Sist endret: 11. oktober 2023, kl.11.50

IN1000 Obligatorisk oppgave 7

Spillelister og sanger

Frist for innlevering: 23. oktober kl.23.59

Introduksjon

I denne oppgaven skal du lage to klasser og et program som bruker disse til å finne, organisere 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.

Sammen med denne oppgaveteksten finner du en datafil (*musikk.txt)*, et testprogram for Sang-klassen (*test_sang.py*), en start på Spilleliste-klassen (*spilleliste.py*) og et testprogram for Spilleliste-klassen (*test_spilleliste.py*). Disse trenger du under utvikling av klassene, og brukes av gruppelærer for vurdering av innleveringen din.

Hvilke filer som skal leveres for oblig 7 er oppgitt for hver enkelt oppgave og oppsummert nederst i dette dokumentet.

For å få godkjent denne obligatoriske oppgaven må du ha løst oppgave 1-3 (testprogrammer for Sang og Spilleliste må fungere). Om du ikke helt har fått til oppgave 4 må du beskrive i kommentar i Devilry hva som var vanskelig.

Oppgave 1: Teorispørsmål

Leveres:

1. 1a) leveres i bildefil (*teori1a.png, teori1a.jpg eller teori1a.pdf*). Digital tegning eller håndtegning fotografert med mobil.
2. 1b-d) besvares i én fil, *teori.txt*

Prøv deg gjerne på teorioppgavene før du har løst resten av oppgaven (men etter å ha *lest hele* obligteksten!) – det vil gjøre programmeringen enklere. Når du har programmert oppgave 2-4 kan du gå tilbake og eventuelt forbedre svarene dine.

Tegning av datastrukturer gir oss bedre forståelse av hva som skjer under utføring av et program, og er nyttig for å huske i detalj hvilke variabler, objekter og referanser vi opererer på. Dette er spesielt nyttig for å bygge opp en mental modell i starten av faget – men også når man skal arbeide med komplekse datastrukturer senere.

a) Lag en tegning av datastrukturen etter at følgende kode er utført:

```
def hovedprogram():
    ny_spilleliste = Spilleliste('musikk')
    ny_spilleliste.les_fra_fil()
```

Du trenger ikke tegne mer enn to Sang-objekter. Bruk notasjonen fra forelesningen i uke 8, 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 lest inn filen med sangene (spillelisten) i konstruktøren, dvs. 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 én, 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**

Leveres:

- Filen *sang.py* med klassen **Sang**

Du skal skrive en klasse **Sang** som en egen modul i filen *sang.py.* Tabellen nedenfor angir klassens grensesnitt. Viktig: Ikke gå videre til neste oppgave før test_sang.py kjører uten feil med din Sang-klasse, og legg gjerne til egne tester!

Metode	**Parameter (Type)**	**Returverdi**	**Kommentar**
__init__	tittel (streng) artist (streng)	-	Konstruktør som lagrer sangens tittel og artist. Tittel og artist er strenger som består av en eller flere bokstavsekvenser atskilt av blanke tegn. Eksempler: "Simon and Garfunkel", "The Rolling Stones", "Prince".
spill	-	-	(ingen parametere annet enn **self**). Metoden "spiller av" musikken i sangen den kalles på – i dette programmet betyr det at den skriver meldingen "Nå spilles <tittel > med <artist>" ut til terminalen.
sjekk_artist	navn (streng)	True/False	Returnerer **True** hvis ett eller flere ord (delstrenger atskilt av blanke/ space) i parameteren navn finnes i sangens artist. For en sang med artist “Lady Gaga and Bradley Cooper” så skal *sjekk_artist(“Lady”)* returnere **True,**

			sjekk_artist("Lord Gaga") returnere **True** (minst et ord finnes i artistnavnet), mens *sjekk_artist("Sadley")* og *sjekk_artist("a")* skal returnere **False**. ("a" er ikke et separat ord). Ordene skal defineres som like uavhengig av små/store bokstaver.
sjekk_tittel	tittel (streng)	True/False	Metoden sjekker om oppgitt tittel er lik sangens tittel og returnerer **True** ved likhet, ellers **False**. Titlene skal defineres som like uavhengig av små/ store bokstaver.
sjekk_artist_og_tittel	artist (streng) tittel (streng)	True/False	Metoden returnerer **True** dersom både tittelen og artisten (samme regler som i sjekk-metodene ovenfor) stemmer, ellers **False**.
streng_til_fil	-	streng	(ingen parametere annet enn self). Returnerer en streng med objektets innhold på samme form som beskrevet i metoden les_fra_fil i Spilleliste, avsluttet med linjeskift. Eks: "Help;The Beatles\n"
__str__ (frivillig, gjennomgås i uke 9)	-	streng	(ingen parametere annet enn self). Returnerer en streng med innholdet i Sang-objektet på følgende format (eksempel): "Help med The Beatles"

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

Oppgave 3: Klassen **Spilleliste**

Leveres:

- Filen *spilleliste.py* med klassen **Spilleliste**

I filen *spilleliste.py* finner du definisjon av klassen **Spilleliste**. Konstruktøren er ferdig skrevet, mens resten av metodene i grensesnittet definert nedenfor skal implementeres av deg. I klassen Spilleliste skal du kalle på metodene i Sang-klassen, og ikke aksessere instansvariablene i Sang direkte. Viktig: Ikke gå videre til neste oppgave før test_spilleliste.py kjører uten feil med din Spilleliste-klasse, og legg gjerne til egne tester!

Klassen **Spilleliste** skal i tillegg til konstruktør tilby følgende metoder i grensesnittet:

Metode	Parameter (Type)	Retur-verdi	Kommentar
les_fra_fil	-	-	(ingen parametere annet enn **self**). Metoden åpner en fil med spillelistens navn (instansvariabel) fulgt av ".txt". Filen inneholder 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 et nytt **Sang**-objekt for hver linje, og legge disse inn i spillelisten.
legg_til_sang	ny_sang (Sang)	-	ny_sang er Sang-objektet som skal legges til spillelisten
fjern_sang	sang (Sang)	-	Metoden tar inn et Sang-objekt som skal fjernes fra listen med sanger.
spill_alle	-	-	(ingen parametere annet enn **self**). Metoden spiller hver enkelt sang i listen.
finn_sang_tittel	tittel (streng)	Sang/ None	Leter gjennom listen av sanger etter et Sang-objekt med oppgitt tittel og returnerer den første den finner. Finnes ikke tittelen i spillelisten returneres **None**.
hent_artist_ utvalg	artistnavn (streng)	list	Går gjennom alle sanger i spillelisten og returnerer en liste med sang-objekter der artisten har et eller flere navn fra parameteren **artistnavn**.
skriv_til_fil	-	-	(ingen parametere annet enn **self**). Åpner en fil med navnet til spillelisten fulgt av ".txt" (for eksempel *musikk.txt* hvis spillelisten har navn musikk). Om filen finnes fra før er det greit å overskrive innholdet. Hver sang i spillelisten skal skrives ut på en linje. Husk å lukke filen til slutt.
__str__ *(frivillig, gjennomgås i uke 9)*	-	streng	(ingen parametere annet enn **self**). Returnerer en brukervennlig streng med navnet på spillelisten etterfulgt av informasjon om alle sangene i listen (en på hver linje). Tips: "\n" gir linjeskift i en streng.

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

Sist endret: 11. oktober 2023, kl.11.50

Oppgave 4: Et musikkarkiv

Leveres:

- Filen *musikkarkiv.py* med en prosedyre **hovedprogram** og et kall på denne.
- Alle filene du har lagret spillelistene på i punkt 6 nedenfor.

Du skal nå skrive et program som lar en bruker organisere et enkelt musikkarkiv, implementert ved hjelp av klassene **Sang** og **Spilleliste**. Instansvariabler i disse klassene skal ikke aksesseres direkte fra **hovedprogram** - du skal kun bruke metodene i grensesnittene deres.

For å få bedre oversikt over hvordan du kan løse oppgave 4 vil det lønne seg å utvide tegningen din fra oppgave 1a) med ordboken og de tre spillelistene beskrevet nedenfor (du trenger ikke vise sang-objekter i de nye spillelistene). Denne tegningen trenger du ikke levere inn.

Prosedyren **hovedprogram** skal gjøre følgende:

1. Definer en variabel **mine_spillelister** og opprette en tom ordbok for denne.
2. Opprette et **Spilleliste**-objekt med navn "musikk" og deretter lese inn sangene fra filen *musikk.txt* i dette objektet ved hjelp av metoden **les_fra_fil.** Legg denne spillelisten inn i ordboken **mine_spillelister** med navnet på spillelisten som nøkkel og referansen til **Spilleliste**-objektet som verdi**.**
3. Opprett et **Spilleliste**-objekt med spilleliste-navn "queen". Hent ut alle sanger med "Queen" i artistnavnet fra spillelisten med navn musikk, og legg disse sangene en og en inn i den nye spillelisten queen. Legg deretter denne spillelisten også inn i **mine_spillelister**.
4. Opprett nok et **Spilleliste**-objekt med et navn du velger selv. Deretter, opprett 3 **Sang**-objekter der du selv velger tittel og artist, og legg disse til den nye spillelisten. Legg deretter denne spillelisten også inn i **mine_spillelister**.
5. Slå opp den spillelisten du selv valgte navnet på i **mine_spillelister,** og spill av alle sangene i denne spillelisten.
6. Skriv alle spillelistene du har opprettet ut på hver sin fil. Filnavnet skal være navnet på spillelisten etterfulgt av ".txt" (for eksempel *musikk.txt* for spillelisten musikk).

Følgende filer skal leveres for oblig 7:

- *teori1a.png (eller annet format)*
- *teori.txt*
- *sang.py*
- *test_sang.py*
- *spilleliste.py*
- *test_spilleliste.py*

- *musikkarkiv.py*
- *musikk.txt og andre filer du skriver ut i oppgave 4*