Funksjoner kan enkelt testes

- En typisk funksjon har som formål å beregne en bestemt returverdi basert på argumentene den mottar
  - For bestemte argumentverdier er det ofte åpenbart hva returverdien burde være - *sortere([4,1,2])* bør returnere *[1,2,4]*
  - Vi kan dermed enkelt lage tester:
    ```
    assert sortere([4,1,2]) == [1,2,4]
    assert sortere([4,1,-2]) == [-2,1,4]
    ```
- `[innenfor.py]`

Testing av funksjoner kan ha mange formål

- Kontrollere at en funksjon virker etter hensikten
  - Systematisk sjekke hva den returnerer for ulike argumenter
- Presist spesifisere hva en funksjon skal gjøre
  - Vi bruker det f.eks. for å lage presise oppgavetekster: *"Altså skal f.eks. kallet pris (true, 10) returnere 0, kallet pris(false,10) returnere 100 ..."* (oppg 4, eksamen 2014)
- Gjøre oss oppmerksom på hva en funksjon må håndtere før vi skriver koden for den!
  - Dette kalles test-drevet utvikling, og betyr at testene lages først og inspirerer utvikling av selve koden i funksjonen

Test-drevet utvikling

- Når man vet hva en funksjon skal gjøre kan man skrive testen før selve funksjonen!
  - `assert gang_med_to(3) == 6`
- Man kan ikke teste alle muligheter, men forsøk å dekke litt variert type argumenter
  (ulikt som kunne tenkes å feile)
  - ```
    assert gang_med_to(0) == 0
    assert gang_med_to(-3) == -6
    assert gang_med_to(2.4) == 4.8
    ```
    ```

Test-drevet utvikling

- Skriv deretter gjerne en (tom) funksjon som skal feile
  - ```
    def gang_med_to(tall):
        return 0
    ```
- Forsøk til slutt å skrive riktig funksjon
  - ```
    def gang_med_to(tall):
        return tall*2
    ```
  - Se at programmet (asserts) ikke lenger feiler ved kjøring
- {ganging.py}