Ini memastikan bahwa operand 1 dan operand 2 disiapkan dengan baik oleh kelas induk.
- Properti dat di dalam combination 1 add berfungsi untuk menyimpan hasil penjumlahan dari dua operand yang diterima, yaitu self.operand. Variabel ini akan berisi hasil operasi penjumlahan ketika kelas ini dieksekusi.
- Properti fmt adalah format string yang menyimpan representasi tekstual dari operasi penjumlahan, dalam bentuk "{0} + {1})". Di sini, {0} direpresentasikan dengan operand 1 dan {1} dengan operand 2, saat digunakan untuk melakukan operasi pada dua angka.
- combination 1 sub juga menerapkan konsep inheritance, menurunkan properti dan method dari kelas combination 1. Seperti pada combination 1_add, hal ini memungkinkan kelas combination 1_sub untuk menggunakan logika dan atribut yang direpresentasikan di kelas induk.
- combination 1 sub juga menerima dua argumen, operand 1 dan operand 2. Kemudian, konstruktor dari kelas induk dipanggil dengan super. init untuk memastikan bahwa kelas induk melakukan inisialisasi yang diperlukan, sama seperti pada combination 1 add.
- combination 1 sub, properti dat digunakan untuk menyimpan hasil pengurangan antara dua operand, yaitu self.operand Nilai ini akan berisi hasil operasi pengurangan ketika kelas diinstansiasi.
- combination 1 add, kelas combination 1 sub juga memiliki properti fmt yang berfungsi untuk menyimpan representasi tekstual dari operasi matematika yang dilakukan. format string digunakan untuk menggambarkan operasi pengurangan antara dua operand.

```
class combination_1_div(combination_1):
    def __init__(self, operand_1, operand_2):

        super().__init__(operand_1, operand_2)
        if self.operand_2 == 0:
            self.dat = 1000
        else:
            self.dat = self.operand_1 / self.operand_2
        self.fmt = '( {0} / {1} )'
```

- combination 1 div menurunkan properti dan metode dari kelas induknya, combination 1. Dengan menggunakan inheritance ini, kelas combination 1 div dapat memanfaatkan inisialisasi dan fitur yang sudah dipresentasikan di kelas induk.
- Setelah operand diinisialisasi, kode memeriksa apakah operand 2 bernilai 0 dengan kondisi if self.operand. Jika benar, properti dat diatur menjadi 1000. Jika operand 2 tidak sama dengan nol, maka operasi pembagian dilakukan, dan hasilnya disimpan dalam properti dat sebagai operand1 / operand 2.

```
# pengecekan kemungkinan kombinasi kedua dengan 3 operand (penggunaan tanda kurung)
def combination_2a(operand_1, operand_2, operand_3,
                   oper_1, oper_2):
    return oper_1(operand_1, oper_2(operand_2, operand_3))

# pengecekan kemungkinan kombinasi kedua dengan 3 operand (penggunaan tanda kurung)
def combination_2b(operand_1, operand_2, operand_3,
                   oper_1, oper_2):
    return oper_1(oper_2(operand_1, operand_2), operand_3)
```

- Combination *2a* mengombinasikan dua fungsi (oper_1 dan oper_2) dan pada tiga operand_3 dengan cara menerapkan operasi oper_2 terlebih dahulu pada operand_2 dan operan_3,lalu menggunakan hasilnya bersama pada operand_1 dalam operasi oper_1

```
# pengecekan kemungkinan kombinasi dengan semua operand dan 3 jenis operasi dengan tanda kurung
def combination_3a(operand_1, operand_2, operand_3, operand_4,
                   oper_1, oper_2, oper_3):
    return oper_1(
        combination_2a(operand_1, operand_2, operand_3, oper_2, oper_3), operand_4)
```

- Combination_3a menerima tujuh argumen yaitu operand_1, operand_2,operand_3 dan operand_4 tujuh,operand tersebut bertujuan sebagai input dalam operasi
- Oper_1, oper_2,oper_3:tiga fungsi yang akan diterapkan pada operand
- Fungsi pertama yang di panggil di dalam combination_3a adalah combination_2a (operand_1, operand_2, operand_3,oper_2, oper_3). Fungsi combination_2a ini adalah melakukan operasi oper_3 (operand_2, operand_3).
- Hasil dari combination 2a kemudian diteruskan sebagai argumen pertama ke dalam fungsi oper_1 bersama dengan operand 4.

```
# pengecekan kemungkinan kombinasi lain dengan semua operand
def combination_3b(operand_1, operand_2, operand_3, operand_4,
                oper_1, oper_2, oper_3):
    return oper_1(
        combination_2b(operand_1, operand_2, operand_3, oper_2, oper_3), operand_4)

# pengecekan kemungkinan kombinasi lain dengan semua operand
def combination_3c(operand_1, operand_2, operand_3, operand_4,
                oper_1, oper_2, oper_3):
    return oper_1(
        operand_1, combination_2a(operand_2, operand_3, operand_4, oper_2, oper_3))

# pengecekan kemungkinan kombinasi lain dengan semua operand
def combination_3d(operand_1, operand_2, operand_3, operand_4,
                oper_1, oper_2, oper_3):
    return oper_1(
        operand_1, combination_2b(operand_2, operand_3, operand_4, oper_2, oper_3))

# pengecekan kemungkinan kombinasi lain dengan semua operand
def combination_3e(operand_1, operand_2, operand_3, operand_4,
                oper_1, oper_2, oper_3):
    return oper_1(
        oper_2(operand_1, operand_2), oper_3(operand_3, operand_4))
```

- Fungsi combination 3b, fungsi ini mengembalikan hasil dari kombinasi operand dengan operasi tertentu
- Dalam kasus ini oper_1 hasil dari combination 2b, yang merupakan kombinasi dua operand pertama , serta operand lainnya,yang digabungkan menggunakan oper_2 dan oper_3.
- Fungsi combination 3c, fungsi ini bedanya adalah dalam urutan operand yang dioperasikan. kali ini oper_1 digunakan mengoperasikan operand 1 dan hasil dari combination 2a, yang , melakukan operasi pada operand 2, operand 3, dan operand 4
- Fungsi ini mencoba berbagai kombinasi operasi pada operand untuk mendapatkan hasil yang berbeda