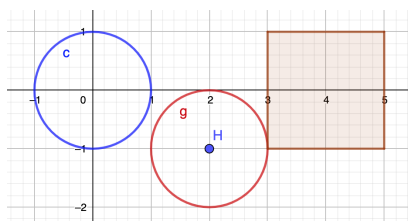


# Labb 3 - Geometri OOP

**Syftet** med den här laborationen är att använda objektorienterad programmering i Python för att designa program med god struktur.



I den här laborationen ska du börja med att planera hur du vill strukturera dina klasser med hjälp av UML och därefter implementera din planering i Python.

## Uppgift

Du ska skapa följande geometriska klasser:

- rektangel
- cirkel

Notera att du även kan behöva ytterligare klass(er) om du vill utnyttja arv och/eller komposition.

Samtliga klasser ska ha minst:

- en areamethod
- en omkretsmethod
- en egen implementation av `==` för att göra jämförelse
- en `__repr__`
- x och y som representerar mittpositionen av objektet
- en translationsmethod som gör det möjligt att förflytta x och y
- en metod som checkar om en viss punkt befinner sig innanför i objektet
- felhantering

Dina klasser ska finnas i en .py-fil. Man ska exempelvis kunna använda dina klasser på följande sätt:

```
from geometry_shapes import Circle

cirkel1 = Circle(x=0,y=0, radius=1) # enhetscirkel
cirkel2 = Circle(x=1,y=1, radius=1)
rektangel = Rectangle(x=0,y=0,side1=1, side2=1)

print(cirkel1==cirkel2) # True

print(cirkel2==rektangel) # False

print(cirkel1.is_inside(0.5, 0.5)) # True

cirkel1.translate(5,5)

print(cirkel1.is_inside(0.5, 0.5)) # False

cirkel1.translate("TRE",5) # ge ValueError med lämplig kommentar
```

Börja med att skapa en planering med [diagrams.net](https://diagrams.net) och därefter implementerar du din planering i Python. Tips är att ha en .ipynb-fil vid sidan om som importerar dina klasser för att göra manuella tester medan du skriver klasserna.

## Bonusuppgifter (frivilliga)

- skapa en kubklass med motsvarande funktionaliteter
- skapa en sfärklass med motsvarande funktionaliteter
- enhetstesta dina klasser med enkla testfall
  - ha en separat .py-fil för enhetstesterna

## Bedömning

Om du har fått någon kodsnuitt från någon annan eller hittat i någon sida är det **viktigt** att du källhänvisar. Skriv en kommentar bredvid koden som du har tagit.

## Godkänt

- gjort en enkel planering med UML
- löst uppgift på korrekt sätt
- koden är kommenterad med relevanta kommentarer, docstrings ska användas
- variabelnamnen är bra valda

- gjort flera relevanta git commits

## **Väl Godkänt**

- koden är effektiv och enkel att följa
- koden är välstrukturerad med lämpliga klasser, funktioner, metoder
- koden återanvänds där det går istället för att återupprepa
- kommentarerna är datavetenskapligt korrekta
- gjort samtliga uppgifter