

LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK (PBO) – [TUGAS 4]



Disusun Oleh
Reza Chairul Manam
120140086

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

1 Soal Nomor 1 — Menghitung Akar Kuadrat

1.1 Input Soal

Di tugas ini, kalian akan membuat kalkulator sederhana yang menggunakan beberapa **Dunder Method** untuk melakukan operasi seperti: Program ini meminta pengguna untuk memasukkan angka. Jika inputnya bukan angka, program akan menampilkan pesan "Input tidak valid. Harap masukkan angka yang valid." Jika angka yang dimasukkan negatif atau nol, program akan menampilkan pesan error yang relevan, seperti "Akar kuadrat dari nol tidak diperbolehkan." Jika inputnya valid (angka positif), program akan menghitung dan menampilkan akar kuadratnya. Terapkan exception dan error handling dalam menangani eror yang terjadi di dalam program.

1.2 Source Code

```
# Tugas 1:
import math

class SquareRootCalculator:
    def __init__(self):
        pass

    def get_input(self):
        while True:
            try:
                user_input = input("Masukkan angka: ")
                number = float(user_input)
                if number < 0:
                    print("Input tidak valid. Harap masukkan angka positif.")
                    continue
                elif number == 0:
                    print("Error: Akar kuadrat dari nol tidak diperbolehkan.")
                    continue
                return number
            except ValueError:
                print("Input tidak valid. Harap masukkan angka yang valid.")

    def calculate_square_root(self, number):
```

```

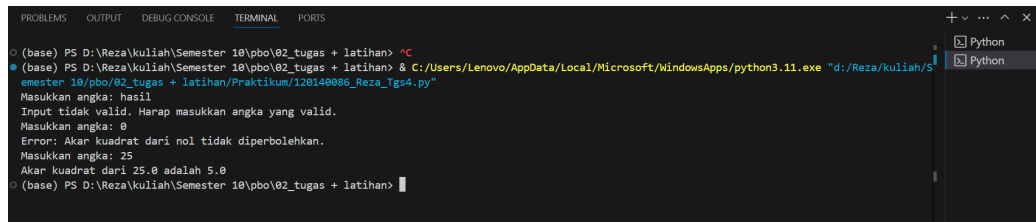
        return math.sqrt(number)

    def run(self):
        number = self.get_input()
        result = self.calculate_square_root(number)
        print(f"Akar kuadrat dari {number} adalah {result}")
    )

if __name__ == "__main__":
    calculator = SquareRootCalculator()
    calculator.run()

```

1.3 Output Hasil



```

(base) PS D:\Reza\kuliah\Semester 10\pbo\02_tugas + latihan> ^C
(base) PS D:\Reza\kuliah\Semester 10\pbo\02_tugas + latihan> & C:/Users/Lenovo/AppData/Local/Microsoft/WindowsApps/python3.11.exe "d:/Reza/kuliah/Semester 10/pbo/02_tugas + latihan/Praktikum/120140086_Reza_Tgs4.py"
Masukkan angka: hasil
Input tidak valid: Harap masukkan angka yang valid.
Masukkan angka: 0
Error: Akar kuadrat dari nol tidak diperbolehkan.
Masukkan angka: 25
Akar kuadrat dari 25.0 adalah 5.0
(base) PS D:\Reza\kuliah\Semester 10\pbo\02_tugas + latihan>

```

1.4 Penjelasan

Program ini merupakan implementasi sederhana dari perhitungan akar kuadrat menggunakan konsep Pemrograman Berorientasi Objek (OOP) dalam Python. Program ini terdiri dari kelas `SquareRootCalculator` yang menangani input pengguna, validasi angka, serta perhitungan akar kuadrat menggunakan metode `math.sqrt()`. Program ini juga menerapkan exception handling untuk mencegah kesalahan input, seperti memasukkan teks atau angka negatif, sehingga lebih aman dan user-friendly. Jika pengguna memasukkan nilai yang tidak valid, program akan memberikan pesan error yang sesuai dan meminta input ulang hingga mendapatkan angka positif yang valid untuk perhitungan akar kuadrat.

2 Soal Nomor 2 — Manajemen Daftar Tugas (To-Do List)

2.1 Input Soal

Program ini meminta pengguna untuk menambahkan, menghapus, dan menampilkan daftar tugas. Program ini menangani beberapa exception yang mungkin terjadi, seperti input yang tidak valid, mencoba menghapus tugas yang tidak ada, dan input kosong. Terapkan exception dan error handling, serta penerapan raising exception dalam menangani error yang terjadi di dalam program.

2.2 Source Code

```
# Tugas 2:
class ToDoList:
    def __init__(self):
        self.tasks = []

    def tambah_tugas(self):
        tugas = input("Masukkan tugas: ").strip()
        if tugas:
            self.tasks.append(tugas)
            print("Tugas berhasil ditambahkan!\n")
        else:
            print("Error: Tugas tidak boleh kosong.\n")

    def hapus_tugas(self):
        if not self.tasks:
            print("Daftar tugas kosong.\n")
            return
        self.tampilkan_tugas()
        try:
            index = int(input("Masukkan nomor tugas yang ingin dihapus: ")) - 1
            if 0 <= index < len(self.tasks):
                self.tasks.pop(index)
                print("Tugas berhasil dihapus!\n")
            else:
                print("Error: Nomor tugas tidak valid.\n")
        except ValueError:
            print("Error: Masukkan angka yang valid.\n")
```

```

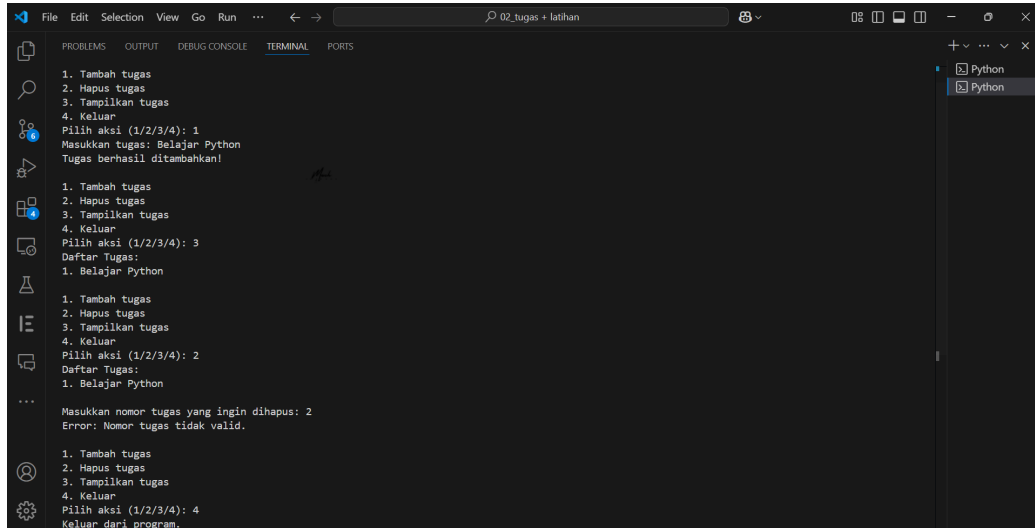
def tampilkan_tugas(self):
    if not self.tasks:
        print("Daftar tugas kosong.\n")
    else:
        print("Daftar Tugas:")
        for i, tugas in enumerate(self.tasks, 1):
            print(f"{i}. {tugas}")
        print()

def jalankan(self):
    while True:
        print("1. Tambah tugas\n2. Hapus tugas\n3.
              Tampilkan tugas\n4. Keluar")
        pilihan = input("Pilih aksi (1/2/3/4): ")
        if pilihan == "1":
            self.tambah_tugas()
        elif pilihan == "2":
            self.hapus_tugas()
        elif pilihan == "3":
            self.tampilkan_tugas()
        elif pilihan == "4":
            print("Keluar dari program.")
            break
        else:
            print("Error: Pilihan tidak valid.\n")

if __name__ == "__main__":
    to_do = ToDoList()
    to_do.jalankan()

```

2.3 Output Hasil



```
File Edit Selection View Go Run ... ← → 02_tugas + latihan
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

1. Tambah tugas
2. Hapus tugas
3. Tampilkan tugas
4. Keluar
Pilih aksi (1/2/3/4): 1
Masukkan tugas: Belajar Python
Tugas berhasil ditambahkan!

1. Tambah tugas
2. Hapus tugas
3. Tampilkan tugas
4. Keluar
Pilih aksi (1/2/3/4): 3
Daftar Tugas:
1. Belajar Python

1. Tambah tugas
2. Hapus tugas
3. Tampilkan tugas
4. Keluar
Pilih aksi (1/2/3/4): 2
Daftar Tugas:
1. Belajar Python

Masukkan nomor tugas yang ingin dihapus: 2
Error: Nomor tugas tidak valid.

1. Tambah tugas
2. Hapus tugas
3. Tampilkan tugas
4. Keluar
Pilih aksi (1/2/3/4): 4
Keluar dari program.
```

2.4 Penjelasan

Program ini merupakan implementasi sederhana dari manajemen daftar tugas (To-Do List) menggunakan konsep Pemrograman Berorientasi Objek (OOP) dalam bahasa Python. Program ini terdiri dari sebuah kelas `ToDoList` yang memiliki beberapa metode untuk menambahkan, menghapus, dan menampilkan daftar tugas. Pengguna dapat memilih aksi melalui menu interaktif, dan program menangani berbagai kemungkinan kesalahan, seperti input kosong atau indeks tugas yang tidak valid, menggunakan exception handling. Dengan pendekatan OOP, program ini lebih terstruktur dan mudah dikembangkan.

3 Soal Nomor 3 — Sistem Manajemen Hewan (Zoo Management System)

3.1 Input Soal

Pada kasus ini, kalian akan membuat sebuah sistem untuk mengelola berbagai jenis hewan di kebun binatang. Setiap hewan memiliki karakteristik dan perilaku yang berbeda, dan sistem ini akan mengimplementasikan konsep-konsep OOP yang telah dipelajari sebelumnya beserta error handling yang kita pelajari hari ini.

3.2 Source Code

```
# Tugas 3
from abc import ABC, abstractmethod

class Animal(ABC):
    def __init__(self, name, age):
        if not name or not isinstance(name, str):
            raise ValueError("Nama hewan harus berupa string yang valid.")
        if not isinstance(age, int) or age <= 0:
            raise ValueError("Usia harus berupa angka positif.")

        self.__name = name
        self.__age = age

    @abstractmethod
    def make_sound(self):
        pass

    def get_name(self):
        return self.__name

    def get_age(self):
        return self.__age

    def set_name(self, name):
        if not name or not isinstance(name, str):
            raise ValueError("Nama hewan harus berupa string yang valid.")
        self.__name = name
```

```

def set_age(self, age):
    if not isinstance(age, int) or age <= 0:
        raise ValueError("Usia harus berupa angka
            positif.")
    self.__age = age

class Lion(Animal):
    def make_sound(self):
        return "Roar!"

class Elephant(Animal):
    def make_sound(self):
        return "Trumpet!"

class Monkey(Animal):
    def make_sound(self):
        return "Ooh ooh aah aah!"

def main():
    zoo = []
    while True:
        print("\n1. Tambah Hewan\n2. Tampilkan Hewan\n3.
            Keluar")
        choice = input("Pilih aksi (1/2/3): ")

        if choice == "1":
            print("Pilih jenis hewan: 1. Singa, 2. Gajah,
                3. Monyet")
            animal_type = input("Masukkan pilihan (1/2/3):
                ")
            name = input("Masukkan nama hewan: ")
            try:
                age = int(input("Masukkan usia hewan: "))
                if animal_type == "1":
                    zoo.append(Lion(name, age))
                elif animal_type == "2":
                    zoo.append(Elephant(name, age))
                elif animal_type == "3":
                    zoo.append(Monkey(name, age))
                else:
                    print("Pilihan tidak valid.")
            except ValueError as e:

```



```

        print(f"Error: {e}")

    elif choice == "2":
        if not zoo:
            print("Belum ada hewan di kebun binatang.")
        else:
            print("Daftar Hewan di Kebun Binatang:")
            for animal in zoo:
                print(f"{animal.get_name()} ({animal.get_age()} tahun) - Suara: {animal.make_sound()}")

    elif choice == "3":
        print("Keluar dari program.")
        break
    else:
        print("Pilihan tidak valid.")

if __name__ == "__main__":
    main()

```

3.3 Output Hasil

```

120140086_Reza_Tgs4.py M
Praktikum > 120140086_Reza_Tgs4.py > ...
142 def main():
172
173     elif choice == "3":
...
(base) PS D:\Reza\kuliah\Semester 18\pbo\02_tugas + latihan> & C:/Users/Lenovo/AppData/Local/Microsoft/WindowsApps/python3.11.exe "d:/Reza/kuliah/Semester 18/pbo/02_tugas + latihan/Praktikum/120140086_Reza_Tgs4.py"
1. Tambah Hewan
2. Tampilkan Hewan
3. Keluar
Pilih aksi (1/2/3): 1
Pilih jenis hewan: 1. Singa, 2. Gajah, 3. Monyet
Masukkan pilihan (1/2/3): 1
Masukkan nama hewan: Simba
Masukkan usia hewan: 12

1. Tambah Hewan
2. Tampilkan Hewan
3. Keluar
Pilih aksi (1/2/3): 2
Daftar Hewan di Kebun Binatang:
Simba (12 tahun) - Suara: Roar!

1. Tambah Hewan
2. Tampilkan Hewan
3. Keluar
Pilih aksi (1/2/3):

```

3.4 Penjelasan

Program ini merupakan sistem manajemen hewan di kebun binatang yang menerapkan konsep Pemrograman Berorientasi Objek (OOP) dalam Python. Kelas abstrak `Animal` digunakan sebagai blueprint untuk semua hewan, dengan metode

`make_sound()` yang wajib diimplementasikan oleh setiap subclass seperti `Lion`, `Elephant`, dan `Monkey`. Program juga menerapkan enkapsulasi dengan atribut `private` yang hanya bisa diakses melalui metode `getter` dan `setter`. Selain itu, program menangani error pada input pengguna, seperti nama kosong atau usia yang tidak valid, dengan `exception handling`. Dengan pendekatan ini, sistem menjadi lebih modular, fleksibel, dan aman dari kesalahan input.

4 Lampiran

Pada bagian ini disertakan tautan yang berkaitan dengan tugas praktikum. Tautan GitHub berisi kode sumber yang telah dibuat selama praktikum, sedangkan tautan Overleaf merupakan tempat penyusunan laporan ini.

Repositori GitHub Tugas: Berikut adalah tautan menuju repositori GitHub yang berisi seluruh kode sumber tugas praktikum:

<https://github.com/rezachairul/Latihan-PBO-2025/tree/main/Praktikum>

Dokumen Laporan di Overleaf: Untuk melihat laporan ini secara langsung di Overleaf, gunakan tautan berikut:

<https://www.overleaf.com/read/bxrvfsgybfry#3c9404>