

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK (PBO) – [TUGAS 3]**



Disusun Oleh

Nadine Aura Rahmadhani 123140195

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI SUMATERA

2025

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."

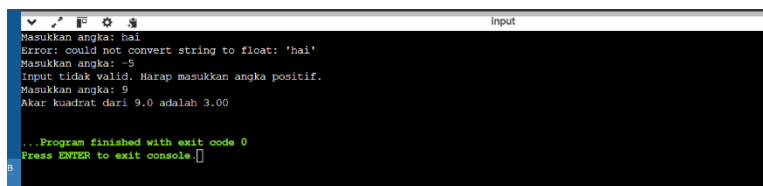
Jika angka yang dimasukkan negatif atau nol, program akan menampilkan pesan error yang relevan, seperti "Akar kuadrat dari nol tidak diperbolehkan."

Jika inputnya valid (angka positif), program akan menghitung dan menampilkan akar kuadratnya. Berikan Penjelasan

Contoh: Pada tugas praktikum ini, diminta untuk membuat sebuah pola segitiga menggunakan karakter (*) dengan ukuran segitiganya bergantung kepada nilai inputan Height. Jelaskan juga kode pemrogramannya di sini

```
1 import math
2
3 def hitung_akar():
4     while True:
5         try:
6             angka = float(input("Masukkan angka: "))
7             if angka < 0:
8                 print("Input tidak valid. Harap masukkan angka positif.")
9             elif angka == 0:
10                raise ValueError("Akar kuadrat dari nol tidak diperbolehkan.")
11            else:
12                print(f"Akar kuadrat dari {angka} adalah {math.sqrt(angka):.2f}")
13                break
14        except ValueError as e:
15            print(f"Error: {e}")
16        except Exception:
17            print("Input tidak valid. Harap masukkan angka yang valid.")
18
19 hitung_akar()
```

Output Hasil (Screenshot)



```
Input
Masukkan angka: hai
Error: could not convert string to float: 'hai'
Masukkan angka: -5
Input tidak valid. Harap masukkan angka positif.
Masukkan angka: 9
Akar kuadrat dari 9.0 adalah 3.00
...Program finished with exit code 0
Press ENTER to exit console.
```

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.

```

1 def main():
2     daftar_tugas = []
3
4     while True:
5         print("\nPilih aksi:")
6         print("1. Tambah tugas\n2. Hapus tugas\n3. Tampilkan daftar tugas\n4. Keluar")
7
8         pilihan = input("Masukkan pilihan (1/2/3/4): ").strip()
9
10        if pilihan == "1": # Tambah tugas
11            tugas = input("Masukkan tugas yang ingin ditambahkan: ").strip()
12            if tugas:
13                daftar_tugas.append(tugas)
14                print("Tugas berhasil ditambahkan!")
15            else:
16                print("Error: Tugas tidak boleh kosong.")
17
18        elif pilihan == "2": # Hapus tugas
19            if not daftar_tugas:
20                print("Daftar tugas kosong.")
21                continue
22
23            try:
24                no_tugas = int(input("Masukkan nomor tugas yang ingin dihapus: ")) - 1
25                if 0 <= no_tugas < len(daftar_tugas):
26                    print(f"Tugas '{daftar_tugas.pop(no_tugas)}' berhasil dihapus.")
27                else:
28                    print("Error: Nomor tugas tidak ditemukan.")
29            except ValueError:
30                print("Error: Masukkan angka yang valid.")
31

```

```

22
23        try:
24            no_tugas = int(input("Masukkan nomor tugas yang ingin dihapus: ")) - 1
25            if 0 <= no_tugas < len(daftar_tugas):
26                print(f"Tugas '{daftar_tugas.pop(no_tugas)}' berhasil dihapus.")
27            else:
28                print("Error: Nomor tugas tidak ditemukan.")
29        except ValueError:
30            print("Error: Masukkan angka yang valid.")
31
32        elif pilihan == "3": # Tampilkan daftar tugas
33            if not daftar_tugas:
34                print("Daftar tugas kosong.")
35            else:
36                print("Daftar Tugas:")
37                for i, tugas in enumerate(daftar_tugas, 1):
38                    print(f"{i}. {tugas}")
39
40        elif pilihan == "4": # Keluar
41            print("Keluar dari program.")
42            break
43
44        else:
45            print("Error: Pilihan tidak valid.")
46
47 if __name__ == "__main__":
48     main()

```

Hasil 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: Membaca Buku
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. Membaca Buku

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: Mencuci Baju
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. Membaca Buku
2. Mencuci Baju

```

```
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 3
Daftar Tugas:
1. Membaca Buku
2. Mencuci Baju

Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 2
Masukkan nomor tugas yang ingin dihapus: 2
Tugas 'Mencuci Baju' berhasil dihapus.

Pilih aksi:
1. Tambah tugas
2. Hapus tugas
3. Tampilkan daftar tugas
4. Keluar
Masukkan pilihan (1/2/3/4): 3
Daftar Tugas:
1. Membaca Buku

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

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.

```
1 from abc import ABC, abstractmethod
2
3 class Animal(ABC):
4     def __init__(self, name, age):
5         if not name.strip():
6             raise ValueError("Nama tidak boleh kosong.")
7         if age <= 0:
8             raise ValueError("Usia harus lebih dari 0.")
9         self.__name = name
10        self.__age = age
11
12    @abstractmethod
13    def make_sound(self):
14        pass
15
16    def get_info(self):
17        return f"{self.__name}, {self.__age} tahun"
18
19 class Dog(Animal):
20     def make_sound(self): return "Guk guk!"
21
22 class Cat(Animal):
23     def make_sound(self): return "Meong!"
24
25 class Bird(Animal):
26     def make_sound(self): return "Cuit cuit!"
27
28 class Zoo:
29     def __init__(self):
30         self.animals = []
31
32     def add_animal(self):
33         try:
34             name = input("Masukkan nama hewan: ").strip()
```

```

33-         try:
34-             name = input("Masukkan nama hewan: ").strip()
35-             age = int(input("Masukkan usia hewan: "))
36-             species = input("Jenis hewan (dog/cat/bird): ").strip().lower()
37-
38-             if species == "dog":
39-                 self.animals.append(Dog(name, age))
40-             elif species == "cat":
41-                 self.animals.append(Cat(name, age))
42-             elif species == "bird":
43-                 self.animals.append(Bird(name, age))
44-             else:
45-                 print("Error: Jenis hewan tidak dikenali.")
46-                 return
47-
48-             print("Hewan berhasil ditambahkan!")
49-         except ValueError as e:
50-             print(f"Error: {e}")
51-
52-     def show_animals(self):
53-         if not self.animals:
54-             print("Tidak ada hewan di kebun binatang.")
55-         else:
56-             for i, animal in enumerate(self.animals, 1):
57-                 print(f"{i}. {animal.get_info()} - Suara: {animal.make_sound()}")
58-
59-     def run(self):
60-         while True:
61-             print("\n1. Tambah Hewan\n2. Tampilkan Hewan\n3. Keluar")
62-             choice = input("Pilih aksi (1/2/3): ").strip()
63-
64-             if choice == "1":
65-                 self.add_animal()
66-             elif choice == "2":
67-                 self.show_animals()
68-             elif choice == "3":
69-                 print("Keluar dari program.")
70-                 break
71-             else:
72-                 print("Error: Pilihan tidak valid.")
73-
74- if __name__ == "__main__":
75-     Zoo().run()

```

```

59-     def run(self):
60-         while True:
61-             print("\n1. Tambah Hewan\n2. Tampilkan Hewan\n3. Keluar")
62-             choice = input("Pilih aksi (1/2/3): ").strip()
63-
64-             if choice == "1":
65-                 self.add_animal()
66-             elif choice == "2":
67-                 self.show_animals()
68-             elif choice == "3":
69-                 print("Keluar dari program.")
70-                 break
71-             else:
72-                 print("Error: Pilihan tidak valid.")
73-
74- if __name__ == "__main__":
75-     Zoo().run()

```

Hasil output

```

1. Tambah Hewan
2. Tampilkan Hewan
3. Keluar
Pilih aksi (1/2/3): 1
Masukkan nama hewan: Anjing
Masukkan usia hewan: 3
Jenis hewan (dog/cat/bird): dog
Hewan berhasil ditambahkan!

1. Tambah Hewan
2. Tampilkan Hewan
3. Keluar
Pilih aksi (1/2/3): 1
Masukkan nama hewan: Kucing
Masukkan usia hewan: 2
Jenis hewan (dog/cat/bird): cat
Hewan berhasil ditambahkan!

1. Tambah Hewan
2. Tampilkan Hewan
3. Keluar
Pilih aksi (1/2/3): 1
Masukkan nama hewan: Burung
Masukkan usia hewan: 1
Jenis hewan (dog/cat/bird): bird
Hewan berhasil ditambahkan!

1. Tambah Hewan
2. Tampilkan Hewan
3. Keluar
Pilih aksi (1/2/3): 2
1. Anjing, 3 tahun - Suara: Guk guk!
2. Kucing, 2 tahun - Suara: Meong!
3. Burung, 1 tahun - Suara: Cuit cuit!

1. Tambah Hewan
2. Tampilkan Hewan
3. Keluar

```