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

Disusun Oleh :

Annisa Salsabila123140070

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI SUMATERA

2025

Soal nomor 1 Menghitung Akar Kuadrat

1. Menghitung Akar Kuadrat

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.

contoh output:

```
Masukkan angka: Hello
Input tidak valid. Harap masukkan angka yang valid.
Masukkan angka: -5
Input tidak valid. Harap masukkan angka positif.
Masukkan angka: 0
Error: Akar kuadrat dari nol tidak diperbolehkan.
Masukkan angka: 25
Akar kuadrat dari 25 adalah 5.0
```

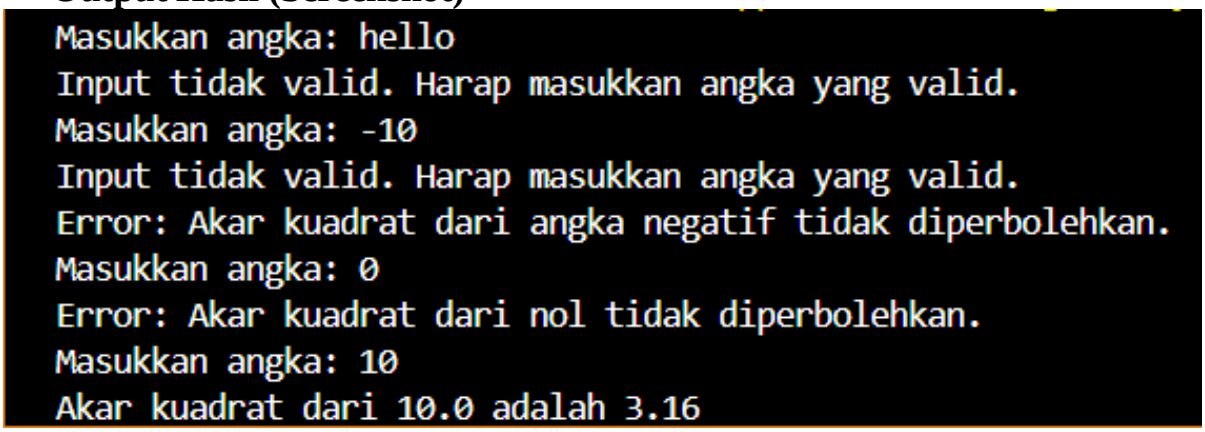
Penjelasan

Soal ini memerintahkan untuk menulis program yang menghitung akar kuadrat dari bilangan yang diinput pengguna. Program harus bisa menangani berbagai jenis input, seperti input tidak valid (tidak merupakan bilangan), bilangan negatif, dan bilangan nol. Error handling dan exception handling harus diterapkan untuk memberikan pernyataan error yang tepat kepada pengguna pada setiap jenis kesalahan input. Jika input adalah valid (positif number), program akan menghitung dan menampilkan akar kuadratnya dari number tersebut. Contoh output dibuat untuk membantu menunjukkan perilaku program yang diharapkan.

Source Code

```
import math

def
hitung_akar_kuadrat():
    while True:
        try:


 angka_input =
input("Masukkan angka:
")


            angka =
float(angka_input)

            if angka <
0:

 print("Input tidak
valid. Harap masukkan
angka yang valid.")

 print("Error: Akar
kuadrat dari angka
negatif tidak
diperbolehkan.")
            elif angka
== 0:

 print("Error: Akar
kuadrat dari nol tidak
diperbolehkan.")
            else:
                akar =
math.sqrt(angka)

 print(f"Akar kuadrat
dari {angka} adalah
{akar:.2f}")
                break

        except
ValueError:

 print("Input tidak
valid. Harap masukkan
angka yang valid.")
hitung_akar_kuadrat()
```

Output Hasil (Screenshot)



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

2. Manajemen Daftar Tugas (To-Do List)

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.

contoh output:

```
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: Belajar Python
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:
- Belajar Python
```

Penjelasan

Membuat sebuah aplikasi berbasis teks terkait dengan pengelolaan daftar tugas tersebut. Pengguna dapat membuat tugas baru, menghapus tugas lama, atau melihat daftar tugas yang telah ada.

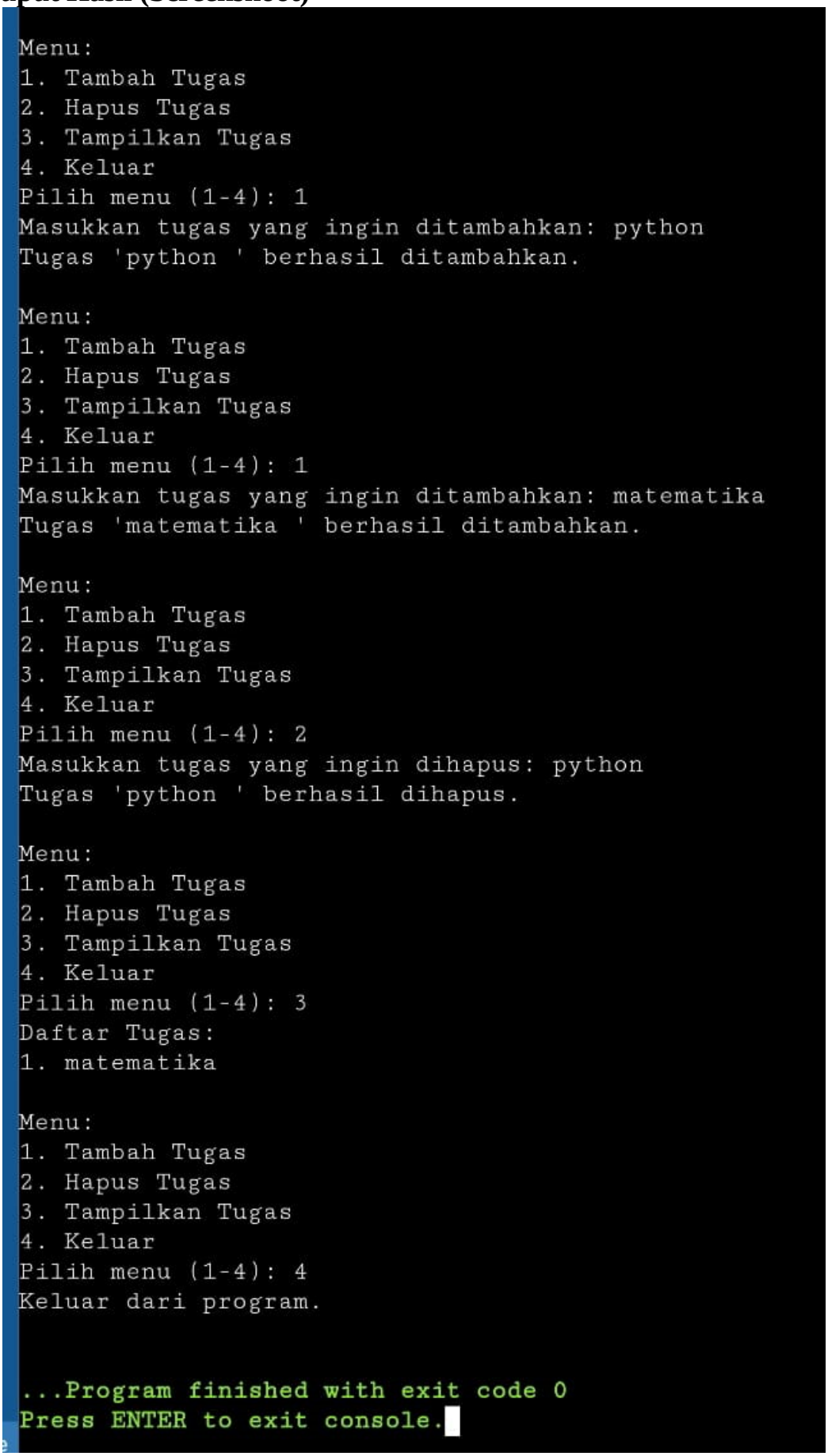
Mengedepankan penanganan kesalahan. Dengan demikian, pengguna input data yang tidak valid ataupun keadaan-keadaan lain yang biasanya menyebabkan crash pada program tidak menjadi masalah. Harus ada penerapan konsep exception maupun raising exception agar program tetap dapat berjalan dan menampilkan bukan pesan aneh.

Source Code

```
class TugasError(Exception):
    """Kelas exception khusus
untuk kesalahan tugas."""
    pass

class DaftarTugas:
    def __init__(self):
        self.tugas = []

    def tambah_tugas(self, tugas):

        if not tugas:
            raise
TugasError("Tugas tidak boleh
kosong.")
        self.tugas.append(tugas)
        print(f"Tugas '{tugas}'
berhasil ditambahkan.")

    def hapus_tugas(self, tugas):
        if tugas not in
self.tugas:
            raise
TugasError(f"Tugas '{tugas}' tidak
ditemukan dalam daftar.")
        self.tugas.remove(tugas)
        print(f"Tugas '{tugas}'
berhasil dihapus.")

    def tampilkan_tugas(self):
        if not self.tugas:
            print("Daftar tugas
kosong.")
        else:
            print("Daftar Tugas:")

            for i, tugas in
enumerate(self.tugas, start=1):
                print(f"{i}.
{tugas}")
```

```
def main():
    daftar_tugas = DaftarTugas()

    while True:
        print("\nMenu:")
        print("1. Tambah Tugas")
        print("2. Hapus Tugas")
        print("3. Tampilkan
Tugas")
        print("4. Keluar")

        try:
            pilihan =
int(input("Pilih menu (1-4): "))
            if pilihan == 1:
                tugas =
input("Masukkan tugas yang ingin
ditambahkan: ")

 daftar_tugas.tambah_tugas(tugas)
            elif pilihan == 2:
                tugas =
input("Masukkan tugas yang ingin
dihapus: ")

 daftar_tugas.hapus_tugas(tugas)
            elif pilihan == 3:

 daftar_tugas.tampilkan_tugas()
            elif pilihan == 4:
                print("Keluar dari
program.")
                break
            else:
                print("Pilihan
tidak valid. Silakan pilih antara
1-4.")
        except ValueError:
            print("Input tidak
valid. Harap masukkan angka.")
        except TugasError as e:
            print(e)

if __name__ == "__main__":
    main()
```

Ouput Hasil (Screenshoot)



Soal nomor 3 Sistem Manajemen Hewan (Zoo Management System)

3. Sistem Manajemen Hewan (Zoo Management System)

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 eror handling yang kita pelajari hari ini.

Deskripsi Program:

1. **Polimorfisme:** Setiap hewan dapat memiliki metode `make_sound()` yang berbeda-beda sesuai dengan jenis hewan tersebut. Metode ini akan dipanggil melalui objek hewan yang berbeda jenis (misalnya, anjing, kucing, dll), kalo mau tambah metode lain boleh.
2. **Abstraksi:** Kalian akan membuat kelas abstrak `Animal` yang mendefinisikan metode `make_sound()` yang harus diimplementasikan oleh semua kelas turunan.
3. **Enkapsulasi:** Data terkait hewan akan disembunyikan di dalam kelas dan hanya bisa diakses atau dimodifikasi melalui metode tertentu.
4. **Inheritance:** Semua hewan akan mewarisi kelas dasar `Animal` dan dapat menambahkan perilaku atau atribut mereka sendiri.
5. **Exception Handling:** Program akan menangani error jika ada input yang tidak valid, misalnya, saat menambahkan hewan tanpa nama atau usia yang benar.

Penjelasan

Soal ini menjelaskan tentang proyek pembuatan sebuah "Sistem Manajemen Hewan (Zoo Management System)". Dalam soal ini, akan membuat sistem untuk mengelola berbagai jenis hewan di kebun binatang. Sistem ini akan mengimplementasikan konsep-konsep Pemrograman Berorientasi Objek (PBO) yang telah dipelajari sebelumnya, termasuk penanganan kesalahan (error handling) yang dipelajari hari ini.

Deskripsi programnya meliputi: Polimorfisme, di mana setiap hewan akan memiliki metode make_sound() yang berbeda sesuai jenisnya dan dapat dipanggil melalui objek hewan yang berbeda (misalnya, anjing, kucing, dll.), dan juga diperbolehkan menambahkan metode lain. Abstraksi akan diimplementasikan dengan membuat kelas abstrak Animal yang mendefinisikan metode make_sound() yang wajib diimplementasikan oleh semua kelas turunannya. Enkapsulasi akan memastikan bahwa data terkait

hewan disembunyikan di dalam kelas dan hanya dapat diakses atau dimodifikasi melalui metode tertentu. Inheritance (pewarisan) akan memungkinkan semua hewan mewarisi kelas dasar Animal dan menambahkan perilaku atau atribut spesifik mereka. Terakhir, Exception Handling (penanganan pengecualian) akan diterapkan untuk mengatasi kesalahan jika ada input yang tidak valid, contohnya saat menambahkan hewan tanpa nama atau usia yang benar.

Source Code

```python
from abc import ABC,
abstractmethod

# Kelas Abstrak
class Animal(ABC):
    def __init__(self, nama,
usia):
        self.__nama = nama
        self.__usia = usia

    @abstractmethod
    def make_sound(self):
        pass

    def get_nama(self):
        return self.__nama

    def get_usia(self):
        return self.__usia

    def set_nama(self, nama):
        self.__nama = nama

    def set_usia(self, usia):
        if usia < 0:
            raise
ValueError("Usia tidak boleh
negatif.")
        self.__usia = usia

# Kelas Turunan
class Anjing(Animal):
    def make_sound(self):
        return "Guk! Guk!"

class Kucing(Animal):
    def make_sound(self):
        return "Meow!"

class Burung(Animal):
    def make_sound(self):
        return "Cuit! Cuit!"
```

```
# Sistem Manajemen Hewan
class ZooManagementSystem:
    def __init__(self):
        self.__hewan_list = []

    def tambah_hewan(self,
hewan):
        if not isinstance(hewan,
Animal):
            raise
TypeError("Objek yang
ditambahkan harus merupakan
instansi dari kelas Animal.")

 self.__hewan_list.append(hewan
)
    def tampilkan_hewan(self):
        for hewan in
self.__hewan_list:
            print(f"Nama:
{hewan.get_nama()}, Usia:
{hewan.get_usia()}, Suara:
{hewan.make_sound()}")

# Fungsi utama
def main():
    zoo = ZooManagementSystem()

    while True:
        try:
            print("\n=== Sistem
Manajemen Kebun Binatang ===")
            print("1. Tambah
Hewan")
            print("2. Tampilkan
Hewan")
            print("3. Keluar")
            pilihan =
int(input("Pilih menu: "))

            if pilihan == 1:
                nama =
input("Masukkan nama hewan: ")
                usia =
int(input("Masukkan usia hewan:
"))
                jenis =
input("Masukkan jenis hewan
(anjing/kucing/burung):
").lower()
```

```
                if jenis ==
"anjing":
                    hewan =
Anjing(nama, usia)
                elif jenis ==
"kucing":
                    hewan =
Kucing(nama, usia)
                elif jenis ==
"burung":
                    hewan =
Burung(nama, usia)
                else:
                    print("Jenis
hewan tidak valid.")
                    continue


 zoo.tambah_hewan(hewan)
                print(f"Hewan
{nama} berhasil ditambahkan.")

            elif pilihan == 2:

 zoo.tampilkan_hewan()

            elif pilihan == 3:
                print("Keluar
dari sistem.")
                break

            else:
                print("Pilihan
tidak valid. Silakan coba
lagi.")

        except ValueError as e:
            print(f"Kesalahan:
{e}")
        except Exception as e:
            print(f"Terjadi
kesalahan: {e}")

if __name__ == "__main__":
    main()
```

Output Hasil (Screenshoot)

Lampiran

1. **Link Percakapan LLM**
2. **Web Referensi - DuniaIlkom**
3. **https://www.blackbox.ai/chat/K66nW50**
4. **https://www.blackbox.ai/chat/UxvjziA**
5. **https://www.blackbox.ai/chat/Oh0zWDD**