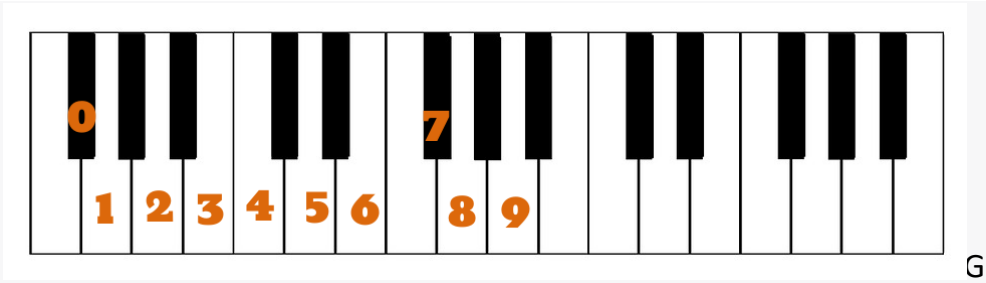
Fibonacci Composer

La sequenza di Fibonacci è una serie di numeri in cui ogni numero successivo è la somma dei due numeri precedenti. I numeri della sequenza di Fibonacci hanno straordinarie proprietà matematiche, e possono essere utilizzati anche nella composizione musicale per creare melodie armoniose e ritmi interessanti.

Utilizzando i numeri della sequenza di Fibonacci, associandoli alle note di una scala maggiore, saremo in grado di creare melodie uniche e interessanti.

Nello specifico ho assegnato i numeri da 1 a 8 alle note di una scala maggiore, che viene scelta dall'utente al momento della composizione, ed ho aggiunto lo 0 e il 9 su ogni lato seguendo la naturale estensione della scala.



G maj

Viene scelta poi la lunghezza della sequenza. Ogni numero con più di 2 cifre, viene scomposto in modo da avere una mappatura delle note sui singoli numeri

La melodia creata viene poi modificata utilizzando un processo additivo in modo tale che verranno usate note in ordine inverso. Si utilizzano diversi pattern di durate in modo da creare una diversa isoritmia,
Per finire si usa il concetto di simmetria musicale per avere la melodia al contrario. (Retrogrado)

L'armonia, per accompagnare la melodia, viene creata aggiungendo tanti accordi della scala di riferimento, quanti sono gli elementi all'interno delle battute della melodia principale.

Il progetto è stato sviluppato in Python 3 attraverso l'utilizzo le librerie music21, arvo.
Per la parte grafica sono state utilizzate le librerie tkinter e customtkinter.
Per la rappresentazione della composizione su piano cartesiano è stata utilizzata la libreria matplotlib.

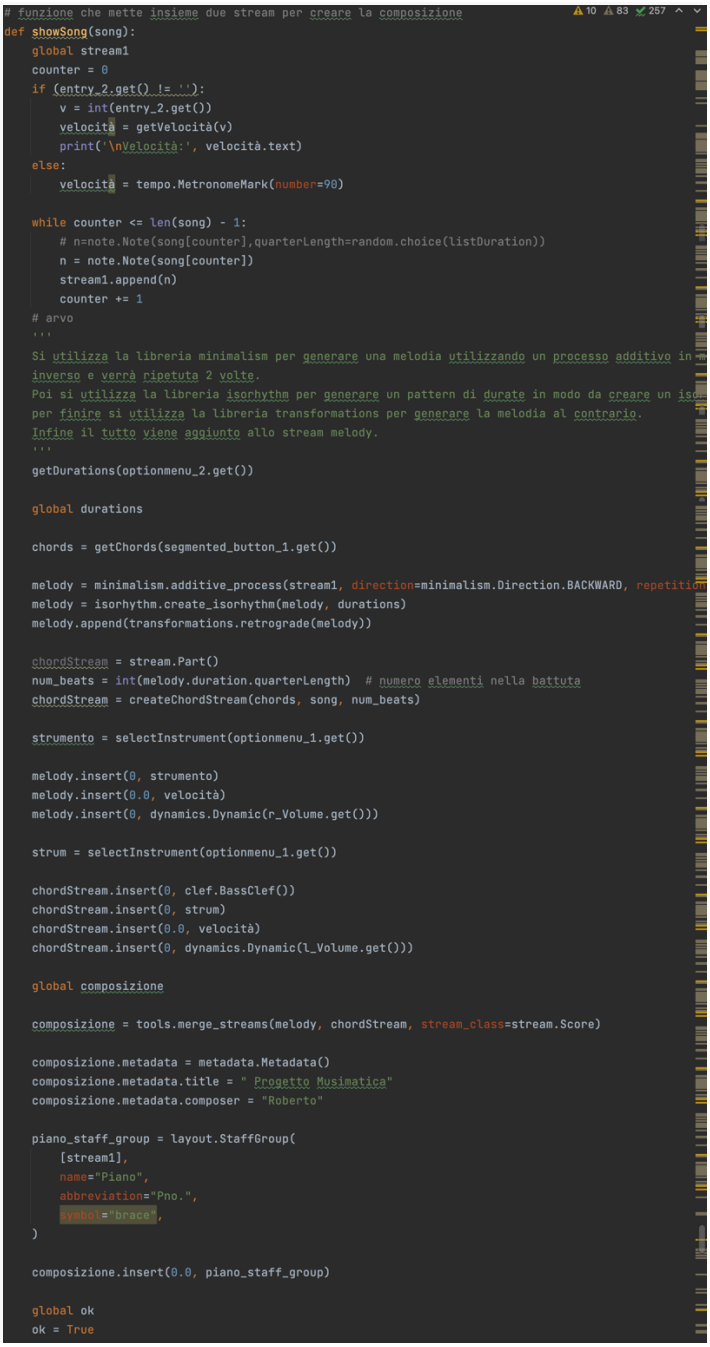
Funzioni mappatura sequenza fibonacci in note in dettaglio:

```python
# funzione che crea la melodia utilizzando la sequenza di fibonacci
def fibonacciComposer(scala, n):
    song = []
    a = 0
    b = 1

    for i in range(1, n + 1):
        fib = a + b
        a = b
        b = fib
        song = fib2Array(fib, song, scala)
        print(song)
        # text_msg.insert(tkinter.END, "\nNote inserite:  ")
        # text_msg.insert(tkinter.END, song)

    # print('Note nella canzone: ',end='')
    # for i in song:
    #   print (i + ', ', end='')
    text_msg.insert(tkinter.END, "\n  ")
    text_msg.insert(tkinter.END, "\n----------------- Note nella canzone -------------\n  ")
    text_msg.insert(tkinter.END, song)
    return song


# funzione che converte un numero della seq di fibonacci in array di interi
def fib2Array(seq, song, scala):
    # convert numeri in un array di interi
    res = [int(x) for x in str(seq)]
    # stampa risultato
    print("Numero estratto dalla seq di fibonacci ", res)
    # text_msg.insert(tkinter.END, "\nNumero estratto dalla seq di fibonacci:  ")
    # text_msg.insert(tkinter.END, res)

    # Analizza i singoli numeri che compongono un elemento della seq di fibonacci e assegna ad
    i = 0
    while (i <= len(res) - 1):
        if (res[i] == 0):
            song.append(scala[0])
        elif (res[i] == 1):
            song.append(scala[1])
        elif (res[i] == 2):
            song.append(scala[2])
        elif (res[i] == 3):
            song.append(scala[3])
        elif (res[i] == 4):
            song.append(scala[4])
        elif (res[i] == 5):
            song.append(scala[5])
        elif (res[i] == 6):
            song.append(scala[6])
        elif (res[i] == 7):
            song.append(scala[7])
        elif (res[i] == 8):
            song.append(scala[8])
        elif (res[i] == 9):
            song.append(scala[9])
        i = i + 1
    return song
```

In fibonacciComposer viene creata la sequenza di fibonacci in base alla lunguezza specificata dall'utente (n).
Per ogni elemento della sequenza viene utilizzata la funzione fib2Array che, come suggerisce il nome, trasforma la serie di numeri in un array di numeri indipendenti, che verranno usati per riempire la nostra melodia (song) utilizzando la regola di mappatura specificata sopra.

La funzione showSong mette insieme tutti i pezzi per creare la composizione definitiva. Utilizza gli oggetti stream e note di music21 per creare la partitura. Vengono poi utilizzate le funzioni della libreria arvo per modificare la melodia di base.

Si utilizza la libreria minimalism per generare una melodia utilizzando un processo additivo in modo tale che verranno usate note in ordine inverso e verrà ripetuta 2 volte.
Poi si utilizza la libreria isorhythm per generare un pattern di durate in modo da creare un isoritmo; per finire si utilizza la libreria transformations per generare la melodia al contrario.
Infine, il tutto viene aggiunto allo stream melody.
Vengono poi aggiunti allo Strem principale sia la melodia (melody) che l'armonia di accompagnamento (chordStream)

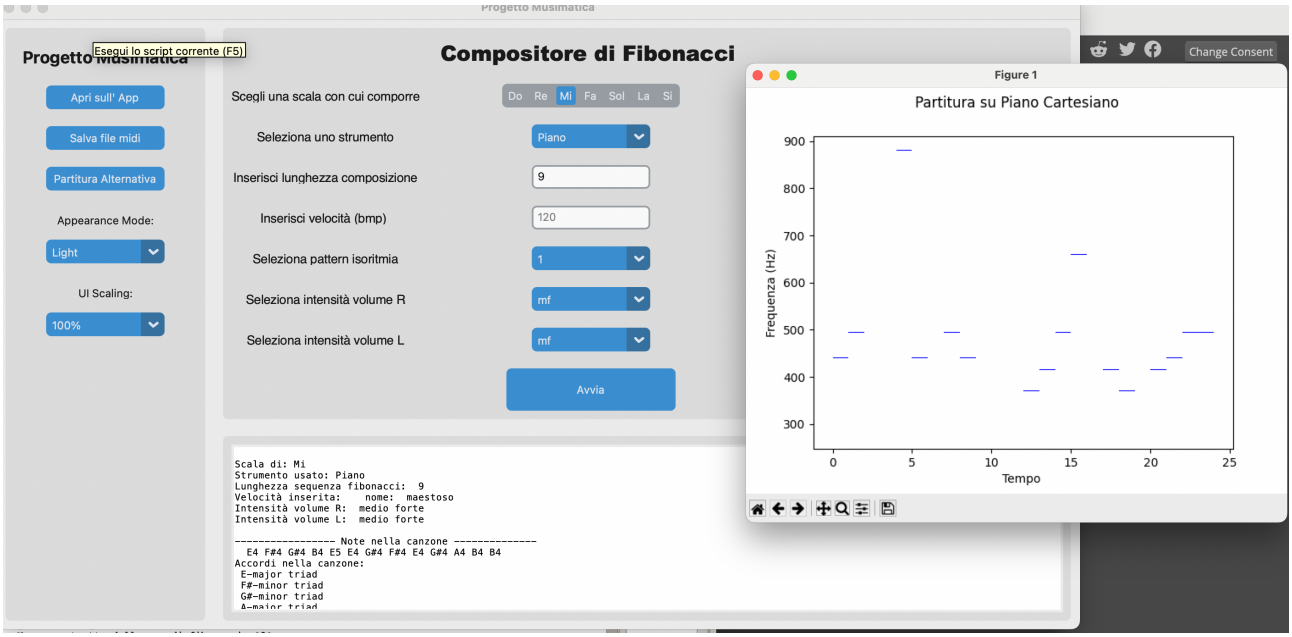


```python
# funzione che mette insieme due stream per creare la composizione
def showSong(song):
    global stream1
    counter = 0
    if (entry_2.get() != ''):
        v = int(entry_2.get())
        velocità = getVelocità(v)
        print('\nVelocità:', velocità.text)
    else:
        velocità = tempo.MetronomeMark(number=90)

    while counter <= len(song) - 1:
        # n=note.Note(song[counter],quarterLength=random.choice(listDuration))
        n = note.Note(song[counter])
        stream1.append(n)
        counter += 1
    # arvo
    '''
    Si utilizza la libreria minimalism per generare una melodia utilizzando un processo additivo in
    inverso e verrà ripetuta 2 volte.
    Poi si utilizza la libreria isorhythm per generare un pattern di durate in modo da creare un iso
    per finire si utilizza la libreria transformations per generare la melodia al contrario.
    Infine il tutto viene aggiunto allo stream melody.
    '''
    getDurations(optionmenu_2.get())

    global durations

    chords = getChords(segmented_button_1.get())

    melody = minimalism.additive_process(stream1, direction=minimalism.Direction.BACKWARD, repetition
    melody = isorhythm.create_isorhythm(melody, durations)
    melody.append(transformations.retrograde(melody))

    chordStream = stream.Part()
    num_beats = int(melody.duration.quarterLength)  # numero elementi nella battuta
    chordStream = createChordStream(chords, song, num_beats)

    strumento = selectInstrument(optionmenu_1.get())

    melody.insert(0, strumento)
    melody.insert(0.0, velocità)
    melody.insert(0, dynamics.Dynamic(r_Volume.get()))

    strum = selectInstrument(optionmenu_1.get())

    chordStream.insert(0, clef.BassClef())
    chordStream.insert(0, strum)
    chordStream.insert(0.0, velocità)
    chordStream.insert(0, dynamics.Dynamic(l_Volume.get()))

    global composizione

    composizione = tools.merge_streams(melody, chordStream, stream_class=stream.Score)

    composizione.metadata = metadata.Metadata()
    composizione.metadata.title = " Progetto Musimatica"
    composizione.metadata.composer = "Roberto"

    piano_staff_group = layout.StaffGroup(
        [stream1],
        name="Piano",
        abbreviation="Pno.",
        symbol="brace",
    )

    composizione.insert(0.0, piano_staff_group)

    global ok
    ok = True
```

Il brano creato può essere visualizzato sotto forma di partitura classica utilizzando programmi come MuseScore.

Il brano può essere salvato nella directory del pc sotto forma di file midi, in modo tale da poterlo utilizzare in altri programmi.

Le note ottenute dalla sequenza di Fibonacci, senza le alterazioni della libreria arvo, possono essere rappresentate usando un sistema di assi cartesiani in cui le ordinate indicano la frequenza e le ascisse il tempo. Ogni nota sarà rappresentata da un segmento orizzontale il cui inizio e la cui fine, e di conseguenza la cui durata, dipendono dalla posizione della nota nella partitura e dalla durata della nota. La Figura riporta la partitura di una sequenza di Fibonacci con lunghezza 9.


Progetto Musimatica
Esegui lo script corrente (F5)
Progetto Musimatica
Apri sull' App
Salva file midi
Partitura Alternativa
Appearance Mode:
Light
UI Scaling:
100%
Compositore di Fibonacci
Scegli una scala con cui comporre
Do Re Mi Fa Sol La Si
Seleziona uno strumento
Piano
Inserisci lunghezza composizione
9
Inserisci velocità (bmp)
120
Seleziona pattern isoritmia
1
Seleziona intensità volume R
mf
Seleziona intensità volume L
mf
Avvia
Scala di: Mi
Strumento usato: Piano
Lunghezza sequenza fibonacci: 9
Velocità inserita: nome: maestoso
Intensità volume R: medio forte
Intensità volume L: medio forte
---------- Note nella canzone ----------
E4 F#4 G#4 B4 E5 E4 G#4 F#4 E4 G#4 A4 B4 B4
Accordi nella canzone:
E-major triad
F#-minor triad
G#-minor triad
Change Consent
Figure 1
Partitura su Piano Cartesiano
Frequenza (Hz)
900
800
700
600
500
400
300
0
5
10
15
20
25
Tempo