Øving 2

INF620 - Vårsemesteret 2021

Øvingsoppgavene er ikke obligatoriske, men vi anbefaler likevel at du gjør de og leverer de innen fristen — Den eneste måten å lære å programmere på er ved å programmere. Ved å gjøre oppgavene får du også testet deg selv og sjekket at du forstår begrepene. Du skal levere én zip-fil, `oving2.zip`, som inneholder de 4 filene `oppg1.py`–`oppg4.py`. For å komprimere en eller flere filer til en zip-fil høyreklikker du filene (i dette tilfellet `oppg1.py`–`oppg4.py`) i maskinens filnavigasjonsprogram og velger `Komprimer` eller `Send til` → `Komprimert mappe`. Frist: Torsdag 16. september kl 23:59

1 Aritmetikk

Svar leveres på fil med navn `oppg1.py`

I forelesningene har du lært om en rekke matematiske operatorer. I de neste oppgavene skal du bruke disse til å gjøre utregninger. Prøv å komme frem til svaret for hånd før du sjekker svaret i Python. Da forsikrer du deg om at du forstår hvordan python bruker operatorene.

1.a
Lag en rekke instrukser som skriver disse regnestykkene til konsollen. Kommenter svaret i en kommentar bak linjen, for eksempel

```
print(1 + 2) # 3
```

Prøv å se for deg svarene før du evaluerer uttrykkene. Kommer svaret til å være et heltall eller flyttall?

- `9 - 3`
- `8 * 2.5`
- `9 / 2`
- `9 / -2`
- `9 // -2`
- `9 % 2`
- `9.0 % 2`
- `9 % 2.0`

- `9 % -2`
- `-9 % 2`
- `9 / -2.0`
- `4 + 3 * 5`
- `(4 + 3) * 5`
- `abs(5 - 10)`

1.b
Skriv instrukser med regnestykkene som er beskrevet nedenfor. Du kan putte instruksene i print()-metoder, men du trenger ikke å skrive svaret som kommentar. For eksempel blir "10 minus 4" til `print(10 - 4)`. Prøv å ikke bruke parentes hvis ikke du må. (Merk at dette er for oppgaven sin del, i noen tilfeller kan parentes gjøre koden lettere å lese selv om det ikke er nødvendig).

- 3 opphøyd i potens 6.
- 4 multiplisert med 2, og dette resultatet addert med 3.
- 4 pluss 2, og dette resultatet multiplisert med 3.
- Antall hele ganger 6 går opp i 100
- Resten man får ved å dele 100 på 6
- 100 delt nøyaktig på 6

Første påskedag er den første søndagen etter den første fullmånen etter vårjevndøgn. For å regne ut datoen kan du bruke følgende algoritme oppfunnet av matematikeren Carl Friedrich Gauss i 1800:

(Kvotienten er det heltallet du får ved å dele et heltall på et annet. Eks: kvotienten til 11/5 er 2).

1. La y være året (f.eks. 1800 eller 2020).
2. Del y på 19 og kall resten for a. Ignorer kvotienten.
3. Del y på 100 slik at du får en kvotient b og en rest c.
4. Del b på 4 slik at du får en kvotient d og en rest e.
5. Del $8 * b + 13$ på 25 slik at du får en kvotient g. Ignorer resten.
6. Del $19 * a + b - d - g + 15$ på 30 slik at du får en rest h. Ignorer kvotienten.
7. Del c på 4 slik at du får en kvotient j og en rest k.
8. Del $a + 11 * h$ på 319 slik at du får en kvotient m. Ignorer resten.
9. Del $2 * e + 2 * j - k - h + m + 32$ på 7 slik at du får en rest r.
10. Del $h - m + r + 90$ på 25 slik at du får en kvotient n. Ignorer resten.
11. Del $h - m + r + n + 19$ på 32 slik at du får en rest p. Ignorer kvotienten.

Første påskedag faller da på dag p av måneden n.

Lag et program som spør brukeren om et årstall og regner ut datoen for første påskedag det året. De første linjene i programmet kan være

```
y = int(input("Skriv et årstall: ")) # Spør brukeren om et årstall y
a = y % 19 # Fra instruks 2
```

Programmet må skrive ut p og n. Her er en eksempelkjøring av programmet der brukeren skrev 2020:

```
Skriv et årstall: 2020
Første påskedag faller på dag 12 av måneden 4.
```

2 Datatyper

Svar leveres på fil med navn `oppg2.py`

Verdier og variabler i Python har en bestemt datatype. Eksempel på datatyper er heltall (`int`), flyttal (`float`) og tekststrenger (`str`). Det er viktig å forstå hvilken datatype man jobber med for å vite hva man kan og ikke kan gjøre med verdien — misforståelser av typer er en vanlig grunn til Python-program krasjer.

De følgende oppgavene besvares ved å skrive svaret til konsollen. F.eks.
"`x = 5 + 3`, hvilken datatype har x?"
Svar:
`print("x har datatypen int")`.

2.a
Hvilken datatype har `x` i følgende scenarioer? Dersom instruksen ikke er lovlig, forklar kort hvorfor.

- `x = 10`
- `x = 10 + 10`
- `x = 5.5`
- `x = 10 + 5.5`
- `x = 5.5 + 5.5`
- `x = "abc"`
- `x = "abc" + "5"`
- `x = "abc" + 5`
- `x = "abc" + str(5)`
- `x = "5" + 5`
- `x = int("5") + 5`

3 Formaterte strenger

Svar leveres på fil med navn `oppg3.py`

For tekstbeskjeder fungerer vanlige tekststrenger helt fint, men for desimaltall og strukturert informasjon er det ofte praktisk å bruke formaterte strenger. Da erstatter du informasjonen i strengen med en `%`, etterfulgt av formateringsinformasjon, etterfulgt av datatypen for informasjonen. Bak strengen skriver du `% (...)` og fyller inn infoen i parentesene, separert med komma. For eksempel, hvis vi vil formatere et flyttall til å ta opp 8 plasser med en presisjon på 3 desimaler, og en streng til å ta opp 12 plasser, kan du skrive

```
tekst = 'Flyttall: %8.3f. Streng: %12s' % (3.1415926, 'tekst')
print(tekst)
```

Ved kjøring av programmet printes følgende:

```
Flyttall:    3.142. Streng:        tekst
```

Hvis vi hadde skrevet `%.3f` hadde Python kun brydd seg om antall desmialer, ikke antall plasser. Merk at vi skrev 'f' fordi vi hadde et flyttall. For heltall bruker vi 'd', og for strenger bruker vi 's'. Det finnes også en annen måte å formatere strenger på, vi bruker den 'gamle' måten. Du kan velge selv hvilken du vil bruke, men vær konsistent. For mer informasjon om hvordan du formaterer strenger kan du gå til pyformat.info (vi bruker det nettsiden kaller den "gamle" måten).

3.a
Gå til toppen av Python-filen (`oppg3.py`) og skriv

```
from math import pi
```

Vi kan nå bruke konstanten `pi` i koden vår. Opphøy pi i potens 2, og print ut svaret med 2 desimaler slik: "Svar: ...". Du bestemmer selv om du vil gjøre alt i en linje eller bruke flere linjer, men pass på at du bruker passende variabelnavn dersom du introduserer variabler.

3.b
Formatkoden `%20s` setter av 20 plasser og høyrejusterer teksten på disse plassene. `%-20s` gjør det samme, men venstrejusterer teksten. Print en linje med 4 kolonner, hver på 13 plasser. Den første kolonnen skal være venstrejustert og inneholde teksten "Navn". De tre neste kolonnene skal være høyrejustert og inneholde 3 navn på mindre enn 13 bokstaver. Eksempel:

```
Navn                 Ola          Lisa          Kari
```

3.c
Nå har vi verktøyet vi trenger for å printe ut en tabell. Lag et program som printer ut tabellen over snødybder. Bruk kolonneinnstillingene fra oppgave 3.b, og bruk 2

desimals presisjon.

Sted	27.jan	28.jan	29.jan
Geilo	0.482	0.501	0.440
Hemsedal	0.472	0.455	0.454
Sirdal	0.253	0.212	0.200

Resultatet skal se slik ut:

```
Sted              27. jan      28. jan      29. jan
-----------------------------------------------------
Geilo               0.48         0.50         0.44
Hemsedal            0.47         0.46         0.45
Sirdal              0.25         0.21         0.20
```

Tips: Når du skal printe en rad med bindestreker, i stedet for å skrive `'----...'` kan du skrive `'-' * 56`, så gjentas bindestreken 56 ganger.

4 Input fra brukeren

Svar leveres på fil med navn `oppg4.py`

4.a
Skriv et program som spør brukeren om to heltall, for så å printe

- summen
- differansen
- produktet
- gjennomsnittet
- avstanden (absoluttverdien av differansen)
- maksimum
- minimum

Print ut svarene slik at tallene blir riktig justert:

```
Sum:           45
Differanse:    -5
Produkt:       500
Gjennomsnitt:  22.50
Distanse:      5
Maksimum:      25
Minimum:       20
```