Allmän information

Klistra in din lösning i textfältet under beskrivningen för varje fråga.

För en tenta med 2 frågor: Betyg U: ≤ 13 poäng. Betyg 3: 14–20 p. Betyg 4: 21–27 p. Betyg 5: 28–35 p.

För en tenta med 3 frågor: Betyg U: ≤ 23 poäng. Betyg 3: 24–35 p. Betyg 4: 36–47 p. Betyg 5: 48–60 p.

Språk: Frågorna ges på engelska och svenska. Du kan använda "hamburger"-menyn (ser ut som ≡) för att välja språk på frågorna. Vid eventuella avvikelser beror det på kurskod vilket språk gäller:

Kurskod	Språk
TDA548, DAT555, DAT645, DAT685, DAT395	svenska
DAT506, DAT505, DAT425, DAT446, DAT445, DAT456, DAT457	English

Tillåtna hjälpmedel:

- Ett (dubbelsidigt) papper med handskrivna anteckningar
- Self-Practice-webbplatsen i sandbox-läge för att testa din kod. Från Safe Exam Browser klickar du på knappen "Python Sandbox" längst ned till vänster på skärmen för att öppna den. Detta öppnar self-practice-webbplatsen som användes under kursen i ett nytt fönster.
  - Observera att din kod inte sparas om du stänger eller laddar om fönstret. Du kan lämna Self-Practice-fönstret öppet och växla mellan fönster med Alt+Tab eller Cmd+Tab.
- Resursen "Python reference" på Inspera.

Otillåtna hjälpmedel: allt annat. I synnerhet är det inte tillåtet att använda följande verktyg:

- AI-verktyg (Copilot, ChatGPT, Claude, …)
- Kommunikationsverktyg (telefon, chatt, e-post, sociala medier, …)
- Elektronisk dokumentation (Google, Stack Overflow, Python-guider eller manualer, …)
- Böcker (förutom ordböcker)
- Anteckningar i tryckt eller elektroniskt format

Resultat och tentamensgranskning

Så fort rättningen är klar (senast en vecka före granskning) kan du se din rättade kopia på Inspera. Standardlösningar för tentan finns på den vanliga webbsidan: https://git.chalmers.se/courses/programming-intro-courses/python-courses-exams

Om du anser att ditt svar har rättats fel kan du komma till tentagranskningen, som äger rum:

- Datum: 2 april
- Tid: 13.00–14.00
- Plats: EDIT Analysen

Förbered inför granskningen:

1. Den specifika fråga eller delfråga som du anser har rättats felaktigt.
2. Förklaring till varför ditt svar faktiskt är korrekt.
3. Din anonyma kod som användes på Inspera.

Obs: Syftet med tentamensgranskningen är inte att diskutera betygsskalan eller tentans svårighetsgrad, utan endast att åtgärda eventuella fel i rättningen.

Q1: Pythagoreisk skala

En serie toner bildar en *skala*. För att spela behaglig musik måste skalan vara *välstämd*, dvs. tonerna inom skalan ska ha matematiskt enkla frekvensförhållanden till varandra.

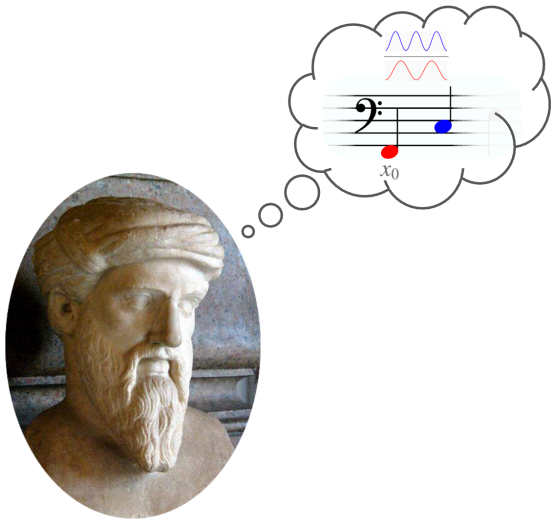
En **pythagoreisk skala** byggs på förhållandet 3:2. Vi kallar skalans grundton för x_0. Då beräknas skalans ton x_{n+1} på följande sätt:

$$x_{n+1} = \begin{cases} \frac{\frac{3}{2}x_n}{2} & \text{om } \frac{3}{2}x_n > 2 \cdot x_0 \\ \frac{3}{2}x_n & \text{annars} \end{cases}$$

Uppgift: Skapa en funktion `pythagorean_scale(x0, scale_length)` . Den tar som argument grundtonens frekvens och skalans längd, och returnerar en skala som en lista av frekvenser.

Exempel:

```
>>> for tone in sorted(pythagorean_scale(1, 7)):
...     print(tone)
...
1
1.125
1.265625
1.423828125
1.5
1.6875
1.8984375
```



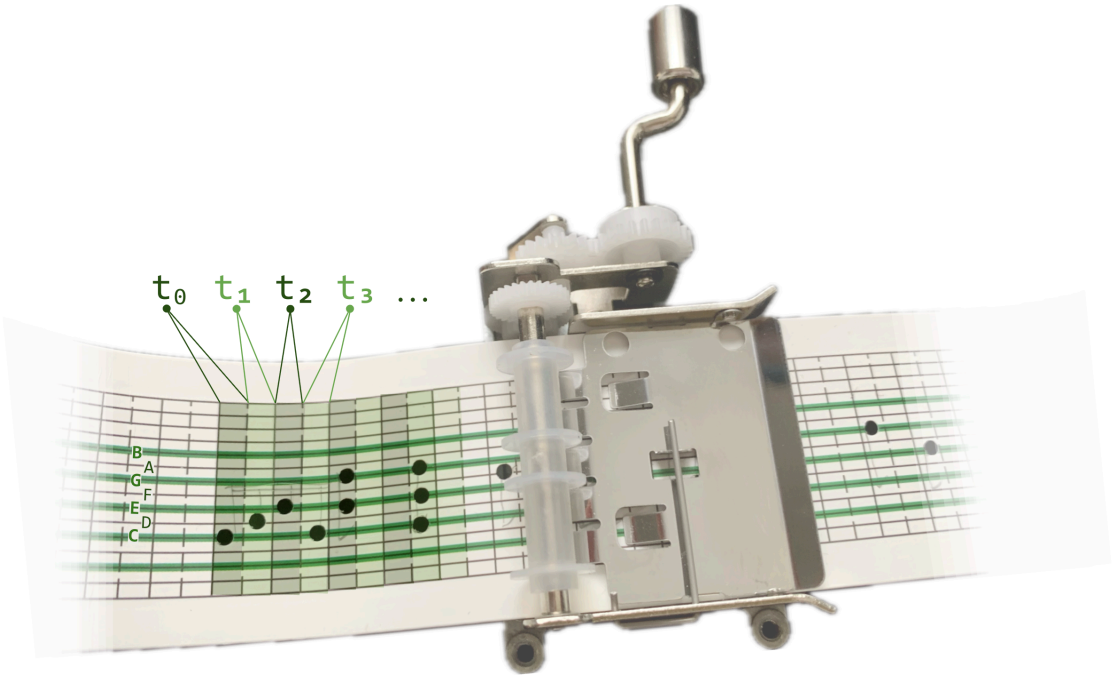
Din funktion kan såklart också sortera skalan, men detta krävs inte.

Facit

```
def pythagorean_scale(x0, scale_length):
    f = x0
    scale = []
    for _ in range(scale_length):
        scale.append(f)
        f = f * 3/2
        if f > 2*x0:
            f = f / 2

    scale.sort()    # Optional
    return scale
```

Q2: Speldosa

En speldosa spelar musik ur en stansad pappersremsa där varje hål motsvarar en not. När remsan matas fram så spelas en melodi.



Kolumnerna visar tidsenheter och raderna noter: t.ex. under t_0 spelas ett C, under t_4 spelas E och G samtidigt.

En textrepresentation av remsan finns i Sandbox 2 med filnamnet `musicstrip.txt`, och är även inklistrad nedan. Ett hål betecknas av 1 och inget hål (dvs. tystnad) av 0:

```
B 0 0 0 0 0 0 0 0
A 0 0 0 0 0 0 0 0
G 0 0 0 0 1 0 0 1
F 0 0 0 0 0 0 0 0
E 0 0 1 0 1 0 0 1
D 0 1 0 0 0 0 0 0
C 1 0 0 1 0 0 0 1
```

Du ska läsa in en musikremsa i ovanstående format och manipulera datan så att du till slut kan "spela" låten.

Uppgift: Skapa en funktion `read_strip(filename)` (10 poäng). Den läser filen och returnerar innehållet som en dictionary där:

- nyckeln är en *not* som sträng ('C', 'D', 'E', 'F', 'G', 'A', 'B')
- värdet är ett *hålmönster* som tillhör noten, som en lista av 1 och 0.

Exempel:

```
>>> read_strip("musicstrip.txt")
{'B': [0, 0, 0, 0, 0, 0, 0, 0], 'A': [0, 0, 0, 0, 0, 0, 0, 0],
 'G': [0, 0, 0, 0, 1, 0, 0, 1], 'F': [0, 0, 0, 0, 0, 0, 0, 0],
 'E': [0, 0, 1, 0, 1, 0, 0, 1], 'D': [0, 1, 0, 0, 0, 0, 0, 0],
 'C': [1, 0, 0, 1, 0, 0, 0, 1]}
```

Uppgift: Skapa en funktion `play_music(score)` (10 poäng). Den ska:

- ta som argument en dictionary med samma innehåll som `read_strip()` returnerar
- returnera en lista av listor `chords`, så att `chords[i]` är det ackordet som spelas under tidsenhet i.

Exempel:

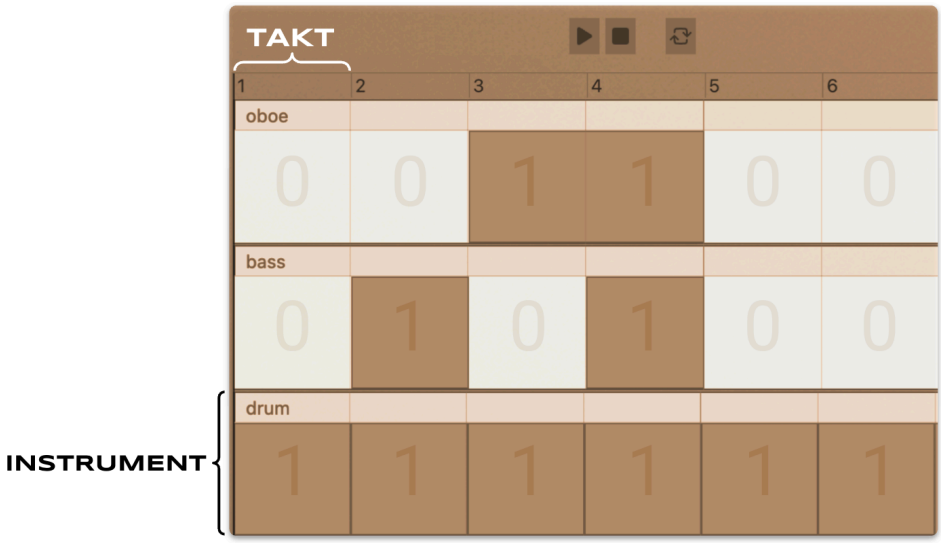
```
>>> for chord in play_music(read_strip("musicstrip.txt")):
...     print(chord)
...
['C']
['D']
['E']
['C']
['G', 'E']
[]
[]
['G', 'E', 'C']
```

Facit

```
def read_strip(filename):
  patterns = {}
  with open(filename) as f:
    for line in f:
      fields = line.split()
      note = fields[0]
      pattern_raw = fields[1:]

      pattern = []
      for hole in pattern_raw:
        pattern.append(int(hole))

      patterns[note] = pattern
  return patterns

def play_music(score):
    l = len(score["A"])
    chords = []
    for i in range(l):
        chord = []
        for k in score:
            if score[k][i]:
                chord.append(k)
        chords.append(chord)
    return chords
```

Q3: Digital Audio Workstation

Speldosor är kul men moderna musiker använder en Digital Audio Workstation (DAW).



(Denna DAW är dock lite primitiv och bara visar *om* ett instrument spelar, inte *vad* det spelar.)

Uppgift: Skapa en klass `DAW` med följande metoder:

Metod: `__init__(self, available_instruments)` (0 poäng)
Initialisera en DAW med instrumenten `available_instruments`. Välj lämpliga attribut så att resten av metoderna har all information som behövs.

Denna metod är obligatorisk för att resten av klassen ska fungera, men den ger inga poäng.

Metod: `keypress(self, instrument)` (5 poäng)

Denna metod anropas när ett instrument börjar spela.

- Om instrumentet inte finns ska metoden avbryta och skriva ut "Error: invalid instrument".
- Om instrumentet redan spelar ska metoden skriva ut "Warning: already pressed".

Observera att datan inte sparas direkt, utan vid nästa anrop till metoden `next_bar` (se nedan).

Metod: `release(self, instrument)` (5 poäng)

Denna metod anropas när ett instrument slutar spela.

- Om instrumentet inte finns ska metoden avbryta och skriva ut "Error: invalid instrument".
- Om instrumentet redan är tyst ska metoden skriva ut "Warning: already released".

Metod: `next_bar(self)` (5 poäng)

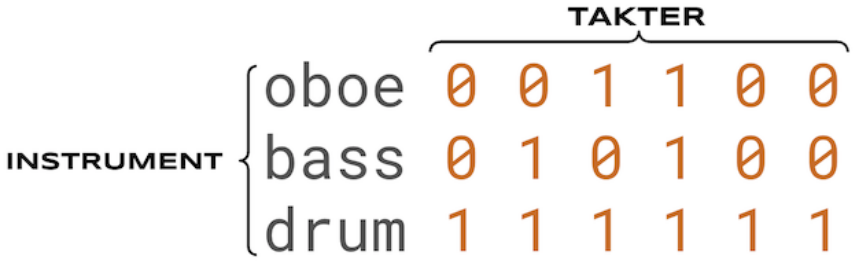
En ny takt börjar med samma tillstånd: spelande instrument fortsätter spela, tysta stannar tysta.

Metod: `previous_bar(self)` (5 poäng)

Gå tillbaka till takten innan (dvs. återskapa tillståndet innan senaste anrop till `next_bar`).

Metod: `print_score(self)` (5 poäng)

Skriv ut låten. Formatet är lik speldosans musikremsa, bara med godtyckliga instrument. Varje rad börjar med instruments namn, i samma ordning som initialiserats.



I utskriftet ska det ingå alla takter upp till senaste anrop av `next_bar()`.

Exempel: Denna kod skapar låten som syns på bilden och även i exemplet ovan.

```
>>> daw = DAW(["oboe", "bass", "drum"])
>>> daw.keypress("drum")
>>> daw.keypress("drum")
Warning: already pressed
>>> daw.keypress("piano")
Error: invalid instrument
>>> daw.keypress("bass")
>>> daw.release("bass")     # OBS. inte med i utskriftet
>>> daw.next_bar()          # 1: drum
>>> daw.print_score()
oboe 0
bass 0
drum 1
>>> daw.keypress("bass")
>>> daw.next_bar()          # 2: drum & bass
>>> daw.release("bass")
>>> daw.release("bass")
Warning: already released
>>> daw.keypress("oboe")
>>> daw.next_bar()          # 3: drum & oboe
>>> daw.keypress("bass")
>>> daw.next_bar()          # 4: drum, bass & oboe
>>> daw.release("oboe")
>>> daw.release("bass")
>>> daw.next_bar()          # 5: drum
>>> daw.next_bar()          # 6: drum
>>> daw.keypress("oboe")    # OBS. inte med i utskriftet
>>> daw.print_score()
oboe 0 0 1 1 0 0
bass 0 1 0 1 0 0
drum 1 1 1 1 1 1
```

Allmänna tips

När man kopierar en variabel vars typ är muterbar, per default är det bara referens som kopieras, inte värdet.

Facit

```
class DAW:
    def __init__(self, available_instruments):
        self.record = []
        self.current_bar = {}
        self.instruments = available_instruments

    def next_bar(self):
        self.record.append(self.current_bar.copy())

    def keypress(self,key):
        if key not in self.instruments:
            print("Error: invalid instrument")
            return
        if self.current_bar.get(key,0) == 1:
            print("Warning: already pressed", key)
        self.current_bar[key] = 1

    def release(self,key):
        if key not in self.instruments:
            print("Error: invalid instrument")
            return
        if self.current_bar.get(key,0) == 0:
            print("Warning: already released", key)
        self.current_bar[key] = 0

    def print_score(self):
        for instr in self.instruments:
            print(instr, end=' ')
            for bar in self.record:
                print(bar.get(instr, 0), end=' ')
            print()

    def previous_bar(self):
        self.current_bar = self.record.pop()
```