LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK (PBO) – [TUGAS 4]

Disusun Oleh

Gilang Surya Agung 123140187

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI SUMATERA

2025

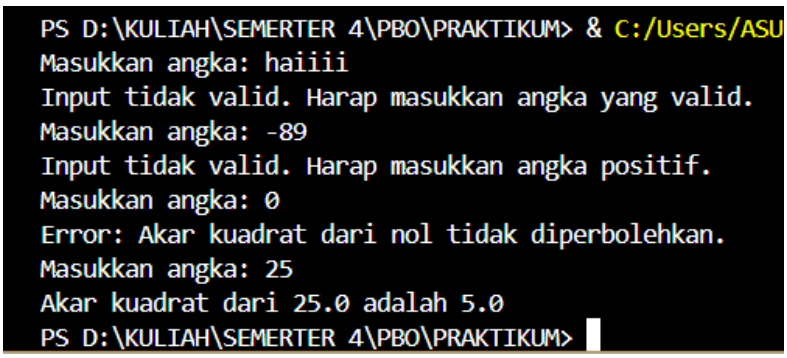
1. **Menghitung akar kuadrat**

Source code :

```python
import math

class ZeroError(Exception):
    pass

def hitung_akar_kuadrat():
    while True:
        try:
            angka = input("Masukkan angka: ")
            angka = float(angka)

            if angka < 0:
                print("Input tidak valid. Harap masukkan angka positif.")
            elif angka == 0:
                raise ZeroError("Error: Akar kuadrat dari nol tidak diperbolehkan.")
            else:
                hasil = math.sqrt(angka)
                print(f"Akar kuadrat dari {angka} adalah {hasil}")
                break

        except ValueError:
            print("Input tidak valid. Harap masukkan angka yang valid.")
        except ZeroError as e:
            print(e)

hitung_akar_kuadrat()
```

Program ini menghitung akar kuadrat angka yang dimasukkan pengguna menggunakan Python. Program mengimpor math untuk fungsi sqrt(), mendefinisikan ZeroError sebagai pengecualian kustom untuk angka nol, dan menggunakan fungsi hitung_akar_kuadrat() dengan perulangan while True. Di dalam try, input dikonversi ke float, dicek: jika negatif, beri pesan; jika nol, picu ZeroError; jika positif, hitung akar kuadrat dan cetak hasil, lalu break. except ValueError tangani input bukan angka, dan except ZeroError tangani nol, keduanya beri pesan error. Program jalan terus sampai input valid, ramah pengguna, dan tangguh.

Output :



```
PS D:\KULIAH\SEMERTER 4\PBO\PRAKTIKUM> & C:/Users/ASU
Masukkan angka: haiiii
Input tidak valid. Harap masukkan angka yang valid.
Masukkan angka: -89
Input tidak valid. Harap masukkan angka positif.
Masukkan angka: 0
Error: Akar kuadrat dari nol tidak diperbolehkan.
Masukkan angka: 25
Akar kuadrat dari 25.0 adalah 5.0
PS D:\KULIAH\SEMERTER 4\PBO\PRAKTIKUM>
```

2. Manajemen Daftar Tugas (To-Do List)

Source code :

```python
class EmptyInputError(Exception):
    pass

class TaskNotFoundError(Exception):
    pass

def tampilkan_menu():
    print("\nPilih aksi:")
    print("1. Tambah tugas")
    print("2. Hapus tugas")
    print("3. Tampilkan daftar tugas")
    print("4. Keluar")

def manajemen_tugas():
    daftar_tugas = []

    while True:
        tampilkan_menu()
        try:
            pilihan = input("Masukkan pilihan (1/2/3/4): ")
            pilihan = int(pilihan)  # Konversi ke integer

            if pilihan not in [1, 2, 3, 4]:
                raise ValueError("Pilihan tidak valid. Masukkan angka antara 1 dan 4.")

            if pilihan == 1:
                tugas = input("Masukkan tugas yang ingin ditambahkan: ").strip()
                if not tugas:
                    raise EmptyInputError("Error: Input tidak boleh kosong.")
                daftar_tugas.append(tugas)
                print("Tugas berhasil ditambahkan!")

            elif pilihan == 2:
                if not daftar_tugas:
                    raise TaskNotFoundError("Error: Daftar tugas kosong, tidak ada yang bisa dihapus.")

                nomor = input("Masukkan nomor tugas yang ingin dihapus: ")
                nomor = int(nomor) - 1
                if nomor < 0 or nomor >= len(daftar_tugas):
                    raise TaskNotFoundError(f"Error: Tugas dengan nomor {nomor + 1} tidak ditemukan.")

                del daftar_tugas[nomor]
                print("Tugas berhasil dihapus!")

            elif pilihan == 3:
                if not daftar_tugas:
                    print("Daftar tugas kosong.")
                else:
                    print("\nDaftar Tugas:")
                    for i, tugas in enumerate(daftar_tugas, 1):
                        print(f"{i}. {tugas}")

            elif pilihan == 4:
                print("Keluar dari program.")
                break

        except ValueError as e:
            if "invalid literal" in str(e):
                print("Error: Masukkan angka yang valid.")
            else:
                print(e)
        except EmptyInputError as e:
            print(e)
        except TaskNotFoundError as e:
            print(e)

manajemen_tugas()
```

Program ini adalah sistem manajemen daftar tugas sederhana dalam Python yang memungkinkan pengguna menambah, menghapus, dan menampilkan tugas. Dua kelas pengecualian kustom, EmptyInputError dan TaskNotFoundError, dibuat untuk menangani input kosong dan tugas yang tidak ditemukan. Fungsi tampilkan_menu() menampilkan opsi (1-4), sedangkan manajemen_tugas() mengelola logika utama dengan daftar tugas kosong dan perulangan while True. Dalam blok try, input pilihan dikonversi ke integer dan divalidasi; jika tidak valid, ValueError dipicu. Pilihan 1 menambah tugas (jika kosong, picu EmptyInputError), pilihan 2 menghapus tugas berdasarkan nomor (jika daftar kosong atau nomor salah, picu TaskNotFoundError), pilihan 3 menampilkan daftar, dan pilihan 4 keluar. Bagian except menangani kesalahan: ValueError untuk input bukan angka atau pilihan salah, EmptyInputError untuk tugas kosong, dan TaskNotFoundError untuk tugas yang tidak ada, semua dengan pesan error yang jelas. Program berjalan terus hingga pengguna keluar, tangguh, dan ramah pengguna.

Output :

```
PS D:\KULIAH\SEMERTER 4\PBO\PRAKTIKUM> & C:/User

Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 1
Masukkan tugas yang ingin ditambahkan: pbo
Tugas berhasil ditambahkan!

Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 1
Masukkan tugas yang ingin ditambahkan: stigma
Tugas berhasil ditambahkan!

Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 3
```

```
Daftar Tugas:
1. pbo
2. stigma

Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 4
Keluar dari program.
PS D:\KULIAH\SEMERTER 4\PBO\PRAKTIKUM>
```

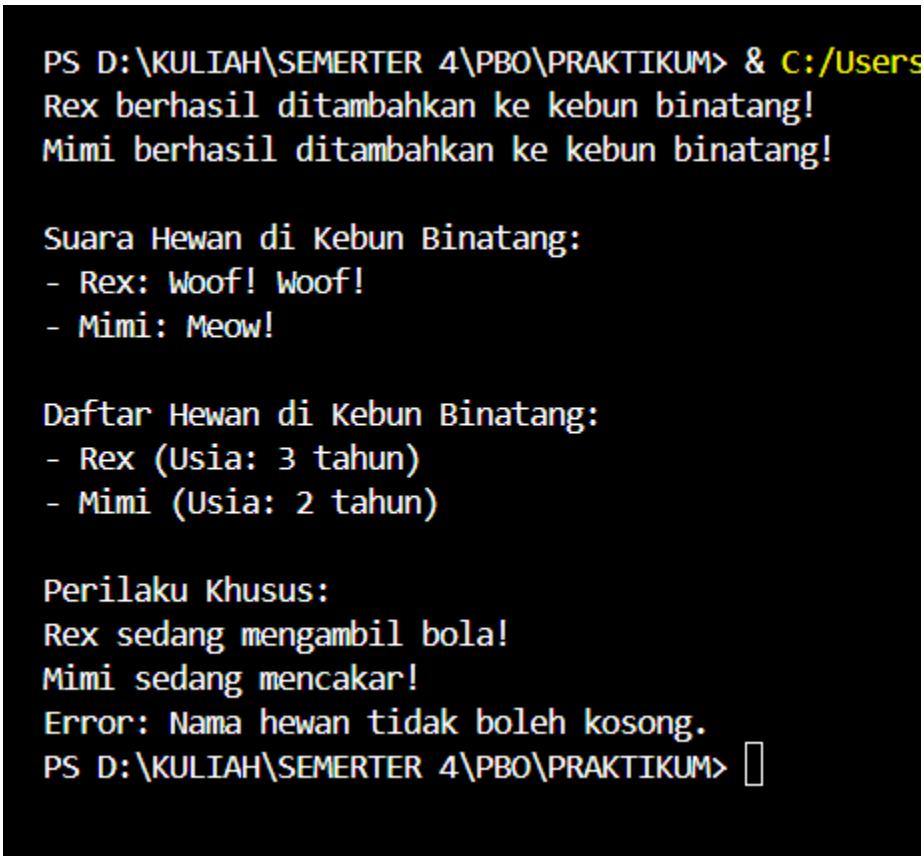
3. Sistem Manajemen Hewan (Zoo Management System)

Source code :

```python
from abc import ABC, abstractmethod  # Untuk abstraksi

# Custom exception untuk error tertentu
class InvalidAgeError(Exception):
    pass

class EmptyNameError(Exception):
    pass

# Kelas abstrak Animal sebagai dasar semua hewan
class Animal(ABC):
    def __init__(self, name, age):
        if not name.strip():  # Cek jika nama kosong
            raise EmptyNameError("Error: Nama hewan tidak boleh kosong.")
        if not isinstance(age, (int, float)) or age < 0:  # Cek usia valid
            raise InvalidAgeError("Error: Usia harus angka positif.")

        # Enkapsulasi: atribut privat dengan underscore
        self._name = name
        self._age = age

    # Metode getter untuk akses data
    def get_name(self):
        return self._name

    def get_age(self):
        return self._age

    # Metode abstrak yang harus diimplementasikan oleh kelas turunan
    @abstractmethod
    def make_sound(self):
        pass

    # Metode tambahan (opsional) untuk contoh perilaku lain
    def info(self):
        return f"{self._name} (Usia: {self._age} tahun)"

# Kelas turunan untuk Anjing
class Dog(Animal):
    def make_sound(self):
        return "Woof! Woof!"  # Polimorfisme: suara anjing

    def fetch(self):  # Perilaku khusus anjing
        return f"{self._name} sedang mengambil bola!"

# Kelas turunan untuk Kucing
class Cat(Animal):
    def make_sound(self):
        return "Meow!"  # Polimorfisme: suara kucing

    def scratch(self):  # Perilaku khusus kucing
        return f"{self._name} sedang mencakar!"

# Kelas Zoo untuk mengelola hewan
class Zoo:
    def __init__(self):
        self.animals = []  # Daftar hewan

    def add_animal(self, animal):
        try:
            self.animals.append(animal)
            print(f"{animal.get_name()} berhasil ditambahkan ke kebun binatang!")
        except Exception as e:
            print(f"Gagal menambahkan hewan: {e}")

    def show_all_sounds(self):
        if not self.animals:
            print("Belum ada hewan di kebun binatang.")
        else:
            print("\nSuara Hewan di Kebun Binatang:")
            for animal in self.animals:
                print(f"- {animal.get_name()}: {animal.make_sound()}")

    def show_all_info(self):
        if not self.animals:
            print("Belum ada hewan di kebun binatang.")
        else:
            print("\nDaftar Hewan di Kebun Binatang:")
            for animal in self.animals:
                print(f"- {animal.info()}")

# Program utama
def main():
    zoo = Zoo()

    # Contoh penggunaan dengan error handling
    try:
        # Tambah hewan valid
        dog = Dog("Rex", 3)
        cat = Cat("Mimi", 2)

        zoo.add_animal(dog)
        zoo.add_animal(cat)

        # Tampilkan suara dan info
        zoo.show_all_sounds()
        zoo.show_all_info()

        # Contoh perilaku khusus
        print(f"\nPerilaku Khusus:")
        print(dog.fetch())
        print(cat.scratch())

        # Contoh error: nama kosong
        invalid_animal = Dog("", 5)
        zoo.add_animal(invalid_animal)  # Ini tidak akan tercapai karena error

    except EmptyNameError as e:
        print(e)
    except InvalidAgeError as e:
        print(e)
    except Exception as e:
        print(f"Terjadi kesalahan: {e}")

# Jalankan program
if __name__ == "__main__":
    main()
```

Program ini adalah Sistem Manajemen Hewan berbasis OOP di Python untuk mengelola hewan di kebun binatang. Modul abc diimpor untuk abstraksi, dan dua pengecualian kustom (InvalidAgeError, EmptyNameError) dibuat untuk menangani usia tidak valid dan nama kosong. Kelas abstrak Animal mendefinisikan metode wajib make_sound() serta atribut privat _name dan _age (enkapsulasi), dengan getter untuk akses. Kelas Dog dan Cat mewarisi Animal, mengimplementasikan make_sound() berbeda (polimorfisme) dan menambah perilaku khusus (fetch, scratch). Kelas Zoo mengelola daftar hewan, dengan metode untuk menambah dan menampilkan suara/info hewan. Fungsi main() menjalankan program, membuat objek Dog dan Cat, menambahkannya ke Zoo, dan menangani error (misalnya nama kosong atau usia negatif) menggunakan try-except. Program ini rapi, fleksibel, dan tangguh berkat OOP dan exception handling.

Output :



Lampiran :

https://chatgpt.com/share/67e2ab9c-02a4-8012-9af4-836b3d1c6bd8