Programmering 1 – ITF10219 – Sensorveiledning

Bemerk at dette dokumentet er ment som en veiledning ikke som en fasit. Hvis sensor er uenig med enkelte av poengutdelingene eller støter på situasjoner hvor veiledningen ikke er dekkende nok kan egen bedømmelse benyttes.

Oppgave 3.1

Oppgaven kan gi inntil 10 poeng distribuert basert på følgende aspekter:

- Benytter operatorer på en god måte. // og % er preferabelt – 5 poeng
  - Hvis andre operatorer blir benyttet, men fremdeles fører til riktige utregninger – 3 poeng
  - Hvis operetorer blir brukt noenlunde riktig, men ikke fører til riktig utregning – 1 til 2 poeng
- "Resultatet" blir en tuple på riktig format – 3 poeng
  - Hvis enklere løsning (format) blir benyttet, for eksempel Minuttene for seg selv (Merk at dette er separat fra selve utregningen). - 1 poeng
- Return blir brukt riktig – 2 poeng

Løsningsforslag:

```
1    def convert_seconds_to_minutes(seconds):
2        minutes = seconds // 60
3        #seconds_left = seconds - minutes*60
4        seconds_left = seconds % 60
5        return (minutes, seconds_left)
```

Oppgave 3.2

Oppgaven kan gi inntil 10 poeng distribuert basert på følgende aspekter:

- Studenten printer utskriften på beskrevet format – 1 poeng
- Studenten klarer å hente ut elementene fra listen mottatt i song-parameteren på riktig måte - 4 poeng
  - Hvis feil indexer men ellers riktig – bare 2 poeng
- Studenten benytter convert_seconds_to_minutes()-funksjonen for å konverere sekundene til minutter – 3 poeng
  - Hvis enklere løsning blir benyttet (Bare skrive ut sekundene direkte) - 0 poeng.
- Studenten henter ut minutter og gjenværende sekunder fra tuple-en returnert fra convert_seconds_to_minutes()-kallet – 2

Løsningsforslag:

```python
def print_song_description(song):
    print(f"Title: {song[0]}")
    print(f"Artist: {song[1]}")
    minutes = convert_seconds_to_minutes(song[2])
    print(f"Length: {minutes[0]} min, {minutes[1]} seconds")
    # alternativt:
    # print(f"Length in Seconds: {song[2]}")
```

Oppgave 3.3

Oppgaven kan gi inntil 10 poeng distribuert basert på følgende aspekter:

- Benytter if, elif og else på en god måte– 5 poeng distribuert basert på det følgende
  - If, elif og else benyttes – 3 poeng
    - Hvis bare if benyttes, eller lignende – 1 poeng
  - Riktige utskrifter/resultater basert på if-testene – 2 poeng.
- Benytter sammenligningsoperatorene på en god måte, spesielt duration <= 0 – 3 poeng
  - Hvis operatoren er feil, men nesten riktig – 1 til 2 poeng
- Returnerer en dictionary på riktig format – 2 poeng

Løsningsforslag:

```python
def create_song(title, artist, duration, album_title):
    if duration <= 0:
        print("Duration must be larger than 0 seconds.")
    elif artist == "Dingus":
        print("Dingus is banned from making songs.")
    else:
        return {"title": title, "artist": artist,
                "duration": duration, "album_title": album_title}
```

Oppgave 3.4

Oppgaven kan gi inntil 10 poeng distribuert basert på følgende aspekter:

- For-løkke som går gjennom sang-dictionariene i playlist-parameteren (liste med dictionaries) - 3 poeng
- Referere til dictionary-elementer på riktig måte (song[“artist”] eller lignende) - 2 poeng
- If-tester som riktig oppdaterer den nye dictionarien (her artist_count) - 3
  - Hvis ikke “nye” artister tas hensyn til (Se else) - 1 til 2 poeng
- Bruke in-operatoren i if-test for å unngå en indre for-løkke (Altså smidig kode) - 2 poeng
  - Samme uttelling hvis man på en annen smidig måte klarer å sjekke om en artist allerede finnes i “telleren”.

Løsningsforslag:

```python
def count_songs_by_artists(playlist):
    artist_count = {}
    for song in playlist:
        song_artist = song["artist"]
        if song_artist in artist_count:
            artist_count[song_artist] += 1
        else:
            artist_count[song_artist] = 1
    return artist_count
```

Oppgave 3.5

Oppgaven kan gi inntil 10 poeng distribuert basert på følgende aspekter:

- Bruker en løkke som forsørger at den nye listen (shuffled_playlist) blir fylt opp (Vil nok generelt måtte være basert på listelengde av den orginale eller den nye listen) - 3 poeng
- Velger tilfeldige sanger på en god måte - 2 poeng
    - Ikke importert random – 1 poeng
- Fyller opp den nye listen med tilfeldige sanger – 2 poeng
- Har skrevet logikk som unngår duplikater – 3 poeng

Løsningsforslag (2 alternativer – Begge anses som gyldige i konteksten av kurset):

```python
import random


def get_shuffled_playlist(playlist):
    shuffled_playlist = []
    while len(shuffled_playlist) < len(playlist):
        random_song = random.choice(playlist)
        if random_song not in shuffled_playlist:
            shuffled_playlist.append(random_song)
    return shuffled_playlist


def get_shuffled_playlist_alt2(playlist):
    shuffled_playlist = []
    song_sequence_copy = playlist[:]
    for x in range(len(playlist)):
        random_index = random.randrange(len(song_sequence_copy))
        random_song = song_sequence_copy.pop(random_index)
        shuffled_playlist.append(random_song)
    return shuffled_playlist
```

Oppgave 3.6

Oppgaven kan gi inntil 10 poeng distribuert basert på følgende aspekter:

- Åpner filen med with-blokk – 3 poeng
- Laster albumene med json.load() - 1 poeng
- Benytter try/except-blokk og håndterer feil på en god måte– 3 poeng
- Har logikk for å hente ut album på en god måte (if/else-blokkene) - Inntil 3 poeng avhengig av aspektene under:
    - Hvis logikken ikke er inneholdt i en else-blokk koblet til try/except - 2 poeng.

Løsningsforslag:

```
def get_album_from_json_file(album_title, filename):
    try:
        with open(filename) as f:
            albums = json.load(f)
    except FileNotFoundError:
        print("Could not open file.")
    else:
        if album_title in albums:
            return albums[album_title]
        else:
            print("Album does not exist.")
```