Innleveringsoppgaver uke 42

Introduksjon

Framover, dersom jeg ber deg om å «lage en funksjon `absolutt_differanse` som tar to paramaterer og returnerer absoluttverdien av de to parameternes differanse», kan det formelt oversettes til følgende pythonkode:

```
def absolutt_differanse(ledd1, ledd2):
    """Returnererer absoluttverdien til differanse mellom 'ledd1' og 'ledd2'."""
    diff = ledd1 - ledd2
    return abs(diff)
```

Etter at du har kjørt programmet, kan du sjekke at funksjonen er opprettet ved å skrive `help(absolutt_differanse)` i konsollen. Du skal da få ut noe à la følgende:

```
Help on function absolutt_differanse in module __main__:

absolutt_differanse(ledd1, ledd2)
    Returnererer absoluttverdien til differanse mellom 'ledd1' og 'ledd2'.
```

Videre kommer jeg til å gi tester funksjonen skal bestå, eksempelvis som i tabellen under.

`ledd1`	`ledd2`	`absolutt_differanse(ledd1, ledd2)`
1	1	0
-5	15	20
15	-5	20
-15	-5	10
0	-5	5

Du kan da teste funksjonen din ved å kontrollere at den returnerer riktige verdier, ved for eksempel å skrive i konsollen `absolutt_differanse(-5, 15)` for å se at det evaluerer til verdien `20`.

Matematiske funksjoner

Som andre navn i python, tilstreber vi at også funksjonsnavnene er så beskrivende som mulig. Bokstaver er billige, og du har lettere for å forstå hva funksjonen `kvadratrot` gjør, enn `frt`.

Unntaket til dette er når vi definere enkle matematiske funksjoner, som $f(x) = 3x^2 \cdot \lg x - 15$. Vi ville altså fått:

```
import math

def f(x):
    return 3 * x ** 2 * math.log10(x) - 15
```

(Her har jeg også droppet å ta med en dokumentasjonsstreng.)

Oppgave 1. Bruk funksjonen over og lag et program som skriver ut ti x- og tilhørende $f(x)$-verdier for funksjonen over.

For x-verdiene fra og med 20, til og med 29, får vi altså følgende utskrift:

```
x     --     f(x)
-----------------------
20    --    1546.2
21    --    1734.3
22    --    1934.2
23    --    2146.1
24    --    2370.0
25    --    2606.1
26    --    2854.6
27    --    3115.4
28    --    3388.7
29    --    3674.6
```

Kall fila `f"uke42_{fornavn}_matfunk.py"` (altså, `uke41_trond_matfunk.py`, om du er så heldig å hete Trond).

Typer og funksjonstyper

Vi har tidligere snakket om at ulike verdier har ulike *type*. For eksempel er `3` av typen `int`, altså et heltall eller «integer». Vi skal nå se på litt flere typer, blant annet *funksjoner*.

Oppgave 2. Skriv av det følgende pythonprogrammet og fyll ut det som står de doble likhetstegnene, lever fila som `f"uke42_{fornavn}_typer.py`.

```
def utregning(x):
    """Tar inn et tall x, og gjør litt matte."""
    dobbel_x = 2 * x
    litt_mindre = dobbel_x - 5
    return litt_mindre * dobbel_x

def er_stort(testtall, stort_tall):
    """Tar inn to tall og returnerer hvorvidt 'testtall'
    er større enn 'stort_tall'.
    """
    returverdi = False
    if testtall > stort_tall:
        returverdi = True
    return returverdi


# I alle linjene under, fyll ut det som står i anførselstegn
# når du skriver venstre side inn i konsollen.

# Eksempel:
# type(3.1) == 'float'
# siden
# >>> type(3.1)
# <class 'float'>

# Oppgaver:
# type(3) ==
# type(2 + 2) ==
# type(utregning(5)) ==
```

```
svar = utregning(5)

# type(svar) ==
# type(er_stort(3, 2)) ==
# type(er_stort(2, 3)) ==
# type(er_stort) ==
```

Derivering av funksjoner

Fra matematikken er vi vant til å innføre den deriverte via dens *definisjon*.

Definisjon 1. Den derivert til funksjonen $f(x)$ ved punktet $x = a$ finner vi som

$$f(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Vi skal nå bruke denne definisjonen til å regne ut den deriverte for en matematisk funksjon, $f(x) = 3x^2 \cdot \lg x - 15$.

Om jeg ønsker å finne verdien til den deriverte av denne funksjonen i, for eksempel, $x = 4$, kan vi finne et bedre og bedre overslag for dette ved å bruke mindre og mindre verdier for Δx.

```
import math

def f(x):
    return 3 * x ** 2 * math.log10(x) - 15

delta_x = 0.001    # velger en veldig liten verdi for delta x
eval_x = 4
delta_f = f(eval_x + delta_x) - f(eval_x)

derivert = delta_f / delta_x

print(f"Den deriverte av funksjonen ved x = {eval_x} er omtrent {derivert:.2f}.")
```

```
Den deriverte av funksjonen ved x = 4 er omtrent 19.66.
```

Denne måten å regne ut en derivert på kalles en *framoverdifferanse*, siden vi finner den deriverte i x ved å *legge til* Δx.

Oppgave 3. Utvid tabellen fra forrige oppgave til også å ha en kolonne for den deriverte av funksjonen ved de ulike x-verdiene. Bruk *framoverdifferanse* for å finne den deriverte. Lever fila som `f"uke42_{fornavn}_framoverdiff.py`.

```
x     --     f(x)     --     f'(x)
-------------------------------------
20    --    1546.2    --    182.19
21    --    1734.3    --    193.97
22    --    1934.2    --    205.87
23    --    2146.1    --    217.89
24    --    2370.0    --    230.03
25    --    2606.1    --    242.27
26    --    2854.6    --    254.62
27    --    3115.4    --    267.07
28    --    3388.7    --    279.61
29    --    3674.6    --    292.25
```

Utfordring: Derivasjonsfunksjon

Lag en funksjon med følgende spesifikasjon:

```
def derivert_av_funk_ved_x(funksjon, x):
    """Returnerer den numeriske verdien av den deriverte av funksjonen
    'funksjon' når x er 'x'.

    Bruk en 'framoverdifferanse' (se forrige oppgave) med 'delta_x = 0.0001'.

    Eksempel:
    def f(x):
        return x ** 2

    >>> derivert_av_funk_ved_x(f, 2)   =>  4.0001000000078335

    """

    # Her gjør du utregningene dine
```

Merk, her sender vi *navnet til en funksjon* inn som en parameter til funksjonen.

Oppgave 4. Lag en funksjon som tar inn navnet på en funksjon og en x-verdi. Returner et overslag for den deriverte til funksjonen ved denne x-verdien ved hjelp av framoverdifferanse med en $Deltax = 0.0001$. Lever fila som `f"uke42_{fornavn}_framoverdiffunk.py`.