IN1000 Obligatorisk oppgave 7

Spillelister og sanger

Frist for innlevering: 24. oktober kl.23.59 **Sist endret:** 11. okt. 2022

Introduksjon

I denne oppgaven skal du lage to klasser til et program for å finne og spille musikk lagret i et arkiv på din maskin. De ordene som er uthevet med fet skrift i teksten er enten Python nøkkelord eller navn som skal brukes i det ferdige programmet. Det er laget test-programmer for begge klasser (lagt ved *sangtester.py, spillelistetester.py, musikk.txt*), de skal kjøres med klassene dine uten feilmeldinger før du leverer. Du må gjerne utvide test-programmene, men ikke fjerne eller endre noe.

Begge klasser og test-programmer skal leveres. Dessuten bilde/ tegning og en tekstfil som svarer på teorispørsmålene i oppgave 1. Det er viktig at du kommenterer i Devilry hvordan løsningen din fungerer og hva som har vært utfordrende og enkelt. Dersom løsningen din ikke virker for alle testene, skal du beskrive hvilke tester som slår feil og foreslå hvorfor.

Filer som skal leveres: *teorisporsmal.txt, bilde/tegning, sang.py, spilleliste.py, musikk.txt, sangtester.py, spillelistetester.py*

Oppgave 1: Teorispørsmål

Les gjennom hele teksten først slik at du får en oversikt over oppgaven. Oppgave 1 må ikke løses først, men er ment for at man skal vise forståelse for hvordan klassene er bygd opp. Svar på oppgave 1a) leveres som en digital tegning eller f eks et bilde tatt med mobilen (filtype: *.png*, *.jpg* eller *.pdf*), mens oppgave 1b), 1c) og 1d) besvares i en tekstfil (filtype: *.txt*)

a) Lag en tegning av datastrukturen i programmet *spillelistetester* slik den ser ut etter at en spilleliste er lest inn fra filen musikk.txt. Du trenger ikke tegne mer enn to Sang-objekter. Bruk notasjonen fra forelesningene, med variabler som bokser, objekter som sirkler eller bokser med runde hjørner, og referanser som piler fra en variabel til et objekt. Du kan tegne strenger som innhold i enkle variabler, mens en liste tegnes som et objekt.

b) Klassen Spilleliste kunne hatt filnavn som parameter til konstruktøren, og lest inn spillelisten ved opprettelsen av et nytt Spilleliste-objekt. Ser du noen fordel ved ikke å gjøre det i konstruktøren?

c) I klassen Spilleliste er det en instansvariabel som lagrer alle sangene i en liste. Nevn minst en, gjerne to årsaker til at det er naturlig å velge en liste fremfor en ordbok her.

d) Hva må endres i Spilleliste-klassen om rekkefølgen på tittel og artist byttes om i datafilen *musikk.txt* ?

Oppgave 2: Klassen **Sang**

Du skal skrive en klasse **Sang** som en egen modul i filen *sang.py*, med følgende grensesnitt:

Klassen skal ha en *konstruktør* med parametere for artist og tittel (i tillegg til **self**). Konstruktøren skal opprette instansvariabler **_artist** og **_tittel**. Disse er strenger som får verdi fra parameterne. Instansvariabelen **_artist** er en streng som består av en eller flere bokstavsekvenser atskilt av blanke tegn. Eksempler: "Simon and Garfunkel", "The Rolling Stones", "Prince".

Klassen skal dessuten tilby følgende metoder i grensesnittet:

- **spill** "spiller av" musikken i sangen den kalles for – i dette programmet betyr det at den skriver meldingen "Spiller <info om tittel og artist>" ut på terminalen.

  Det ligger også en frivillig utvidelse på obligsiden, med denne utvidelsen kan du få faktiske sanger til å bli spilt av på PC'en din.

- **sjekk_artist** med parameter **navn** (en streng) på samme form som instansvariabelen **_artist**. Metoden returnerer **True** dersom ett eller flere av navnene i strengen **navn** finnes i **_artist**, ellers **False**.

  For artistnavnet "Lady Gaga and Bradley Cooper" så skal *sjekk_artist("Lady")* returnere **True,** *sjekk_artist("Lord Gaga")* returnere **True** (et ord finnes i artistnavnet), mens *sjekk_artist("Sadley")* og *sjekk_artist("a")* skal returnere **False**.

- **sjekk_tittel** med parameter **tittel** (en streng). Metoden sjekker om oppgitt tittel er den samme som i instansvariabelen og returnerer **True** ved likhet, ellers **False**. Titlene skal defineres som like uavhengig av små/ store bokstaver.

- **sjekk_artist_og_tittel** med parametere **artist** og **tittel**. Metoden returnerer **True** dersom både tittelen og artisten (samme regler som i sjekk-metodene) stemmer med sangens instansvariabler, ellers **False**.

Oblig 6 med lenker til Trix-oppgaver gir god trening i det du trenger i oppgave 2, Se også Trix-oppgave 08.05

Frivillig (nyttig!) utvidelse: Implementere __str__ metoder

Metodenavnet __str__ brukes i Python for metoder som returnerer en tekststreng med «menneskevennlig» presentasjon av innholdet i et objekt – for eksempel *«"Money" av Pink Floyd».* Hvis du har en **__str__** metode i klassen Sang, blir den kalt automatisk hver gang du gjør **print(en_sang)** eller **str(en_sang)**, der **en_sang** er en variabel som refererer til et **Sang**-objekt.

Skriv **__str__** metoder for klassene Sang og Spilleliste (når du løser oppgave 3), og endre klassene slik at det er disse som kalles hver gang du trenger innholdet i et objekt som en streng. Oppdater (og levér) testprogrammene slik at de utnytter at denne metoden finnes i klassene.

Oppgave 3: Klassen **Spilleliste**

I filen *spilleliste.py* finner du definisjon av klassen **Spilleliste**. Konstruktøren er ferdig skrevet, mens metodene beskrevet nedenfor skal ferdigstilles av deg (se rammen med **Frivillig utvidelse** ovenfor for utvidelser). Det forventes at du bruker metodene til Sang objektet der det er mulig.

Skriv metoden **les_fil** med parametere **self** og **filnavn**. Metoden åpner den oppgitte filen, og leser inn data om sanger – en linje per sang, på formatet

```
tittel;artist
```

Siden både tittel og artist kan inneholde blanke tegn, brukes semikolon som skilletegn. Merk også at det er lurt å fjerne tegn for linjeskift på slutten av hver linje. En linje kan for eksempel leses slik:

```
alle_data = linje.strip().split(';')
```

Metoden skal opprette nye **Sang**-objekter etter hvert som filen leses, og legge disse inn i spillelisten.

Klassen skal dessuten tilby følgende metoder i grensesnittet:

- **legg_til_sang** med parametere **self** og **ny_sang**, der **ny_sang** er objektet som skal legges til spillelisten.
- **fjern_sang** med parametere **self** og **sang**. Metoden tar inn et Sang-objekt som skal fjernes fra listen med sanger.
- **spill_sang** med parametere **self** og **sang**. Metoden skal kalle på metoden spill() til Sang-objektet som sendes med som parameter.
- **spill_alle** med parameter **self**, som spiller hver enkelt sang i listen. Pass på å sjekke at rekkefølgen på *tittel;artist* i **les_fra_fil** har blir lest inn riktig ved å se hva som printes i terminalen når metoden blir kalt på.
- **finn_sang** med parametere **self** og **tittel**, som leter gjennom listen av sanger etter en med oppgitt tittel og returnerer den første den finner. Finnes ikke tittelen i spillelisten returneres **None**.
- **hent_artist_utvalg** med parametere **self** og **artistnavn**, som går gjennom alle sanger i spillelisten og returnerer en liste med sanger der artisten har et eller flere navn fra parameteren **artistnavn**.

I filene *sangtester.py* og *spillelistetester.py* ligger det testprogrammer for de to klassene, og i filen *musikk.txt* ligger det data for innlesing til programmet. De vil være nyttige under utvikling av klassene, og brukes av gruppelærer for vurdering av innleveringen din.

Synes du oppgave 3 var vanskelig? Prøv Trix-oppgavene 09.03 og 09.09