DIGI - Jukseark - Python

Datatyper

Man kan sjekke typen til et objekt med funksjonen `type()`.

Navn (norsk)	Navn (engelsk)	Eksempel	Konvertere til
Heltall	Integer	`37`	`int()`
Flyttall (desimaltall)	Float	`23.5`	`float()`
Streng (tekst)	String	`"Hei"` eller `'Hei'`	`str()`
Liste	List	`["Nora", "Odin", "Morten"]`	`list()`
Ordbok	Dictionary	`{"navn": "Odin", "alder": 32}`	`dict()`
Boolsk	Boolean	`True`, `False`, `12>0 and 12<28`	`bool()`
Sett	Set	`{3, -1, 0}`	`set()`

`None` er et spesielt objekt som representerer fraværet av en verdi eller en null-verdi. `None` er et unikt objekt av typen `NoneType`.

Tall og matte

Operator	Operasjon	Eksempel
`+`	Addisjon	`2 + 2 = 4`
`-`	Subtraksjon	`5 - 2 = 3`
`*`	Multiplikasjon	`3 * 3 = 9`
`/`	Divisjon	`22 / 8 = 2.75`
`//`	Heltallsdivisjon	`22 // 8 = 2`
`**`	Eksponent (opphøyd i)	`2 ** 3 = 8`
`%`	Modulo/Rest	`22 % 8 = 6`

Tekst (streng)

Tekstvariabler (streng):

```
setning = "Hei hei, verden!"
navn = "Odin"
# Indeksen til det første elementet er `0`
bokstav = navn[1] # bokstav = "d"
```

Skrive ut tekst:

```
alder = 28
# Skrive ut enkle strenger
print("Min melding: ", setning)
# Skrive ut tekst med variabler (f-strenger)
print(f"Jeg heter {navn}. Jeg er {alder} år gammel.")
```

Be brukeren om å gi oss en streng:

```
navn = input("Hva heter du?")
# Husk å konvertere strengen hvis du vil ha en heltall eller flyttall
alder = int(input("Hvor gammel er du?"))
```

Funksjoner

```
def areal_rektangel(lange, korte):
    """
    Beregn arealet av et rektangel gitt dens korte og lange sider.
    """
    return korte * lange
```

Funksjoner må *kalles opp* for at de skal kjøre, det gjøres med `funksjonsnavn(parameter1, parameter2, ...)`

```
areal = areal_rektangel(10, 5) # Funksjonskall
print("Arealet av rektangelet er", areal)
```

Funksjoner trenger ikke å ha parametere, da defineres de med en tom parentes. En funksjon alltid returnerer noe, selv om det finnes ingen `return`-setning! Uten en `return`-setning, returneres `None`.

If-setninger

Maksimalt én kodeblokk kjøres! (hvis det er en 'else'-blokk, er det alltid nøyaktig én).

If

```
alder = 28
if alder >= 18:
    print("Du er en voksen")
```

If-else

```
```

```
alder = 28
if alder >= 18:
    print("Du er en voksen")
else:
    print("Du er en barn")
```

If-elif

```
alder = 28
if alder >= 18:
    print("Du er en voksen")
elif alder >= 13:
    print("Du er en tenåring")
```

If-elif-else

```
alder = 28
if alder >= 18:
    print("Du er en voksen")
elif alder >= 13:
    print("Du er en tenåring")
else:
    print("Du er et barn")
```

If-elif-elif-else

Det er ingen begrensning på hvor mange elif man kan ha i en if-setning.

```
alder = 28
if alder >= 18:
    print("Du er en voksen")
elif alder >= 13:
    print("Du er en tenåring")
elif alder < 1:
    print("Du er et spedbarn")
elif alder < 5:
    print("Du er et småbarn")
else:
    # 5 <= alder < 13
    print("Du er et barn")
```

Sammenligninger

Operator	Betydning

Operator	Betydning
`==`	lik
`!=`	ikke lik
`<`	mindre enn
`>`	større enn
`<=`	mindre enn eller lik
`>=`	større enn eller lik
`and`	og
`or`	eller
`not`	ikke

Lister

```
tom_liste = []
land_liste = ["Norge", "Danmark", "Sverige"]
karakter_liste = [4, 3, 2, 8, 3]
rotete_liste = [2024, 0.1, "DIGI", True, None]

# Indeksen til det første elementet er `0`
print(land_liste[1])          # "Danmark"
```

Ordbøker

```
tom_ordbok = {}

hovedsteder = {
    "Norge": "Oslo",          # "Norge" er *nøkkelen* og "Oslo" *verdien*
    "Danmark": "København",
    "Sverige": "Stockholm"
}

# Får tilgang til en verdi ved å bruke dens tilsvarende nøkkel.
print(hovedsteder["Danmark"]) # "København"

# Oppdater en verdi
hovedsteder["Norge"] = "Bergen"

# Opprett et nytt nøkkel-verdi par
hovedsteder["Frankrike"] = "Paris"
```

Løkker

For-løkke med range

```
# Teller fra 0 til 9
# range(slutt)
for i in range(10):
    print(i)
```

```
# Teller fra 5 til 9
# range(start, slutt)
for i in range(5, 10):
    print(i)
```

```
# Teller fra 5 til 15 med steg på 2 (5,7,9,11,13,15)
# range(start, slutt, steg)
for i in range(5, 16, 2):
    print(i)
```

```
# Teller fra 10 til 6 med steg på -1 (10,9,8,7,6)
for i in range(10, 5, -1):
    print(i)
```

For-løkke og sekvenser

```
# Skriver ut landene i listen en etter en
liste = ["Norge", "Sverige", "Danmark"]
for land in liste:
    print(land)

# Skriver ut bokstavene og tegnene en etter en
streng = "Hallo, verden!"
for bokstav in streng:
    print(bokstav)
```

For-løkke og ordbok

```
hovedsteder = {
    "Norge": "Oslo",
    "Danmark": "København",
    "Sverige": "Stockholm"
}

# Skriver ut nøkler og verdier
```

```
for (land, hovedstad) in hovedsteder.items():
    print(f'Nøkkel: {land}  |  Verdi: {hovedstad}')
```

While-løkke

```
# Teller fra 0 til 5, og skriver ut "Etter løkken"
i = 0
while i <= 5:
    print(i)
    # Oppdater indeksvariabelen
    i = i + 1
print("Etter løkken")
```

Tekst fil

For å håndtere tekst fil i python, bruker vi `open()` funksjonen i tekst-modus (`"t"`).

Lese fra en tekst fil

Bruk "read"-modus (`"r"`) om du vil lese en fil.

```
with open("min_fil.txt", mode="rt") as f:
    # Les alle linjene og lagre dem i en liste: hvert element
    # i listen er en linje
    linjene = f.read().split("\n")

# Jobb med linjene slik du ønsker
print(linjene)
# ...
```

Skrive i en tekst fil

Bruk "append"-modus (`"a"`) om du vil legge til tekst på slutten av en fil.

```
# Husk å legge til "\n" på slutten av hver linje for å
# ha et linjeskift mellom linjene
nye_linjer = ["Hei! Vi lærer Python\n", "Hva lærer du?\n"]
```

```
with open("min_fil.txt", mode="at") as f:
    # Skrive en streng
    f.writelines("Vi Legger til få linje: \n")
    # Skrive en liste av linjer
    f.writelines(nye_linjer)
```

Bruk "write"-modus (`"w"`) om du vil overskrive en fil med en ny tekst.

```
with open("min_fil.txt", mode="wt") as f:
    f.writelines("Helt nytt innhold: \n")
    f.writelines(nye_linjer)
```

Vanlige innebygde funksjoner og metoder

Innebygde funksjoner

Sekvens metoder

```
liste = [2024, 0.1, "DIGI", True, None]
streng = "Hei hei, verden!"
```

Indeksering syntaks: `liste[start:slutt:steg]` med standardverdier (dvs. om utelatt):

- `start = 0`
- `slutt = -1`
- `steg = 1`

Metode / Operasjon	Resultat	Beskrivelse
`liste[1]`	`0.1`	Hent ut det som ligger på indeks `1`
`streng[-1]`	`"!"`	Hent ut det som ligger `-1` fra slutten av sekvensen
`liste[-2]`	`True`	Hent ut det som ligger `-2` fra slutten av sekvensen
`streng[:3]`	`Hei`	Hent ut de som ligger fra indeksen 0 til (ikke inkludert) indeksen 3
`liste[1:4]`	`[0.1, 'DIGI', True]`	Hent ut de som ligger fra indeksen 1 til (ikke inkludert) indeksen 4
`streng[1:11:2]`	`'e e,v'`	Hent ut hver annen element, startende fra indeksen 1 til (ikke inkludert) indeksen 11
`len(liste)`	`5`	Lengden på sekvensen
`len(streng)`	`16`	Lengden på sekvensen
`["DIGI", True] + [2024, 0.1]`	`["DIGI", True, 2024, 0.1]`	Slå sammen sekvenser
`'verd' in streng`	`True`	Sjekk om noe er i sekvensen
`None not in liste`	`False`	Sjekk om noe ikke er i sekvensen
`liste.index("DIGI")`	`2`	Finn plassering til noe i sekvensen

Foranderlige sekvens metoder

Disse metodene er "in-place"-metoder. Det betyr at de ikke returnerer sekvensen, og i stedet oppdaterer de sekvensen på stedeten. Dette er mulig fordi vi her jobber med foranderlige sekvenser.

Metode / Operasjon	Resultat	Beskrivelse
`liste[1] = "!"`	`[2024, '!', 'DIGI', True, None]`	Sett det som ligger på indeks `1` til `"!"`
`liste.append(42)`	`[2024, '!', 'DIGI', True, None, 42]`	Legg noe til på slutten av sekvensen
`liste.insert(1, 42)`	`[2024, 42, '!', 'DIGI', True, None, 42]`	Sett noe inn i sekvensen på en gitt indeks
`liste.remove(42)`	`[2024, '!', 'DIGI', True, None, 42]`	Fjern et gitt element fra sekvensen.
`liste.pop(2)`	`[2024, '!', True, None, 42]`	Fjern elementet som ligger på en gitt indeks

Liste metoder

```
karakterene  = [1, 8, 3, 0, 10, 1]
vennene = ["Nora", "Odin", "Morten"]
```

Metode / Operasjon	Resultat	Beskrivelse
`karakterene.sort()`	`[0, 1, 1, 3, 8, 10]`	Sorter listen (om mulig!)
`vennene.sort(reverse=True)`	`['Odin', 'Nora', 'Morten']`	Sorter listen (om mulig!) motsatt vei

Streng metoder

Disse metodene er ikke "in-place"-metoder. De returnerer en ny sekvens og lar den opprinnelige sekvensen være intakt. Dette skyldes at vi her jobber med uforanderlige sekvenser.

Metode / Operasjon	Resultat	Beskrivelse
`streng.upper()`	`"HEI HEI, VERDEN!"`	Gjør alle bokstaver store
`streng.lower()`	`"hei hei, verden!"`	Gjør alle bokstaver små
`streng.startswith('hei')`	`False`	Sjekke om strengen begynner med
`streng.endswith('!')`	`True`	Sjekke om strengen slutter med
`streng.split(",")`	`['Hei hei', ' verden!']`	Deler strengen i en liste
`streng.split(" ")`	`['Hei', 'hei,', 'verden!']`	Deler strengen i en liste

Metode / Operasjon	Resultat	Beskrivelse
`streng.replace('e', 'i')`	`Hii hii, virdin!`	Erstatter en understreng i strengen
`streng.replace('ver', 'klo')`	`Hei hei, kloden!`	Erstatter en understreng i strengen
`" og ".join(vennene)`	`"Nora og Odin og Morten"`	Slår en list sammen til en streng