Spillelister og sanger

Obligatorisk oppgave 7, IN1000 (2019)

Frist for innlevering: 23.10. kl 23:59.
(Lever gjerne før, skriv «Klar for retting» i Devilry)

I denne oppgaven skal du lage et program for å finne og spille musikk lagret i et arkiv på din maskin. De ordene som er uthevet med bold i teksten er enten Python nøkkelord eller navn som skal brukes i det ferdige programmet. Det er laget test-programmer for begge klasser, de skal brukes før du leverer. Du må gjerne utvide test-programmene med flere tester og utskrifter, men de som er der skal ikke fjernes.

Begge klasser og test-programmer skal leveres. Dessuten bilde/ scannet figur og en tekst-fil som svarer på teorispørsmålene i oppgave 3. 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.

Oppgave 1 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 tittel og artist (i tillegg til self). Konstruktøren skal opprette instansvariable **_tittel** og **_artist**. 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.

sjekkArtist 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**.

sjekkTittel 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.

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

Frivillig 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 skriver en metode med dette navnet i klassen din, blir den kalt automatisk hver gang du gjør print(referanse til et objekt) eller str(referanse til et objekt) på et objekt av den klassen.

Skriv __str__ metoder for klassene Sang (og Spilleliste når du løser oppgave 2), og endre klassene slik at det er disse som kalles hver gang du trenger innholdet i et objekt som en streng. Foreslå gjerne også mulige endringer i testprogrammene når denne metoden finnes i klassene.

Oppgave 2 Klassen Spilleliste

I filen *spilleliste.py* finner du definisjon av klassen **Spilleliste**. Konstruktøren er ferdig skrevet, mens følgende metoder skal ferdigstilles av deg:

lesFraFil 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 kan være lurt å fjerne tegn for linjeskift på slutten av hver linje. Lesing av en linje kan da for eksempel gjøres slik:

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

Metoden skal opprette nye **Sang**-objekter etter hvert som filen leses, og legge disse inn i spillelisten. Filen lukkes til slutt. Se punktet **Frivillig utvidelse** nedenfor for mulige utvidelser.

leggTilSang med parametere **self** og **nySang**, der **nySang** er det nye objektet som skal legges til spillelisten.

fjernSang med parametere **self** og **sang**.

spillSang med parametere **self** og **sang**.

spillAlle med parameter **self**, som spiller hver enkelt sang i listen.

finnSang 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.

hentArtistUtvalg med parametere **self** og **artistnavn**. Metoden går gjennom alle sanger i spillelisten og tar vare på de som har en eller flere navn fra parameteren **artistnavn** i navnet på artisten. Disse returneres i en liste med sanger.

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

Oppgave 3 Teorispørsmål

Svar på oppgave a) leveres som scannet dokument (evt et bilde tatt med mobilen) mens oppgave b) og c) besvares i en tekstfil.

a) Lag en tegning, gjerne på papir, av datastrukturen i 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 (selv om de egentlig er objekter), mens en liste tegnes som et objekt som refereres til fra en listevariabel.

b) I klassen Spilleliste er det en instansvariabel som lagrer alle sangene i en liste. Nevn minst en, helst to årsaker til at en ordbok ikke egner seg like godt i dette tilfellet.

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