ÖVNINGSUPPGIFTER

OOP - Arv

1. Fordon

Skapa ett Python-program som modellerar en hierarki av fordon med hjälp av arv. Börja med en basklass **Fordon (Vehicle)** och skapa sedan två eller fler underklasser (t.ex. Bil, Cykel, Motorcykel) som ärver från basklassen **Fordon**. Varje klass ska ha specifika attribut och metoder som är relaterade till den typ av fordon den representerar.

1. Definiera basklassen **Fordon (Vehicle)** med gemensamma attribut som märke, modell och år, samt metoder som **start()**, **stop()**, och **tanka (fuel_up())**.

2. Skapa underklasser för olika typer av fordon, t.ex. **Bil (Car)**, **Cykel (Bicycle)** och **Motorcykel (Motorcycle)**. Varje underklass ska ärva från basklassen **Fordon (Vehicle)** och lägga till attribut och metoder specifika för den typen av fordon. Till exempel kan klassen **Bil (Car)** ha attribut som **antal_dörrar (num_doors)**, och klassen **Cykel (Bicycle)** kan ha attribut som **antal_växlar (num_gears)**.

3. Implementera specifika metoder för varje underklass. Till exempel kan klassen **Bil (Car)** ha en metod för att tuta, och klassen **Cykel (Bicycle)** kan ha en metod för att ringa i ringklockan.

4. Skapa instanser av varje fordontyp och demonstrera deras specifika metoder och attribut. Till exempel kan du skapa en bil, cykel och motorcykel och anropa metoder som **start()**, **stop()**, samt deras specifika metoder som **tuta (honk_horn())** eller **ring_i_klockan (ring_bell())**.

Körningsexempel:

```
Toyota Corolla från 2020 har startat.
Toyota Corolla tutar.
Toyota Corolla från 2020 har stannat.

Crescent Sting från 2022 har startat.
Crescent Sting ringer i ringklockan.
Crescent Sting från 2022 har stannat.

Harley-Davidson Street 750 från 2019 har startat.
Harley-Davidson Street 750 med 750cc motor varvar motorn.
Harley-Davidson Street 750 från 2019 har stannat.
```

2. Polymorfism (Utmaning)

Skapa ett Python-program som utforskar polymorfism med hjälp av en hierarki av former. Börja med en basklass **Form (Shape)** och skapa sedan två eller fler underklasser (t.ex. Cirkel, Rektangel, Triangel) som ärver från basklassen **Form**. Varje formklass ska ha sin egen implementation av metoder som **area()** och **perimeter()**. Dessa metoder ska beräkna arean och omkretsen för respektive form.

1. Definiera basklassen **Form (Shape)** med metoder som **area()** och **perimeter()**. Du kan initiera eventuella gemensamma attribut i basklassen.

2. Skapa underklasser för olika former, t.ex. **Cirkel (Circle)**, **Rektangel (Rectangle)** och **Triangel (Triangle)**. Varje underklass ska ärva från basklassen **Form (Shape)** och implementera sin egen version av metoderna **area()** och **perimeter()**.

3. Implementera specifika metoder för varje underklass. Till exempel kan klassen **Cirkel (Circle)** ha en metod för att beräkna sin area baserat på radien, och klassen **Rektangel (Rectangle)** kan ha en metod för att beräkna sin area baserat på längd och bredd.

Skapa instanser av varje formtyp och demonstrera användningen av polymorfism genom att anropa metoderna **area()** och **perimeter()** på dem.