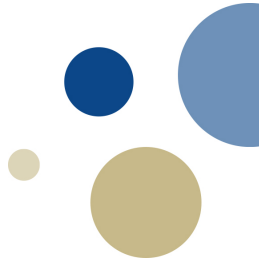
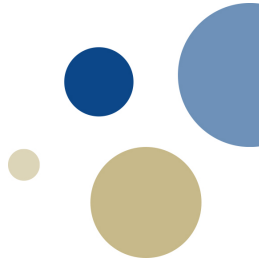
# Datatyper



Variabler har bestemte typer, men kan byttes med tilordningsoperatoren.

```
a = "hei" # type er str
a = 1 # type er int
```

# Datatyper



Noen aritmetiske operatorer fungerer for andre typer enn tall, men ikke alltid.

```
liste1 = [1, 2, 3]
liste2 = [4, 5]
liste3 = liste1 + liste2

liste4 = liste3 * 2
```

# Datatyper

Eksempler på datatyper:

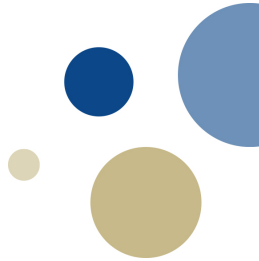
- int: heltall, 1023913213, 0, -1
- float: flyttall, 0.1, 1.2e-10, 3.1415e6
- str: streng, "hei", "sacasdcasdc"
- list: liste, [1, 2, 3], ["hei", 1.3e2]
- tuple: tupler, (1, 2, 3), ("hei", 1.3e2)

# Datatyper

Funksjoner for å konvertere mellom datatyper. De gjør forskjellige ting for forskjellig input-typer, og fungerer ikke for alle.

| Funksjon | Eksempel | Resultat |
|---|---|---|
| bin() | bin(92) | ’0b1011100’ |
| bool() | bool(12), bool(0) | True, False |
| chr() | chr(97) | ’a’ |
| ord() | ord(’a’) | 97 |
| float() | float(1) | 1.0 |
| int() | int(2.6), int("2") | 2, 2 |
| list() | list(range(2)) | [0, 1] |
| str() | str(100) | ’100’ |

# Datatyper



Type gir hvilken datatype en variabel er, dermed kan vi sjekke med en if. Kan også bruke *isinstance*

```
a = 1.23
if type(a) == float:
    print("a er en float")
if isinstance(a, float):
    print("a er fortsatt en float")
```

# Datatyper

Dette kan ofte være nyttig i funksjoner for å gjøre forskjellige ting for forskjellige parametere.