

**LAPORAN PRAKTIKUM  
PEMROGRAMAN BERORIENTASI OBJEK (PBO) – [TUGAS BRP]**



Disusun Oleh

Andryano Shevchenko Limbong      123140205

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI  
INSTITUT TEKNOLOGI SUMATERA  
2025**

## Soal nomor 1

### a. Input soal

#### Tugas

- Di tugas ini, kalian akan membuat kalkulator sederhana yang menggunakan beberapa Dunder Method untuk melakukan operasi seperti:
  - $+$ ,  $-$ ,  $*$ ,  $/$ ,  $^$  (eksponen), dan  $\log$ .
  - Kalian bisa melihat contoh program Point untuk membuatnya.

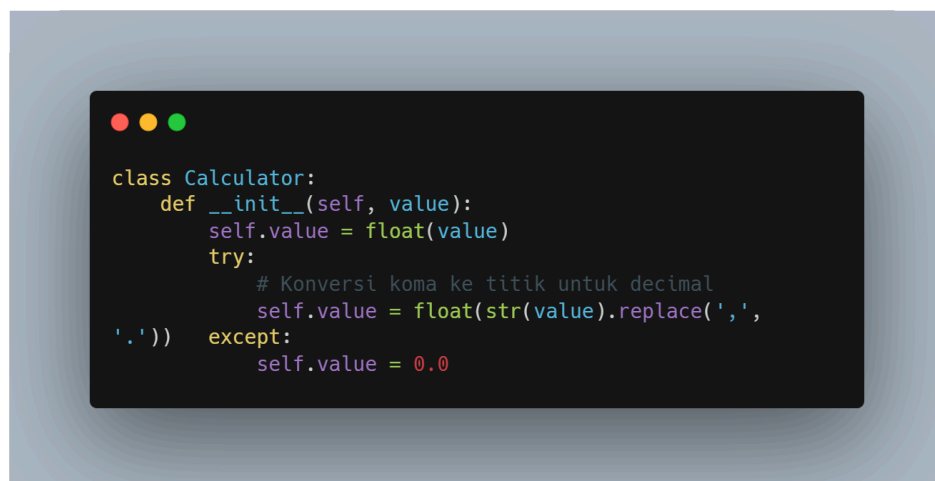
### b. Penjelasan

#### 1. Import Library Math dan Tkinter



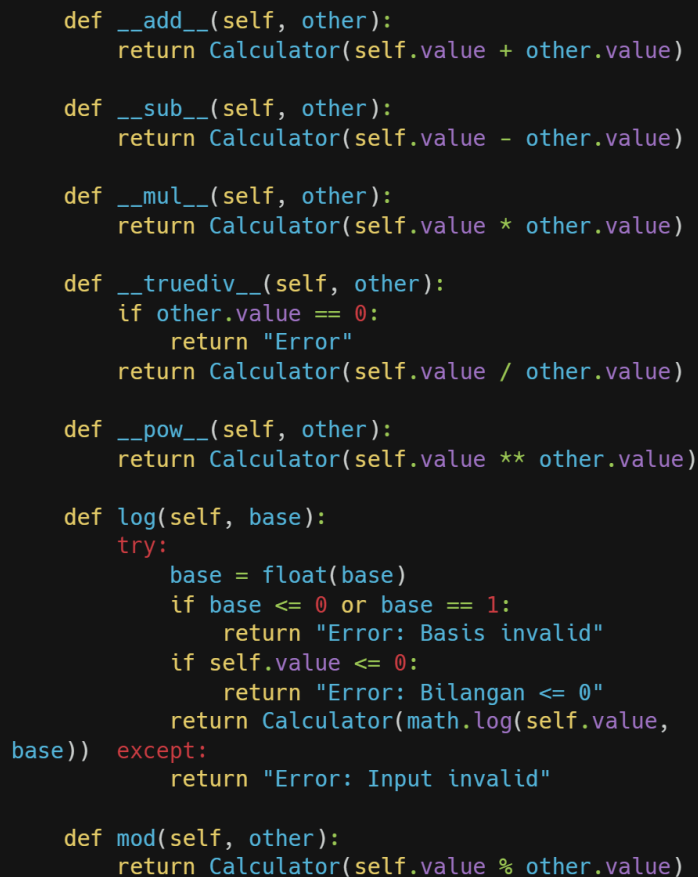
#### 2. Membuat Kelas Calculator

Kelas ini digunakan untuk melakukan operasi matematika. Setiap operasi diimplementasikan dengan Dunder Method.



Konstruktor `__init__` mengonversi input ke float. Jika pengguna memasukkan angka dengan koma (,) sebagai desimal, akan diubah ke titik (.). Jika input tidak valid, maka nilai default adalah 0.0.

### 3. Menambahkan Operasi Matematika



```
def __add__(self, other):
    return Calculator(self.value + other.value)

def __sub__(self, other):
    return Calculator(self.value - other.value)

def __mul__(self, other):
    return Calculator(self.value * other.value)

def __truediv__(self, other):
    if other.value == 0:
        return "Error"
    return Calculator(self.value / other.value)

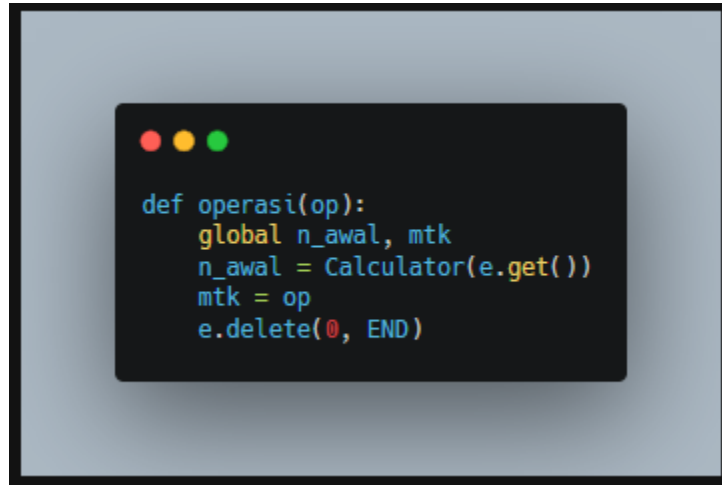
def __pow__(self, other):
    return Calculator(self.value ** other.value)

def log(self, base):
    try:
        base = float(base)
        if base <= 0 or base == 1:
            return "Error: Basis invalid"
        if self.value <= 0:
            return "Error: Bilangan <= 0"
        return Calculator(math.log(self.value,
base))
    except:
        return "Error: Input invalid"

def mod(self, other):
    return Calculator(self.value % other.value)
```

Setiap operasi dikemas dalam metode Dunder (`__add__`, `__sub__`, `__mul__`). Metode `__truediv__` menangani error jika pembagian oleh nol. `mod()` digunakan untuk sisa bagi (%). `__pow__(x^2)` untuk operasi pangkat dan metode logaritma untuk  $a \log b = c$ .

4. Operasi untuk menyimpan angka pertama dan jenis operasi.




```
def operasi(op):  
    global n_awal, mtk  
    n_awal = Calculator(e.get())  
    mtk = op  
    e.delete(0, END)
```

operasi(op) berfungsi untuk menyimpan angka pertama dan jenis operasi.

5. Operasi Matematika dengan Code yang panjang (perlu dijelaskan lebih lanjut)

#### Pangkat (\_\_pow\_\_)



```
def pangkat():  
    global n_awal  
    n_awal = Calculator(e.get())  
    e.delete(0, END)  
    e.insert(0, n_awal.__pow__(Calculator(2)).value)
```

Menghitung **pangkat dua** dari angka yang dimasukkan.

e.insert(0, n\_awal.\_\_pow\_\_(Calculator(2)).value) dapat dibaca sebagai  $10(^2)$

#### Logaritma

```

def logaritma():
    global n_awal
    try:
        # Ambil dan parse input
        input_text = e.get()
        if ',' not in input_text:
            raise ValueError("Format: [basis],[bilangan]\nContoh: 2,8")

        basis, bilangan = map(float, input_text.split(','))

        # Validasi input
        if basis <= 0 or basis == 1:
            raise ValueError("Basis harus >0 dan ≠1")
        if bilangan <= 0:
            raise ValueError("Bilangan harus >0")

        # Hitung logaritma
        n_awal = Calculator(bilangan)
        hasil = n_awal.log(basis)

        # Tampilkan hasil
        e.delete(0, END)
        e.insert(0, hasil.value if isinstance(hasil, Calculator) else
hasil)

```

Pengguna memasukkan basis dan bilangan dalam format basis,bilangan (misalnya 2,8). Jika input tidak valid, akan menampilkan pesan error dan constraints sebagai berikut:

- Harus menggunakan koma (,) untuk memisahkan basis dan bilangan.
- Jika pengguna hanya memasukkan satu angka atau menggunakan titik (.) sebagai pemisah, program akan menampilkan error.
- Basis harus lebih dari 0 ( $\text{basis} > 0$ ) karena logaritma tidak didefinisikan untuk basis nol atau negatif.
- Basis tidak boleh 1 ( $\text{basis} \neq 1$ ) karena logaritma basis 1 selalu tak terdefinisi ( $\log_1(x)$  tidak valid).

## samadengan

```
def samadengan():
    global n_awal, mtk
    n_akhir = Calculator(e.get())
    e.delete(0, END)
    if mtk == "Penjumlahan":
        e.insert(0, (n_awal + n_akhir).value)
    elif mtk == "Pengurangan":
        e.insert(0, (n_awal - n_akhir).value)
    elif mtk == "Perkalian":
        e.insert(0, (n_awal * n_akhir).value)
    elif mtk == "Pembagian":
        hasil = n_awal / n_akhir
        e.insert(0, hasil.value if isinstance(hasil, Calculator) else
        hasil)
    elif mtk == "SisaBagi":
        e.insert(0, (n_awal.mod(n_akhir)).value)
```

- Menjalankan operasi yang telah dipilih sebelumnya (+, -, x, /, %).
- Jika hasilnya error (pembagian 0), akan ditampilkan "Error".

## Hapus

```
def hapus():
    e.delete(0,
    END)
```

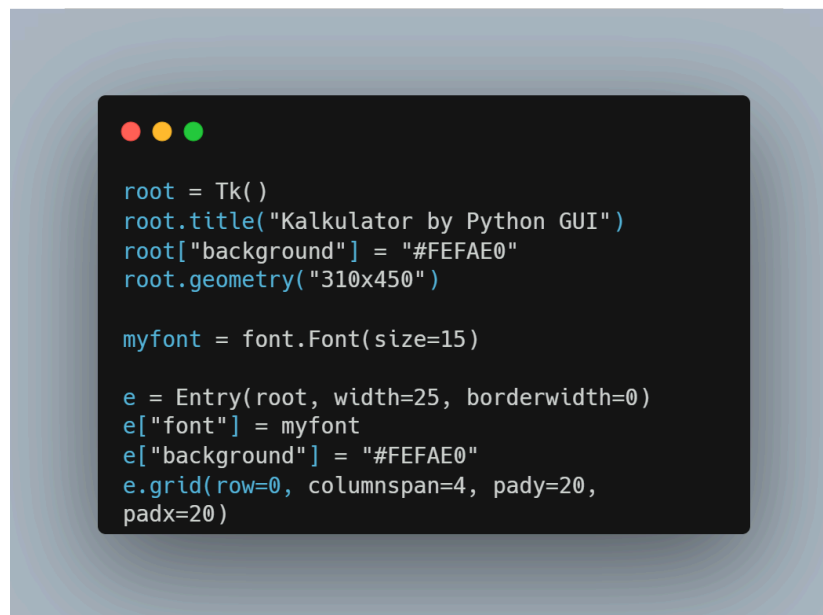
- e adalah Entry widget dari Tkinter (tempat pengguna memasukkan angka).
- delete(0, END) menghapus semua teks dari indeks pertama (0) hingga terakhir (END).
- Setelah dieksekusi, **Entry menjadi kosong**.

## 6. Fungsi Tombol



- cetak(nilai) menambahkan angka yang ditekan ke input. Misalnya, jika menekan 2, maka layar akan menampilkan 2.

## 7. Membuat Antarmuka (GUI)



- Tk() membuat jendela utama.
- title() memberi nama kalkulator.
- background mengatur warna.
- geometry("310x450") menentukan ukuran jendela.
- Entry() membuat kotak input angka.
- grid(row=0, columnspan=4) membuat input berada di baris pertama.

## 8. Membuat Tombol dan Menempatkan di GUI python

```
# Tombol angka
tombol = []
for i in range(10):
    tombol.append(Button(root, text=str(i), padx=30, bg="#798645", fg="white", pady=20, command=lambda
x=i: cetak(x)))

tombol[1].grid(row=3, column=0, pady=2)
tombol[2].grid(row=3, column=1, pady=2)
tombol[3].grid(row=3, column=2, pady=2)
tombol[4].grid(row=2, column=0, pady=2)
tombol[5].grid(row=2, column=1, pady=2)
tombol[6].grid(row=2, column=2, pady=2)
tombol[7].grid(row=1, column=0, pady=2)
tombol[8].grid(row=1, column=1, pady=2)
tombol[9].grid(row=1, column=2, pady=2)
tombol[0].grid(row=4, column=1, pady=2)

koma = Button(root, text=",", padx=30, bg="#798645", fg="white", pady=20, command=lambda: cetak(","))
koma.grid(row=4, column=2, pady=2)

# Tombol operasi
tam = Button(root, text="+", padx=30, bg="#626F47", fg="white", pady=20, command=lambda:
operasi("Penjumlahan"))
kur = Button(root, text="-", padx=30, bg="#626F47", fg="white", pady=20, command=lambda:
operasi("Pengurangan"))
bag = Button(root, text="/", padx=30, bg="#626F47", fg="white", pady=20, command=lambda:
operasi("Pembagian"))
kal = Button(root, text="x", padx=30, bg="#626F47", fg="white", pady=20, command=lambda:
operasi("Perkalian"))
pang = Button(root, text="x^2", padx=30, bg="#626F47", fg="white", pady=20, command=pangkat)
log = Button(root, text="a log b", padx=25, bg="#626F47", fg="white", pady=20, command=logaritma)
sisbag = Button(root, text="%", padx=30, bg="#626F47", fg="white", pady=20, command=lambda:
operasi("SisaBagi"))
hap = Button(root, text="C", padx=30, bg="#626F47", fg="white", pady=20, command=hapus)
equal = Button(root, text="=", padx=60, bg="#31511E", fg="white", pady=20, command=samadengan)

# Menempatkan tombol
tam.grid(row=1, column=3, pady=2)
kur.grid(row=2, column=3, pady=2)
bag.grid(row=3, column=3, pady=2)
kal.grid(row=4, column=3, pady=2)
hap.grid(row=6, column=0, pady=2)
equal.grid(row=5, column=1, columnspan=2)
pang.grid(row=4, column=0, pady=2)
log.grid(row=5, column=3, pady=2)
sisbag.grid(row=5, column=0, pady=2)

root.mainloop()
```

- Membuat tombol operasi (+, -, x, /, %,  $x^2$ , a log b).
- Menggunakan command=lambda



### c. Source Code

```
from tkinter import *
import tkinter.font as font
import math

class Calculator:
    def __init__(self, value):
        self.value = float(value)
        try:
            # Konversi koma ke titik untuk decimal
            self.value = float(str(value).replace(',', '.'))
        except:
            self.value = 0.0

    def __add__(self, other):
        return Calculator(self.value + other.value)

    def __sub__(self, other):
        return Calculator(self.value - other.value)

    def __mul__(self, other):
        return Calculator(self.value * other.value)

    def __truediv__(self, other):
        if other.value == 0:
            return "Error"
        return Calculator(self.value / other.value)

    def __pow__(self, other):
        return Calculator(self.value ** other.value)

    def log(self, base):
        try:
            base = float(base)
            if base <= 0 or base == 1:
                return "Error: Basis invalid"
            if self.value <= 0:
                return "Error: Bilangan <= 0"
            return Calculator(math.log(self.value, base))
        except:
            return "Error: Input invalid"

    def mod(self, other):
        return Calculator(self.value % other.value)

root = Tk()
root.title("Kalkulator by Python GUI")
root["background"] = "#FEFAE0"
root.geometry("310x450")

myfont = font.Font(size=15)

e = Entry(root, width=25, borderwidth=0)
e["font"] = myfont
e["background"] = "#FEFAE0"
e.grid(row=0, columnspan=4, pady=20, padx=20)

def cetak(nilai):
    current = e.get()
    e.delete(0, END)
    e.insert(0, str(current) + str(nilai))

def operasi(op):
    global n_awal, mtk
    n_awal = Calculator(e.get())
    mtk = op
    e.delete(0, END)

def pangkat():
    global n_awal
    n_awal = Calculator(e.get())
    e.delete(0, END)
    e.insert(0, n_awal.__pow__(Calculator(2)).value)
```

```

def logaritma():
    global n_awal
    try:
        # Ambil dan parse input
        input_text = e.get()
        if ',' not in input_text:
            raise ValueError("Format: [basis],[bilangan]\nContoh: 2,8")

        basis, bilangan = map(float, input_text.split(','))

        # Validasi input
        if basis <= 0 or basis == 1:
            raise ValueError("Basis harus >0 dan ≠1")
        if bilangan <= 0:
            raise ValueError("Bilangan harus >0")

        # Hitung logaritma
        n_awal = Calculator(bilangan)
        hasil = n_awal.log(basis)

        # Tampilkan hasil
        e.delete(0, END)
        e.insert(0, hasil.value if isinstance(hasil, Calculator) else hasil)

    except Exception as err:
        e.delete(0, END)
        e.insert(0, str(err))

def hapus():
    e.delete(0, END)

def samadengan():
    global n_awal, mtk
    n_akhir = Calculator(e.get())
    e.delete(0, END)
    if mtk == "Penjumlahan":
        e.insert(0, (n_awal + n_akhir).value)
    elif mtk == "Pengurangan":
        e.insert(0, (n_awal - n_akhir).value)
    elif mtk == "Perkalian":
        e.insert(0, (n_awal * n_akhir).value)
    elif mtk == "Pembagian":
        hasil = n_awal / n_akhir
        e.insert(0, hasil.value if isinstance(hasil, Calculator) else hasil)
    elif mtk == "SisaBagi":
        e.insert(0, (n_awal.mod(n_akhir)).value)

# Tombol angka
tombol = []
for i in range(10):
    tombol.append(Button(root, text=str(i), padx=30, bg="#798645", fg="white", pady=20, command=lambda
x=i: cetak(x)))

tombol[1].grid(row=3, column=0, pady=2)
tombol[2].grid(row=3, column=1, pady=2)
tombol[3].grid(row=3, column=2, pady=2)
tombol[4].grid(row=2, column=0, pady=2)
tombol[5].grid(row=2, column=1, pady=2)
tombol[6].grid(row=2, column=2, pady=2)
tombol[7].grid(row=1, column=0, pady=2)
tombol[8].grid(row=1, column=1, pady=2)
tombol[9].grid(row=1, column=2, pady=2)
tombol[0].grid(row=4, column=1, pady=2)

koma = Button(root, text=".", padx=30, bg="#798645", fg="white", pady=20, command=lambda: cetak("."))
koma.grid(row=4, column=2, pady=2)


# Tombol operasi
tam = Button(root, text="+", padx=30, bg="#626F47", fg="white", pady=20, command=lambda:
operasi("Penjumlahan"))
kur = Button(root, text="-", padx=30, bg="#626F47", fg="white", pady=20, command=lambda:
operasi("Pengurangan"))
bag = Button(root, text="/", padx=30, bg="#626F47", fg="white", pady=20, command=lambda:
operasi("Pembagian"))
kal = Button(root, text="x", padx=30, bg="#626F47", fg="white", pady=20, command=lambda:
operasi("Perkalian"))
pang = Button(root, text="x²", padx=30, bg="#626F47", fg="white", pady=20, command=pangkat)
log = Button(root, text="a log b", padx=25, bg="#626F47", fg="white", pady=20, command=logaritma)
sisbag = Button(root, text="%", padx=30, bg="#626F47", fg="white", pady=20, command=lambda:
operasi("SisaBagi"))
hap = Button(root, text="C", padx=30, bg="#626F47", fg="white", pady=20, command=hapus)
equal = Button(root, text="=", padx=60, bg="#31511E", fg="white", pady=20, command=samadengan)

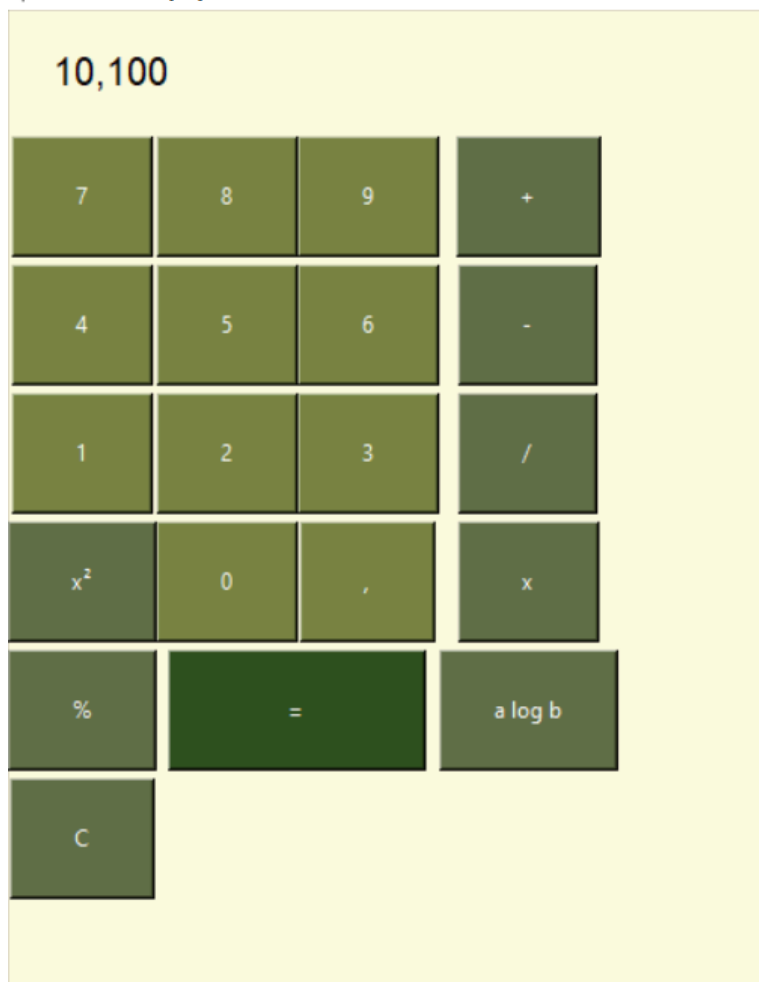
```

```
# Menempatkan tombol
tam.grid(row=1, column=3, pady=2)
kur.grid(row=2, column=3, pady=2)
bag.grid(row=3, column=3, pady=2)
kal.grid(row=4, column=3, pady=2)
hap.grid(row=6, column=0, pady=2)
equal.grid(row=5, column=1, columnspan=2)
pang.grid(row=4, column=0, pady=2)
log.grid(row=5, column=3, pady=2)
sisbag.grid(row=5, column=0, pady=2)

root.mainloop()
```

#### d. Hasil

 Kalkulator by Python GUI



## Soal Nomor 2

### a. Input Soal

Dalam tugas ini, kalian akan mensimulasikan pewarisan golongan darah anak dari orang tua. Untuk tugas ini, kalian akan membuat 3 kelas:

- i. **Father**
  - ii. **Mother**
  - iii. **Child**
- Kelas **Father** dan **Mother** akan memiliki properti **blood\_types**, yang nantinya akan diinput oleh pengguna.
  - Kelas **Child** akan menerima properti tersebut, memilih salah satu alel secara acak dari setiap orang tua, dan menentukan golongan darahnya.
  - Probabilitas pemilihan alel adalah 50-50 untuk ayah dan ibu.

### b. Penjelasan

1. Impor Library Random dan Penjelasan Kelas **Father** dan **Mother**

```
import random

class Father:
    def __init__(self, blood_type):
        self.blood_type = blood_type
        self.alleles = self.get_alleles()

    def get_alleles(self):
        return
self.blood_type_to_alleles(self.blood_type)
    @staticmethod
    def blood_type_to_alleles(blood_type):
        return {
            'A': ['A', 'O'],
            'B': ['B', 'O'],
            'AB': ['A', 'B'],
            'O': ['O', 'O']
        }[blood_type]

class Mother:
    def __init__(self, blood_type):
        self.blood_type = blood_type
        self.alleles = self.get_alleles()

    def get_alleles(self):
        return
self.blood_type_to_alleles(self.blood_type)
    @staticmethod
    def blood_type_to_alleles(blood_type):
        return {
            'A': ['A', 'O'],
            'B': ['B', 'O'],
            'AB': ['A', 'B'],
            'O': ['O', 'O']
        }[blood_type]
```

Fungsi `__init__`

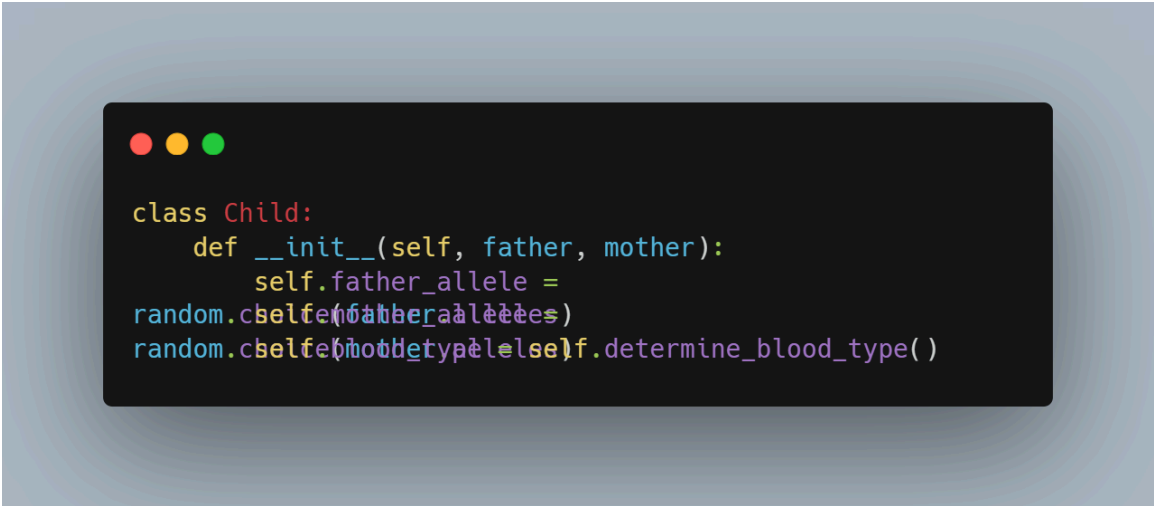
- Menerima golongan darah (`blood_type`).
- Menentukan dua alel yang mungkin dimiliki berdasarkan golongan darah.

Fungsi `blood_type_to_alleles()`

c. Mengonversi golongan darah ke dalam pasangan alel:

- i.  $A \rightarrow ['A', 'O']$
- ii.  $B \rightarrow ['B', 'O']$
- iii.  $AB \rightarrow ['A', 'B']$
- iv.  $O \rightarrow ['O', 'O']$

2. Penjelasan Kelas `Child`



```
class Child:
    def __init__(self, father, mother):
        self.father_allele =
random.choice(father.alleles)
random.choice(mother.alleles)
self.determine_blood_type()
```

- Memilih alel secara acak dari ayah (`father.alleles`) dan ibu (`mother.alleles`).
- Menentukan golongan darah anak menggunakan fungsi `determine_blood_type()`.

### 3. Fungsi determine\_blood\_type()

```
def determine_blood_type(self):
    alleles = {self.father_allele,
self.mother_allele}
    if alleles == {'A', 'B'}:
        return 'AB'
    elif 'A' in alleles and 'O' in alleles:
        return 'A'
    elif 'B' in alleles and 'O' in alleles:
        return 'B'
    elif alleles == {'A', 'A'}:
        return 'A'
    elif alleles == {'B', 'B'}:
        return 'B'
    elif alleles == {'O'}:
        return 'O'
    else:
        return 'Unknown'
```

Aturan pewarisan golongan darah:

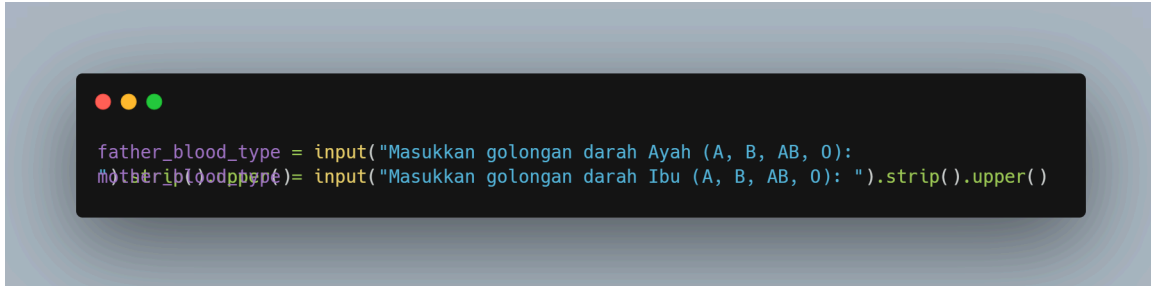
- Jika alelnya A dan B → Golongan darah AB
- Jika ada A dan O → Golongan darah A
- Jika ada B dan O → Golongan darah B
- Jika keduanya A → Golongan darah A
- Jika keduanya B → Golongan darah B
- Jika keduanya O → Golongan darah O

### 4. Menampilkan Hasil

```
def display_info(self):
    print(f"Golongan darah anak: {self.blood_type} (Alel: {self.father_allele}, {self.mother_allele})")
```

Menampilkan golongan darah anak dan alel yang diwarisi dari orang tua.

## 5. Input dari Pengguna



```
father_blood_type = input("Masukkan golongan darah Ayah (A, B, AB, O): ")
mother_blood_type = input("Masukkan golongan darah Ibu (A, B, AB, O): ").strip().upper()
```

Pengguna memasukkan golongan darah ayah dan ibu.

## 6. Membuat Objek dan Menjalankan Program



```
dad = Father(father_blood_type)
mom = Mother(mother_blood_type)
child = Child(dad, mom)
child.display_info()
```

- Membuat objek Father dan Mother berdasarkan input pengguna.
- Membuat objek Child, yang secara otomatis menentukan golongan darah anak.
- Menampilkan hasil menggunakan display\_info().

### c. Source Code

```
import random

class Father:
    def __init__(self, blood_type):
        self.blood_type = blood_type
        self.alleles = self.get_alleles()

    def get_alleles(self):
        return self.blood_type_to_alleles(self.blood_type)

    @staticmethod
    def blood_type_to_alleles(blood_type):
        return {
            'A': ['A', 'O'],
            'B': ['B', 'O'],
            'AB': ['A', 'B'],
            'O': ['O', 'O']
        }[blood_type]

class Mother:
    def __init__(self, blood_type):
        self.blood_type = blood_type
        self.alleles = self.get_alleles()

    def get_alleles(self):
        return self.blood_type_to_alleles(self.blood_type)

    @staticmethod
    def blood_type_to_alleles(blood_type):
        return {
            'A': ['A', 'O'],
            'B': ['B', 'O'],
            'AB': ['A', 'B'],
            'O': ['O', 'O']
        }[blood_type]

class Child:
    def __init__(self, father, mother):
        self.father_allele = random.choice(father.alleles)
        self.mother_allele = random.choice(mother.alleles)
        self.blood_type = self.determine_blood_type()

    def determine_blood_type(self):
        alleles = {self.father_allele, self.mother_allele}
        if alleles == {'A', 'B'}:
            return 'AB'
        elif 'A' in alleles and 'O' in alleles:
            return 'A'
        elif 'B' in alleles and 'O' in alleles:
            return 'B'
        elif alleles == {'A', 'A'}:
            return 'A'
        elif alleles == {'B', 'B'}:
            return 'B'
        elif alleles == {'O'}:
            return 'O'
        else:
            return 'Unknown'

    def display_info(self):
        print(f"Golongan darah anak: {self.blood_type} (Alel: {self.father_allele}, {self.mother_allele})")

father_blood_type = input("Masukkan golongan darah Ayah (A, B, AB, O): ").strip().upper()
mother_blood_type = input("Masukkan golongan darah Ibu (A, B, AB, O): ").strip().upper()

dad = Father(father_blood_type)
mom = Mother(mother_blood_type)

child = Child(dad, mom)
child.display_info()
```



#### d. Hasil

```
PS D:\VSCode\Phyton> python -u "d:\VSCode\Phyton\tempCodeRunnerFile.py"
Masukkan golongan darah Ayah (A, B, AB, O): A
Masukkan golongan darah Ibu (A, B, AB, O): B
Golongan darah anak: AB (Alel: A, B)
PS D:\VSCode\Phyton>
```

#### Lampiran dan Referensi

1. Referensi Video Youtube : <https://www.youtube.com/watch?v=-isfRqTR-Qo>
2. Referensi Repository Github : [https://github.com/ReihanWudd/tkinter\\_calculator.git](https://github.com/ReihanWudd/tkinter_calculator.git)
3. Link Percakapan LLM :  
<https://chatgpt.com/share/67d92b2d-d180-8012-8e28-1f344b943f78>

**(Jika menggunakan LLM atau Referensi website dalam pembuatan laporan, baik untuk generate code ataupun penulisan text silahkan sertakan dokumentasinya bisa berupa link ataupun screenshot percakapan. Jika tidak melampirkan dan ketahuan menggunakan nilai = 0)**

#### Contoh :

1. Link Percakapan LLM
2. Web Referensi - DuniaIlkom