Viktig å vite – Uke 2

*Stikkord:
Input, feilmeldinger, evaluering av uttrykk, typekonvertering, prosedyrer*

Input:

input() er enda en biblioteksfunksjon som er viktig når man skal interagere med brukeren av programmet, og det gjør programmet vårt mer dynamisk.

input(arg) tar et argument som skal være at typen streng. Denne strengen skrives ut til brukeren og kan være et spørsmål eller en instruksjon til hva brukeren skal skrive inn i terminalen.

*Eks:*

*navn = input(‘’Skriv inn navnet ditt: ‘’)*

*Verdt å vite at input alltid returnerer/blir til en streng etter at brukeren skriver det inn, selv om brukeren skriver inn et tall.*

Feilmeldinger:

En feilmelding oppstår dersom python ikke greier å forstå hva som står i koden. Dette skjer kun på grunn av at man har feil i koden, eller at koden er unøyaktig. I feilmeldingen står det også hvor i koden feilen er og hvordan type feil som er oppstått.

Type feilmeldinger som kan være viktig å lære seg:

**SyntaxError 🡪** Oppstår når man glemmer å følge syntax-reglene til python. Kan skje dersom man glemmer et kolon, gåsetegn eller en parentes. (Oppstår ofte)

*Eks:*

*print(‘’hello verden’’ #får feilmelding pga. manglende parantes.*

*setning = hei jeg heter peter #får feilmelding pga. manglende gåsetegn.*

**NameError** 🡪 oppstår dersom du bruker en variabel som ikke er blitt deklarert til en verdi.

*Eks:*

*navn = ‘’Roger’’*

*print(namn)* #vil skrive ut feilmelding ettersom at alder ikke har blitt tilordnet en verdi

**TypeError** 🡪 typen til objektet/variabelen som er brukt passer ikke med måten den er brukt på.

*Eks:*

*navn = ‘’Peter’’*

*alder = 22*

*print(navn+alder) #Vil føre til TypeError ettersom at ikke en streng kan plusses sammen med et heltall.*

**ValueError** 🡪 verdien til en variabel/objekt fører til at python ikke forstår hva som skjer.

*Eks:*

*tall1 = ‘’tre’’*

*tall2 = int(tall1) #Fører til feilmelding ettersom at du ikke kan gjøre ‘’tre’’ til en int.*

Evaluering av uttrykk:

Dette vil si at man avgjør om et utrykk vil bli True eller False.

- For at **and** skal gi sant må begge uttrykk være sanne, feks. a and b evalueres kun til sant dersom både a er true OG b er true.

- For at **or** skal gi sant må ett av uttrykkene være sanne, feks. a and b evalueres kun til sant dersom enten a er true eller b er true eller begge er true.

(Viktig at det ikke er enten-eller!)

- **not** gjør ett uttrykk til det motsatte av hva det er. Dersom a er usann, vil not(a) være sann. Dersom b er sann, vil not(b) være usann.

*Eks:*

*True and False => False*

*True or False => True*

*not False => True*

|  |  |
| --- | --- |
| a == b | a er lik b |
| a != b | a er ikke lik b |
| a > b | a er større enn b |
| a >= b | a er større eller lik b |
| a < b | a er mindre enn b |
| a <= b | a er mindre eller lik b |

Også ulike operasjoner kan bli True eller False.

*Eks:*

*1 == 1 => True*

*1 == 0 => False*

*2 > 1 => True*

*10 <= 0 => False*

*2 >= 2 => True*

Typekonvertering:

Det er mulig å konvertere objekter fra en type til en annen. Dette kan gjøres ved ulike metoder og funksjoner. De mest vanlige er int(arg), float(arg) og str(arg).

*Eks:*

*a = 3 #a er typen heltall*

*b = str(a) #b er typen String*

*a = float(a) #a er nå av typen float, a = 3.0*

Pass på at det som konverteres kan konverteres til den typen som den skal konverteres til, ellers kan det fort dukke opp en feilmelding av typen ValueError.

Prosedyrer: (Viktig)

En prosedyre er en måte man kan lagre en algoritme/flere linjer med kode i et program uten at det vil automatisk kjøre. Dette er spesielt smart å bruke dersom man skal repetere den samme koden flere ganger i løpet av programmet.

*Eks*

*def gangeTo():*

 *tall = int(input(‘’Skriv inn et tall som skal ganges med 2: ’’))*

 *print(tall\*2)*

*gangeTo()*

*gangeTo()*

*gangeTo() #Spør 3 ganger om å gange et tall med 2.*

Veldig viktig for å gjøre koden oversiktlig og ryddig.