Øving 2

INF621 - Høstsemesteret 2021

Sammendrag

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 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 11. november kl 23:59

1 Hvilken dag er det i dag? (24%)

For å kunne skrive ut en dato eller et klokkeslett må vi først formatere informasjonen som en streng. I hver av deloppgavene under lager vi en funksjon som skriver dato og/eller tid på et format spesifisert av oppgaven. Funksjonen skal skrive den tiden og/eller datoen som gjelder ved kjøringen av programmet. Alle eksempelkjøringene er tenkt utført 04.11.21 kl 15:59:55.

1.a
(8%) Først lager vi en funksjon `hva_er_klokken()` som skriver hva klokken er.

```
In [1]: hva_er_klokken()
Klokken er 15:59
```

1.b
(8%) Så lager vi en funksjon `ukedag()` som skriver hvilken ukedag det er idag.

```
In [2]: ukedag()
Det er torsdag!
```

1.c
(8%) Til slutt lager vi en funksjon `tidspunkt()` som skriver ut ukedag, dato og tidspunkt på dagen.

```
In [3]: tidspunkt()
Det er torsdag 4. november og klokken er 15:59:55
```

Som et alternativ til å kjøre koden fra konsollen, kan du på slutten av filen `oppg1.py` legge til

```
if __name__ == '__main__':
    hva_er_klokken()
    ukedag()
    tidspunkt()
```

2 Parsing av dato og tid (30%)

Det er også viktig å kunne lese dato og tid fra en tekststreng. Tekststrengen kan komme som input fra brukeren av programmet, fra en fil eller fra nettet. Siden dato og tid kan skrives på ulike måter er det opp til deg som programmerer å forklare Python hvordan strengen skal tolkes som et bestemt tidspunkt. Oversettelsen av en tekst til noe som er forståelig for et programmeringsspråk kalles parsing.

I de følgende deloppgavene skal du lese inn et tidspunkt fra brukeren på det formatet som er oppgitt i oppgaveteksten. Funksjonen skal returnere et objekt av klassen `datetime`. At en dato er på formatet (d)d.(m)m.yy betyr at input kan være enten '04.11.21' eller '4.11.21'. Tilsvarende er det valgfritt om ensifrede månedsnummer skal ha en innledende 0.

2.a
(10%) Lag en funksjon som parser en dato på formatet: 'dd.mm.yy'

```
In [4]: parse_1('04.11.21')
Out[4]: datetime.datetime(2021, 11, 4, 0, 0)
```

2.b
(10%) Lag en funksjon som parser en dato på formatet: '(d)d. navn-på-måned yyyy'

```
In [5]: parse_2('4. november 2021')
Out[5]: datetime.datetime(2021, 11, 4, 0, 0)
```

2.c
(10%) Lag en funksjon som parser en dato på formatet: '(d)d/(m)m-yyyy TT:MM:SS'

```
In [6]: parse_3('04/11-2021 15:59:55')
Out[6]: datetime.datetime(2021, 11, 4, 15, 59, 55)
```

Som et alternativ til å kjøre koden fra konsollen, kan du på slutten av filen `oppg2.py` legge til

```
if __name__ == '__main__':
    print(parse_1('04.11.21'))
```

```
print(parse_2('4. november 2021'))
print(parse_3('04/11-2021 15:59:55'))
```

3 Regning med dato og tid (16%)

I denne oppgaven skal vi øve på å regne med `datetime` og `timedelta` objekter.

3.a
(16%) Lag en funksjon `tidspunkt_om(n)` som skriver tidspunktet som inntreffer nøyaktig `n` sekunder etter programmet kjøres.

```
In [7]: tidspunkt_om(1)
2021-11-04 15:59:56
In [8]: tidspunkt_om(60)
2021-11-04 16:00:55
In [9]: tidspunkt_om(31536000)
2022-11-04 15:59:55
```

3.b
BONUS: (0%) Lag en funksjon `nedtelling()` som skriver hvor mange hele dager det er fra nå til a) nyttårsdag b) slutten av denne måneden c) eksamensdagen 17. desember d) julaften e) 4. november, klokken 15:59:54.

Hint: Lag en (eller flere) hjelpefunksjoner for å gjøre koden litt kortere.

```
In [10]: nedtelling()
Antall hele dager igjen til nyttårsdag: 57
Antall hele dager igjen til neste måned: 26
Antall hele dager igjen til eksamen: 42
Antall hele dager igjen til julaften: 49
Antall hele dager igjen til 4. november klokken 15:59:54 : 364
```

Som et alternativ til å kjøre koden fra konsollen, kan du på slutten av filen `oppg3.py` legge til

```
if __name__ == '__main__':
    tidspunkt_om(1)
    tidspunkt_om(60)
    tidspunkt_om(31536000)
    nedtelling()
```

4 Temperaturmålinger (30%)

I denne siste oppgaven skal vi øve litt på ordbøker (oppslagstabeller, `dict`), tupler og sortering gjennom å lage et enkelt program som simulerer og organiserer temperaturmålinger. Først skal vi opprette en ordbok `målinger` med simulerte temperaturer. Nøklene i ordboken skal være tupler bestående av et stedsnavn og et tidspunkt. Verdien knyttet til et slikt tuppel representerer temperaturen målt på dette stedet til dette tidspunktet.

4.a
(10%) Vi lager først funksjonen `simuler(målinger, sted, tid, t_min, t_max)` som v.h.a. `randint` legger til en tilfeldig generert temperatur mellom `t_min` og `t_max` til nøkkelen `(sted, tid)` i ordboken `målinger`. Her vil `sted` være en streng og `tid` et `datetime` objekt.

```
In [11]: målinger = {}
In [12]: simuler(målinger, 'Bergen', datetime.now(), 3, 14)
In [13]: målinger
Out[13]:{('Bergen',
datetime.datetime(2021, 11, 4, 15, 59, 55, 000000)): 9}
```

4.b
(10%) Lag en funksjon `simuler_mange(målinger, sim_info)` som legger til flere simulerte målinger i ordboken `målinger`. La `sim_info` være en parameter som inneholder simuleringsinformasjon om flere ulike steder. Parameteren `sim_info` skal være en liste av tupler, hvor hver tuppel består av et stedsnavn, laveste temperatur, høyeste temperatur og hvor mange dager bak i tid vi vil simulere temperaturer for.

4.c
(10%) Gitt en ordbok med temperaturmålinger tilsvarende de vi har sett i oppgavene over, lag en funksjon `utskrift(målinger)` som skriver ut informasjon om alle temperaturmålingene fra varmest til kaldest. Målinger med samme temperatur kan du selv velge rekkefølgen på. Et eksempel på hvordan utskriften kan se ut finner du på neste side.

```
In [14]: målinger = {}
In [15]: sim_info = [('Bergen', 3, 14, 3),
    ...: ('Os', 2, 12, 3),
    ...: ('Knarvik', 5, 15, 3),
    ...: ('Voss', 0, 10, 3)]
In [16]: simuler_mange(målinger, sim_info)
In [17]: utskrift(målinger)
Sted:     Dato:                Temperatur:
Os        03.11.2021 15:59:55           11
Knarvik   03.11.2021 15:59:55           10
Bergen    03.11.2021 15:59:55            9
Voss      03.11.2021 15:59:55            9
Os        02.11.2021 15:59:55            8
Bergen    04.11.2021 15:59:55            7
Os        04.11.2021 15:59:55            7
Knarvik   04.11.2021 15:59:55            6
Voss      04.11.2021 15:59:55            6
Bergen    02.11.2021 15:59:55            5
Knarvik   02.11.2021 15:59:55            5
Voss      02.11.2021 15:59:55            1
```

Som et alternativ til å kjøre koden fra konsollen, kan du på slutten av filen `oppg4.py` legge til

```python
if __name__ == '__main__':
    målinger = {}
    sim_info = [('Bergen', 3, 14, 3),
                ('Os', 2, 12, 3),
                ('Knarvik', 5, 15, 3),
                ('Voss', 0, 10, 3)]
    simuler_mange(målinger, sim_info)
    utskrift(målinger)
```