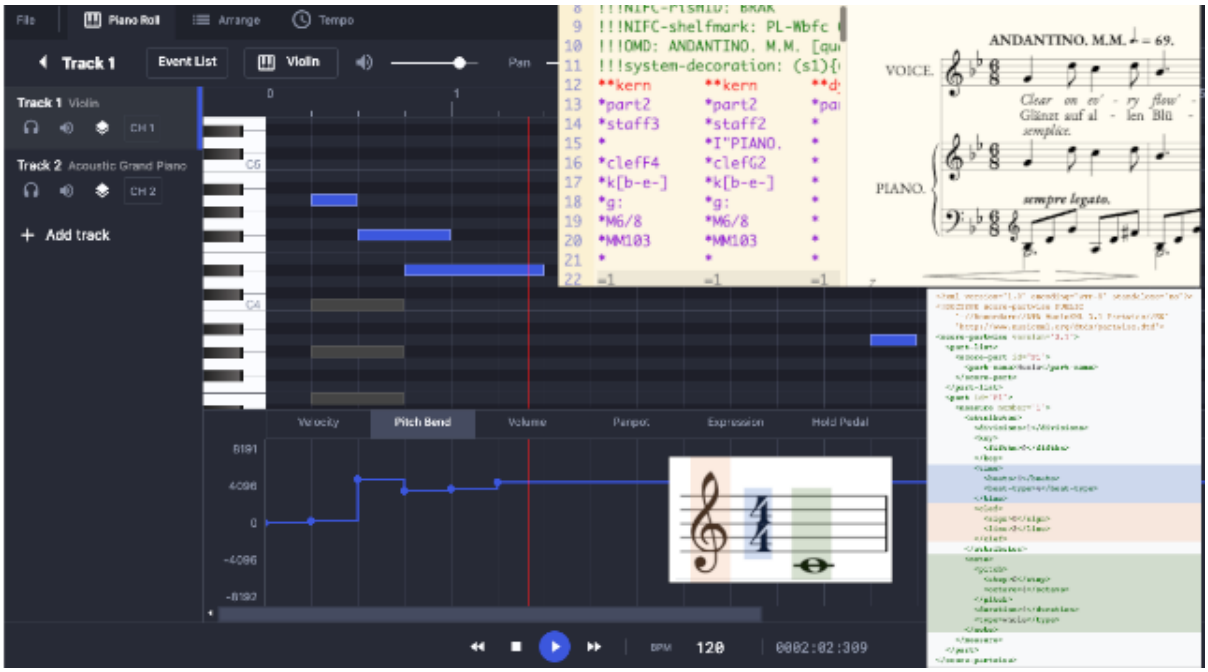
Ελληνικό Μεσογειακό Πανεπιστήμιο
Τμήμα Μουσικής Τεχνολογίας και Ακουστικής

Δημιουργία, επεξεργασία κι απόδοση ψηφιακών αναπαραστάσεων της μουσικής με προγραμματιστικά εργαλεία της Python



Χ. Αλεξανδράκη
Μ. Βάσσης

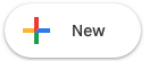
- Ρέθυμνο 2024 -

Πίνακας Περιεχομένων

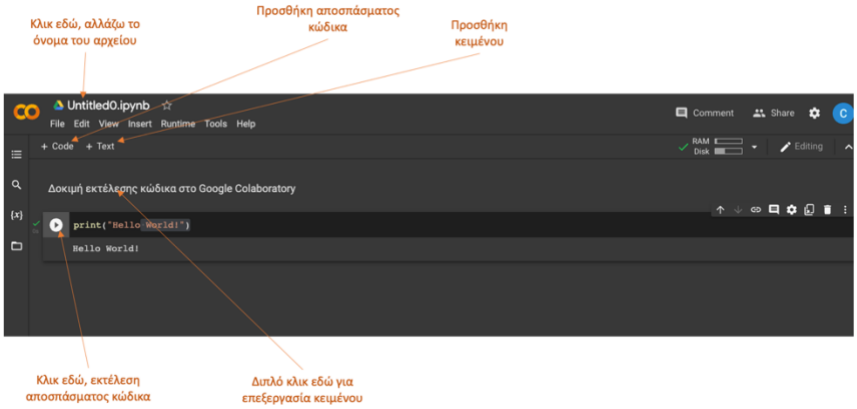
1 Περιβάλλον Εργασίας

Στις ασκήσεις του μαθήματος **Ψηφιακές Αναπαραστάσεις της Μουσικής** θα δουλέψουμε με διαδικτυακά εργαλεία και βιβλιοθήκες ανοικτού κώδικα στην Python. Ειδικότερα για αποθήκευση των αρχείων σας θα χρησιμοποιήσετε το Google Drive και για την υλοποίηση προγραμμάτων που διαβάζουν συνθέτουν κι επεξεργάζονται ψηφιακές μουσικές αναπαραστάσεις θα χρησιμοποιήσουμε το περιβάλλον Google Colaboratory.

1.1 Οδηγίες Χρήσης Google Colaboratory

1. Συνδεθείτε με τον **ιδρυματικό λογαριασμό** σας (π.χ. ta999@edu.hmu.gr) στο drive.google.com
2. Εκεί στο **My Drive** (στη μπάρα αριστερά) δημιουργήστε το φάκελο **DigitalMusicReporesentations** πατώντας το κουμπί 

. Ο φάκελος αυτός θα περιέχει όλες τις ασκήσεις που θα υλοποιήσουμε στο μάθημα.
3. Πλοηγηθείτε στο εσωτερικό του φακέλου και επιλέξτε από το ίδιο κουμπί New->Google Colaboratory. Το περιβάλλον Google Colaboratory μας επιτρέπει να εκτελούμε κώδικα της Python μέσα από ένα Internet Browser (κατά προτίμηση Google Chrome) χωρίς να εγκαταστήσουμε κάτι στον υπολογιστή μας.
4. Φτιάξτε ένα μικρό παράδειγμα κώδικα της Python για να δείτε πώς λειτουργεί, όπως στην Εικόνα 1-1.



Εικόνα 1-1: Δημιουργία αρχείου Google Colaboratory στο Google Drive.

1.2 Πρόσβαση σε αρχεία του Google Drive

Προκειμένου να μπορείτε να διαβάσετε και να γράφετε αρχεία στο περιβάλλον του Google Colaboratory θα πρέπει να εκτελεστεί μία διαδικασία που είναι γνωστή ως file system mounting. Στο Google Colaboratory η διαδικασία αυτή εκτελείται με το ακόλουθο απόσπασμα κώδικα.

```
from google.colab import drive

drive.mount('/content/gdrive')

%cd /content/gdrive/MyDrive/DigitalMusicRepresentations

!ls
```

Προσέξτε ότι η 3η γραμμή κάνει αλλαγή φακέλου για να δώσει πρόσβαση στα αρχεία (π.χ. Midi αρχεία) που έχετε μέσα στον τρέχοντα φάκελο. Στην προκειμένη περιπτωση υποθέτουμε ότι δουλεύετε στο φάκελο **MyDrive/DigitalMusicRepresentations** του Google Drive. Σε διαφορετική περίπτωση τροποποιήστε την 3η γραμμή κώδικα, ώστε να δείχνει το στο φάκελο στον οποίο θα δουλεύετε.

Κατά την εκτέλεση αυτού του αποσπάσματος, θα σας ζητηθεί να επιβεβαιώσετε την ταυτότητα χρήστη ώστε να επιτραπεί η πρόσβαση στα αρχεία σας.
Όπως φαίνεται στην τελευταία γραμμή, το Google Colaboratory σας επιτρέπει να εκτελείτε εντολές του λειτουργικού συστήματος Linux εφόσον αυτές προηγούνται από το χαρακτήρα θαυμαστικό (!). Στην πραγματικότητα το Google Colaboratory σας δίνει πρόσβαση σε ένα εικονικό υπολογιστή στο υπολογιστικό νέφος της Google, ο οποίος χρησιμοποιεί το περιβάλλον Linux.
Όσον αφορά την εντολή αλλαγής φακέλου (cd) θέλω να επηρεάσει το ίδιο το περιβάλλον εκτέλεσης, ώστε η πρόσβαση στο νέο φάκελο να διατηρηθεί. Για τέτοιου τύπου αλλαγές, σε περιβάλλοντα όπως το Google Colab, το σύμβολο % χρησιμοποιείται για τις λεγόμενες magic commands του IPython, που επηρεάζει το ίδιο το περιβάλλον εκτέλεσης.

2 Απόδοση αρχείων Midi στο Google Colaboratory

Τα αρχεία MIDI μπορούν να αποδοθούν ηχητικά και γραφικά (παρτιτούρα) μέσω πληθώρας προγραμμάτων που περιλαμβάνουν μηχανές γραφικής και ηχητικής απόδοσης. Τέτοια προγράμματα είναι το MuseScore, ο Online MIDI Editor, το Finale, το Sibelius, κ.α.

Στις ακόλουθες δύο ενότητες περιγράφεται ο τρόπος για τη γραφική και ηχητική απόδοση αρχείων MIDI με προγραμματιστικό τρόπο και μάλιστα πως να εμφανίσουμε την απόδοση αυτή μέσα στο περιβάλλον του Google Colaboratory.

2.1 Ηχητική απόδοση

Για την ηχητική απόδοση θα χρησιμοποιήσουμε το FluidSynth, το οποίο είναι μια προγραμματιστική βιβλιοθήκη, δηλαδή ένα Application Programming Interface (API) υλοποιημένο στη γλώσσα προγραμματισμού C. Το FluidSynth παρέχει λειτουργίες σύνθεσης ήχου σε πραγματικό χρόνο βασισμένης σε δείγματα ήχου που ακολουθούν το πρότυπο SoundFont2. Το πρότυπο αυτό αναπτύχθηκε τη δεκαετία του ‘90 από τις εταιρίες Creative και E-mu Systems και αφορά στη μορφή, δηλαδή στο file format, των ηχητικών δειγμάτων που χρησιμοποιούνται από MIDI Software Synthesizers.

Για να μπορέσουμε να χρησιμοποιήσουμε το FluidSynth για την ηχητική απόδοση αρχείων MIDI θα πρέπει να χρησιμοποιήσουμε ένα συνδυασμό εντολές του περιβάλλοντος Linux και της Python. Υπενθυμίζεται άλλωστε ότι το Google Colaboratory σας παρέχει πρόσβαση σε έναν υπολογιστή στο νέφος της google, ο οποίος βασίζεται στο λειτουργικό Linux.

Συγκεκριμένα θα χρειαστούν τα εξής:

1. Εγκατάσταση του FluidSynth API στο περιβάλλον Linux

```
# install fluidsynth for sound rendering
!apt-get install fluidsynth > /dev/null
```

2. Ορισμός συνάρτησης για την ηχητική απόδοση. Η παρακάτω συνάρτηση παίρνει ως όρισμα εισόδου ένα MIDI file και στην έξοδο παράγει ένα audio player που περιλαμβάνει την ηχητική απόδοση του ηχητικού αρχείου

```
# Import Python packages for Audio diplay
from IPython.display import  Audio


def renderSound(midifile):
  #run fluidsynth linux command to generate wav file
  !fluidsynth -ni /usr/share/sounds/sf2/FluidR3_GM.sf2  $midifile -F \
$midifile\.wav -r 44100 > /dev/null
  #display wav file
  display(Audio(midifile + '.wav'))
```

3. Κλίση της συνάρτηση

   Για να κληθεί η συνάρτηση αυτή θα πρέπει προηγουμένως να έχει γίνει mount το google drive, ώστε να μπορεί να αναγνωσθεί το αρχείο MIDI. Το αποτέλεσμα αυτής της κλίσης αποδίδεται στο ακόλουθο παράδειγμα:

```
[4]  renderSound('la_bamba.mid')
```

2.2 Γραφική απόδοση

Κατ' αντιστοιχία, για τη γραφική απόδοση της σημειογραφίας που αντιστοιχεί σε ένα αρχείο MIDI θα χρησιμοποιήσουμε το LilyPond. το οποίο είναι ένα προγραμματιστικό εργαλείο για τη γραφική αναπαράσταση μουσικής σημειογραφίας. Το Lilypond χρησιμοποιεί το δικό του file format για τα αρχεία μουσικού περιεχομένου το οποίο είναι ένα ASCII format. Ένα παράδειγμα ενός τέτοιου αρχείου φαίνεται στη συνέχεια:

```
\version "2.14.1"
\include "english.ly"

\score {
  \new Staff {
    \key d \major
    \numericTimeSignature
    \time 2/4
    <cs' d'' b''>16 <cs' d'' b''>8.
    %% Here: the tie on the D's looks funny
    %% Too tall? Left-hand endpoint is not aligned with the B tie?
    ~
    <cs' d'' b''>8 [ <b d'' a''> ]
  }
}
```

Επιπλέον, το LilyPond παρέχει μια σειρά από βοηθητικά προγράμματα (utilities) που επιτρέπουν τη μετατροπή διαφόρων τύπων αρχείων σε .ly.
Τα αρχεία .ly μπορούν στη συνέχεια να μετατραπούν σε αρχείο εικόνας ή γραφικών μέσω του lilypond engraving engine.
Για να μπορέσουμε να χρησιμοποιήσουμε το LilyPond για τη γραφική απόδοση παρτιτούρας από αρχεία MIDI θα πρέπει να χρησιμοποιήσουμε ένα συνδυασμό εντολές του περιβάλλοντος Linux και της Python. Υπενθυμίζεται άλλωστε ότι το Google Colaboratory σας παρέχει πρόσβαση σε έναν υπολογιστή στο νέφος της google, ο οποίος βασίζεται στο λειτουργικό Linux.
Συγκεκριμένα θα χρειαστούν τα εξής:

1. Εγκατάσταση του LilyPond στο περιβάλλον Linux

```
# install lilypond for score  rendering
!apt-get update -qq > /dev/null
!apt-get install -y lilypond > /dev/null
```

2. Ορισμός συνάρτησης για τη γραφική απόδοση παρτιτούρας. Η παρακάτω συνάρτηση παίρνει ως όρισμα εισόδου ένα MIDI file, το οποίο αρχικά μετατρέπεται σε .ly file κι έπειτα εξάγεται σε png προκειμένου να αναπαρασταθεί ως εικόνα. Επισημαίνεται ότι το Lilypond έχει τη δυνατότητα να μετατρέψει τα ly αρχεία και σε άλλα image formats (π.χ. SVG). Εδώ επιλέγουμε το png καθώς το svg format δεν υποστηρίζεται προς το παρόν από το google colaboratory.

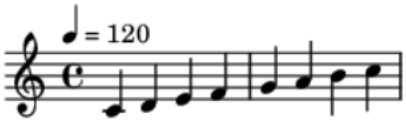
```
# Import Python packages for Image diplay
from IPython.display import  Image

def renderScore(midifile):
  # convert midifile to .ly file
  !midi2ly $midifile -o lily.ly
  !lilypond -fpng lily.ly
  display(Image('lily.png'))
```

3. Κλίση της συνάρτηση

Για να κληθεί η συνάρτηση αυτή θα πρέπει προηγουμένως να έχει γίνει mount το google drive, ώστε να μπορεί να αναγνωσθεί το αρχείο MIDI. Το αποτέλεσμα αυτής της κλίσης αποδίδεται στο ακόλουθο παράδειγμα:

```
renderScore('C_Major.mid')
```

```
LY output to `lily.ly'...
GNU LilyPond 2.22.1
Processing `lily.ly'
Parsing...
Interpreting music...
Preprocessing graphical objects...
Interpreting music...
MIDI output to `lily.midi'...
Finding the ideal number of pages...
Fitting music on 1 page...
Drawing systems...
Converting to PNG...
Success: compilation successfully completed
```



3 Mido - MIDI Objects for Python

Το Python πακέτο mido μας επιτρέπει την ανάγνωση, εγγραφή και επεξεργασία αρχείων MIDI. Για το mido ένα αρχείο Midi αποτελείται από tracks και κάθε track αποτελείται από μηνύματα Midi. Τα midi μηνύματα είναι δύο ειδών: τα meta messages που περιγράφουν ένα track και τα midi messages. Τα meta messages αφορούν περιγραφή των tracks (π.χ. ρυθμός tempo, lyrics, κ.α.) και δε μεταδίδονται ως πληροφορία midi σε ηλεκτρονικά μουσικά όργανα. Τα midi messages είναι εκείνα που περιγράφουν την επικοινωνία των μουσικών οργάνων.

3.1 Εγκατάσταση πακέτου mido

Σε ένα νέο απόσπασμα κώδικα του Colaboratory γράψτε την ακόλουθη γραμμή, ώστε να εγκατασταθεί το πακέτο Mido, ότι δηλαδή θα εκτελούσατε σε ένα λειτουργικό σύστημα στο οποίο έχετε προηγουμένως εγκαταστήσει τη γλώσσα Python.

```
!pip install mido
```

3.2 Ανάγνωση αρχείων MIDI

Υποθέτοντας ότι έχετε ανεβάσει στο φάκελο MyDrive/ DigitalMusicRepresentations το αρχείο C_Major.mid, το παρακάτω απόσπασμα φορτώνει το αρχείο αυτό στη μνήμη του προγράμματος και εκτυπώνει τα περιεχόμενά του.

```
import mido
midifile = mido.MidiFile("C_Major.mid")
print(midifile)
```

Από την εκτύπωση του συγκεκριμένου αρχείου παράγεται το ακόλουθο:

```
MidiFile(type=1, ticks_per_beat=480, tracks=[
  MidiTrack([
    MetaMessage('track_name', name='', time=0),
    MetaMessage('time_signature', numerator=4, denominator=4, clocks_per_click=24,
notated_32nd_notes_per_beat=8, time=0),
    MetaMessage('set_tempo', tempo=500000, time=0),
    MetaMessage('end_of_track', time=0)]),
  MidiTrack([
    Message('control_change', channel=0, control=121, value=0, time=0),
    MetaMessage('track_name', name='', time=0),
    Message('control_change', channel=0, control=10, value=64, time=0),
    Message('control_change', channel=0, control=7, value=100, time=0),
    Message('control_change', channel=0, control=11, value=127, time=0),
    Message('control_change', channel=0, control=101, value=0, time=0),
    Message('control_change', channel=0, control=100, value=2, time=0),
    Message('control_change', channel=0, control=6, value=64, time=0),
    Message('control_change', channel=0, control=101, value=0, time=0),
    Message('control_change', channel=0, control=100, value=1, time=0),
    Message('control_change', channel=0, control=6, value=64, time=0),
```

```
    Message('control_change', channel=0, control=38, value=0, time=0),
    Message('control_change', channel=0, control=101, value=0, time=0),
    Message('control_change', channel=0, control=100, value=0, time=0),
    Message('control_change', channel=0, control=6, value=12, time=0),
    Message('pitchwheel', channel=0, pitch=0, time=0),
    Message('control_change', channel=0, control=1, value=0, time=0),
    Message('program_change', channel=0, program=0, time=0),
    Message('note_on', channel=0, note=60, velocity=127, time=0),
    Message('note_off', channel=0, note=60, velocity=0, time=480),
    Message('note_on', channel=0, note=62, velocity=127, time=0),
    Message('note_off', channel=0, note=62, velocity=0, time=480),
    Message('note_on', channel=0, note=64, velocity=127, time=0),
    Message('note_off', channel=0, note=64, velocity=0, time=480),
    Message('note_on', channel=0, note=65, velocity=127, time=0),
    Message('note_off', channel=0, note=65, velocity=0, time=480),
    Message('note_on', channel=0, note=67, velocity=127, time=0),
    Message('note_off', channel=0, note=67, velocity=0, time=480),
    Message('note_on', channel=0, note=69, velocity=127, time=0),
    Message('note_off', channel=0, note=69, velocity=0, time=480),
    Message('note_on', channel=0, note=71, velocity=127, time=0),
    Message('note_off', channel=0, note=71, velocity=0, time=480),
    Message('note_on', channel=0, note=72, velocity=127, time=0),
    Message('note_off', channel=0, note=72, velocity=0, time=480),
    MetaMessage('end_of_track', time=0)])
])
```

Από αυτήν την εκτύπωση προσέξτε τα εξής:

- **`type=1`** Υπάρχουν τρεις τύποι αρχείων MIDI. Οι τύπο είναι:
    - type 0 (single track): all messages are saved in one track
    - type 1 (synchronous): all tracks start at the same time
    - type 2 (asynchronous): each track is independent of the others
- **`ticks_per_beat=480`** Η μεταβλητή αυτή ορίζει τον τρόπο που θα ορίζεται η διαφορά χρόνου ανάμεσα σε διαδοχικά μηνύματα Midi. Συνήθως η τιμή αυτή ισούται με κάποιο πολλαπλάσιο του 120, ενώ η default (συνηθισμένη) τιμή της είναι 480. Προσέξτε ότι σε όλα τα tracks κάθε μήνυμα περιλαμβάνει μια παράμετρο με το όνομα **`time.`** Η παράμετρος αυτή στην πραγματικότητα περιγράφει τη διαφορά χρόνου του μηνύματος από το προηγούμενο μήνυμα σε ticks. Οπότε, καθώς στο δεύτερο track κάθε μήνυμα note_on ακολουθείται από note_off οι παράμετροι time υποδηλώνουν ότι το σταμάτημα κάθε νότας έρχεται 480 ticks μετά το ξεκίνημα της νότας αυτής, δηλαδή ότι η κάθε νότα έχει διάρκεια ένα beat. Καθώς δε το επόμενο note_on μήνυμα εμφανίζεται με time=0 σημαίνει ότι ότι η επόμενη νότα ενεργοποιείται τη στιγμή που ταματάει η προηγούμενη.

- **`MidiTrack`** Ως τρίτο όρισμα το MidiFile έχει μία λίστα αποτελούμενη από δύο tracks. Το κάθε track περιλαμβάνει μία λίστα από μηνύματα τύπου MetaMessage και απλό Message.
    - **`MetaMessage`** Τα αρχεία Midi περιλαμβάνουν μηνύματα meta, τα οποία περιγράφουν το αρχείο Midi και τα οποία δεν αποστέλλονται κατά την επικοινωνία μουσικών οργάνων. Χρησιμοποιούνται δηλαδή μονάχα για την περιγραφή του αρχείου. Τα μηνύματα αυτά περιλαμβάνουν πληροφορίες για το ρυθμό, το τέμπο, την τονικότητα, τους στοίχους (lyrics) κ.α. Για περισσότερες πληροφορίες δείτε MIDI meta messages | Recordingblogs και για την υλοποίησή τους στο πακέτο mido Meta Message Types. Στο συγκεκριμένο παράδειγμα περιλαμβάνονται τα εξής meta messages:
        - **`track_name`** σε κάθε track περιλαμβάνεται ένα τέτοιο meta message στο οποίο προσδιορίζεται το όνομα του track, εφόσον υπάρχει.
        - **`time_signature`** Το μέτρο (time signature) ορίζεται από το κλάσμα πχ 4/4 (numerator = 4, denominator = 4). Η λέξη numerator σημαίνει αριθμητής και η denominator παρονομαστής.
        - **`set_tempo`** Το tempo στο MIDI αναγράφεται σε μsec = 10^{-6} sec ανά beat. Τα 333333 μsec αντιστοιχούν στα 180 bpm. Γιατί? Ποια τιμή μsec πρέπει να δώσω για tempo 200bpm? Απάντηση – απλή μέθοδο των τριών:

            200 beats σε χρόνο 1min = 60 sec = 60* 10^6 μsec
            1 beat σε πόσο χρόνο x
            x = 60* 10^6μsec/200 = 0.3*10^6 μsec = 300000 μsec

            Εναλλακτικά για τη μετατροπή από MIDI tempo σε bpm μπορεί να χρησιμοποιηθεί η συνάρτηση **`mido.tempo2bpm(333333)`** και για το αντίθετο **`mido.bpm2tempo(180)`**
        - **`end_of_track`** Στα Midi αρχεία track ξεκινάει με ένα meta message track_name και ολοκληρώνεται με ένα end_of_track
    - **`Message`** τα μηνύματα αυτά περιλαμβάνουν τα μουσικά γεγονότα που περιλαμβάνονται σε ένα αρχείο και των οποίων τόσο τύπος, όσο και η παραμετροποίησή τους καθορίζεται από το πρωτόκολλο Midi

3.2.1 Τύποι και Ονόματα Μηνυμάτων

Οι τύποι των μηνυμάτων που υποστηρίζει το mido είναι οι ακόλουθοι:

Name	Keyword Arguments / Attributes
note_off	channel note velocity
note_on	channel note velocity
polytouch	channel note value
control_change	channel control value
program_change	channel program
aftertouch	channel value
pitchwheel	channel pitch
sysex	data
quarter_frame	frame_type frame_value

songpos	pos
song_select	song
tune_request	
clock	
start	
continue	
stop	
active_sensing	
reset	

3.2.2 Τύποι Παραμέτρων

Οι τύποι παραμέτρων (data bytes) που υποστηρίζει το Mido είναι:

Name	Valid Range	Default Value
channel	0..15	0
frame_type	0..7	0
frame_value	0..15	0
control	0..127	0
note	0..127	0
program	0..127	0
song	0..127	0
value	0..127	0
velocity	0..127	64
data	(0..127, 0..127, ...)	() (empty tuple)
pitch	-8192..8191	0
pos	0..16383	0
time	any integer or float	0

3.3 Δημιουργία μεμονωμένων μηνυμάτων MIDI και μετατροπή σε Bytes/Hex

Στο ακόλουθο απόσπασμα φαίνεται το πώς μπορώ να δημιουργώ μηνύματα MIDI να επηρεάζω τις παραμέτρους data bytes τους ή να αφήνω τις default τιμές παραμέτρων, καθώς και να μετατρέπω τα μηνύματα σε Bytes ή Hexadecimal Bytes.

```
# Create Messages
m1 = mido.Message('note_on', note=60)
print(m1)

cc1 = mido.Message('control_change', control=7, value=120)
print(cc1)

pb = mido.Message('pitchwheel', pitch=8191)
print(pb)
```

```
# Μετατροπή σε bytes και δεαεξαδικό
print(m1.bytes())
print(m1.hex())
```

3.4 Δημιουργία αρχείου MIDI και αποθήκευση

Μέσω του πακέτου mido μπορούν να δημιουργηθούν tracks και σε αυτά να εισαχθούν MIDI messages/events. Τα tracks έπειτα εισάγονται σε αρχείο και τέλος το αρχείο αποθηκεύεται στον φάκελο που ορίσαμε.
Περισσότερες λεπτομέρειες στο ακόλουθο link: Create MIDI file
Η συνάρτηση **`mido.MidiFile()`** δημιουργεί ένα νέο αντικείμενο ενός MIDI file. Μπορεί προαιρετικά να πάρει δύο ορίσματα:

- **`type=1`** Υπάρχουν τρεις τύποι αρχείων MIDI. Καλό είναι να χρησιμοποιούμε τον προεπιλεγμένο τύπο, καθώς έχει τους λιγότερους περιορισμούς. Οι τύπο είναι:
  - type 0 (single track): all messages are saved in one track
  - type 1 (synchronous): all tracks start at the same time
  - type 2 (asynchronous): each track is independent of the others
- **`ticks_per_beat=480`** Η μεταβλητή αυτή ορίζει τον τρόπο που θα ορίζεται η διαφορά χρόνου ανάμεσα σε διαδοχικά Midi events. Καλό είναι να τη θέσετε σε κάποιο πολλαπλάσιο του 120 ή να χρησιμοποιήσετε τη default τιμή που είναι 480

```
Create new Midi file
c_maj = mido.MidiFile(type=1, ticks_per_beat=120)
#Create new Midi Track
track = mido.MidiTrack()

# Append Messages to Midi Track
notes =[60, 62, 64, 65, 67, 69, 71, 72]
for n in notes:
  track.append(mido.Message('note_on', note=n, velocity=120, time=0))
  track.append(mido.Message('note_off', note=n, velocity=120, time=120))

# Append Track to MidiFile
c_maj.tracks.append(track)
#Save Midi File
c_maj.save('c_maj.mid')
```

Προσέξτε ότι το συγκεκριμένο αρχείο δεν περιλαμβάνει καθόλου meta messages κι επομένως δεν υπάρχουν πληροφορίες ούτε για το ρυθμό ούτε για το tempo, κι επομένως εάν το ανοίξω με κάποιον midi editor δε θα υπάρχουν μέτρα.
Ένα πιο ολοκληρωμένο παράδειγμα παρουσιάζεται παρακάτω:

```
# Create an empty file
```

```
mid = mido.MidiFile(ticks_per_beat=120)

# create 2 empty tracks
mtrack = mido.MidiTrack()
track1 = mido.MidiTrack()

# append meta messages in the 1st track
mtrack.append(mido.MetaMessage('track_name', name='note', time=0))
mtrack.append(mido.MetaMessage('set_tempo', tempo=333333, time=0))
mtrack.append(mido.MetaMessage('time_signature', numerator=4, denominator=4, time=0))
mtrack.append(mido.MetaMessage('end_of_track', time=0))

# append midi event messages in the second track
track1.append(mido.Message('program_change', program=42, time=0))
track1.append(mido.Message('note_on', note=36, velocity=120, time=0))

# Slide the pitch bend wheel from the bottom to the top position
dt =60
for i in range(-8191, 8191, 500):
  track1.append(mido.Message('pitchwheel', pitch=i, time=dt))

track1.append(mido.Message('note_off', note=36, velocity=0, time=120))
track1.append(mido.MetaMessage('end_of_track', time=0))

# add the two tracks in the midi file
mid.tracks.append(mtrack)
mid.tracks.append(track1)

# Save the Midi file
mid.save('note.mid')
```

Κατεβάστε το αρχείο που μόλις δημιουργήθηκε και ανοίξτε το με MuseScore και τον MIDI editor. Τι παρατηρείτε;

3.5 Επεξεργασία αρχείου MIDI

Η επεξεργασία ενός αρχείου Midi αρχείου μπορεί να περιλαμβάνει μία από τις ακόλουθες ενέργειες:

- Αλλαγή του είδους ή των παραμέτρων κάποιου μηνύματος. Η ενέργεια αυτή απαιτεί τον εντοπισμό του μηνύματος που πρέπει να υποστεί επεξεργασία και αλλαγή των κατάλληλων παραμέτρων. Παράδειγμα η αλλαγή τέμπο στην ενότητα 3.5.2

- Προσάρτηση κάποιου track ή κάποιου μηνύματος. Με τη συνάρτηση **`append()`** όπως γίνεται για παράδειγμα κατά τη δημιουργία ενός νέου αρχείου (ενότητα 3.4)
- Εισαγωγή νέου track ή νέου μηνύματος. Με τη συνάρτηση **`insert()`**, όπως γίνεται για παράδειγμα στην ενότητα 3.5.3
- Διαγραφή υπάρχοντος track ή μηνύματος. Εντοπισμός μηνύματος ή track και διαγραφή του με τη συνάρτηση **`remove()`**. Ένα παράδειγμα παρατίθεται στην ενότητα 3.5.1.

Γενικά, για την Python η δομή δεδομένων MidiFile είναι ένα αντικείμενο (object) που περιλαμβάνει τη μεταβλητή tracks, η οποία είναι μία λίστα από αντικείμενα τύπου MidiTrack. Το κάθε MidiTrack, είναι μία λίστα από αντικείμενα MetaMessage και Message. Επομένως, για το χειρισμό/επεξεργασία της λίστας μπορεί να χρησιμοποιηθεί οποιαδήποτε από τις μεθόδους του ακόλουθου πίνακα.

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list

sort()	Sorts the list

3.5.1 Εντοπισμός μηνύματος ή track μέσω δομών επανάληψης

Στο ακόλουθο παράδειγμα, το πρόγραμμα φορτώνει στη μνήμη το αρχείο C_major.mid και διαγράφει tracks των οποίων το όνομα είναι ‘percussion’, ενώ επίσης τροποποιεί μηνύματα τύπου pitchwheel στην τιμή pitch=0.

```
import mido
midifile = mido.MidiFile('C_Major.mid')
for t in midifile.tracks:
  print('------ NEW TRACK ------')
  if t.name == 'percussion':
    midifile.tracks.remove(t)
  for m in t:
    if m.type == 'pitchwheel':
      m.pitch = 0
```

3.5.2 Αλλαγή tempo

Στο παρακάτω απόσπασμα κώδικα φορτώνουμε ένα αρχείο MIDI του οποίου το αρχικό tempo είναι bpm, βρίσκουμε το MetaMessage με το όνομα set_tempo και του αλλάζουμε την τιμή:

```
fr_elise = mido.MidiFile("Fr_Elise.mid")

print("Each beat contains ", fr_elise.ticks_per_beat, "ticks" )

noTracks = len(fr_elise.tracks)
print("The file has ", noTracks, " tracks")

## TEMPO CHANGE
for m in fr_elise.tracks[0]: #tempo is in the first MIDI track
  if m.is_meta == True and m.type == 'set_tempo':
    m.tempo = 300000 #αλλαγή Tempo σε 200bpm
fr_elise.save("fr_elise_1.mid")
```

Παρατηρήστε το αποτέλεσμα (fr_elise_1.mid) στο MuseScore ή το MIDI Editor.

3.5.3 Αλλαγή Μουσικού Οργάνου

Το παρακάτω απόσπασμα, ψάχνει στο δεύτερο track για program change message. Εάν το βρει βάζει το νέο program (program=19 το εκκλησιαστικό όργανο).
Εάν δεν το βρει, δημιουργεί ένα program change μήνυμα το οποίο και κάνει insert ως το πρώτο μήνυμα του δεύτερου track.

```
## PROGRAM CHANGE
found_PG = False #check if there is a program change message
for m in fr_elise.tracks[1]:
  if m.type == 'program_change':
    found_PG = True
    m.program = 19 #Church Organ

if not found_PG:
  pg_msg = mido.Message('program_change', program=19, time=0) #create a program change message
  fr_elise.tracks[1].insert(0, pg_msg) #insert this message as the first message of the second track

fr_elise.save('fr_elise_2.mid')
```

Παρατηρήστε το αποτέλεσμα (fr_elise_2.mid) στο MuseScore

3.5.4 Αλλαγή Τονικότητας

Το παρακάτω απόσπασμα προσθέτει +7 στα note numbers όλων των note_on και note_off μηνυμάτων.

```
## Αλλαγή Τονικότητας
i=0
for t in fr_elise.tracks:
  i = i+1
  print("--------- TRACK ", i, " ---------")
  for msg in t:
    if msg.type == 'note_on' or msg.type == 'note_off':
      msg.note +=7 # 7 ημιτόνια πάνω

fr_elise.save('fr_elise_3.mid')
```

Παρατηρήστε το αποτέλεσμα (fr_elise_3.mid) στο MuseScore

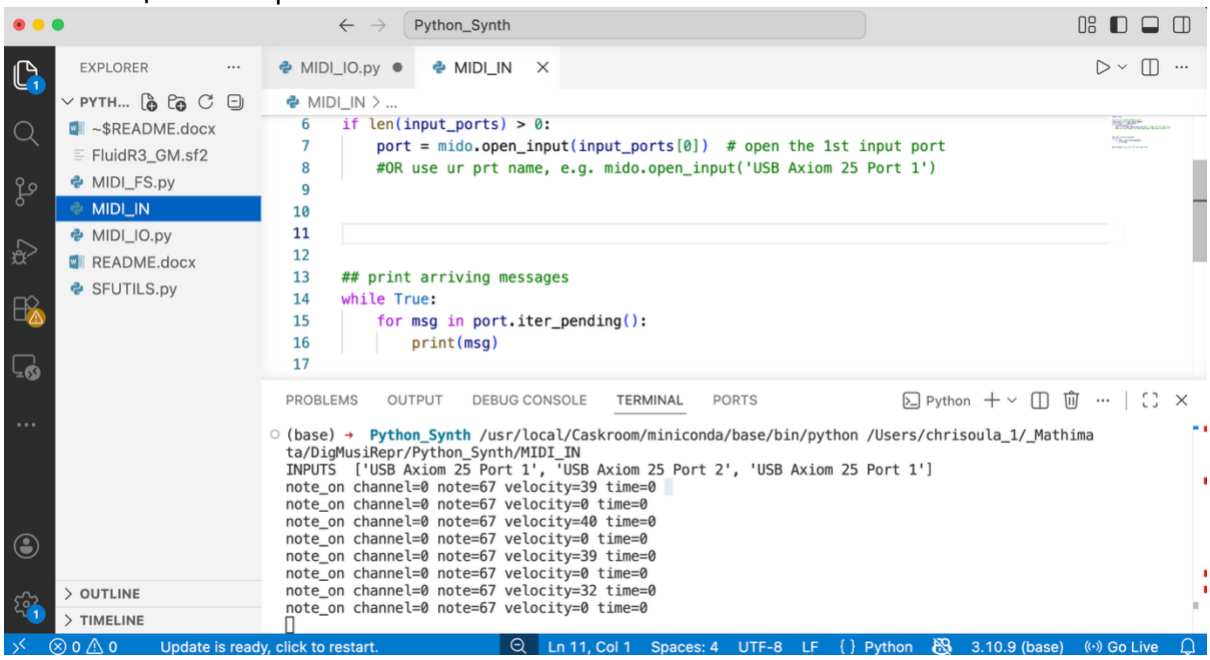
3.5.5 Αντικατάσταση Μεμονωμένου Μηνύματος

Στο παρακάτω αντικαθίστανται όλες οι νότες πάνω από το μεσαίο ντο με τη νότα του μεσαίου ντο.

```
# Αντικατάσταση όλων των νοτών πάνω από το μεσαίο Ντο (> 60)
for track in fr_elise.tracks:

  for msg in track:
    if (msg.type == 'note_on' or msg.type == 'note_off') and msg.note > 60:
      msg.note = 60

fr_elise.save('fr_elise_4.mid')
```

3.6 Δρομολόγηση μηνυμάτων MIDI

Σε περίπτωση που επιθυμείτε να χειριστείτε τα MIDI μηνύματα που προέρχονται από κάποιον εξωτερικό ελεγκτή, όπως ένα MIDI keyboard, ή σε περίπτωση που επιθυμείτε να στείλετε μηνύματα σε κάποια εξωτερική συσκευή, θα πρέπει να εκτελέσετε τον κώδικά σας τοπικά στον υπολογιστή σας και όχι στο Google Colaboratory. Για το σκοπό αυτό, μπορείτε να κατεβάσετε και να εγκαταστήσετε το περιβάλλον Visual Studio Code το οποίο παρέχει επεξεργαστή κειμένου για τον κώδικα καθώς και terminal για την απευθείας εκτέλεση του κώδικα.



Εικόνα 3-1: Το περιβάλλον Visual Studio Code

Όπως φαίνεται στην Εικόνα 3-1, το περιβάλλον αυτό σας επιτρέπει να ανοίξετε ένα φάκελο (File→ Open Folder από το μενού) και να περιηγηθείτε στα αρχεία του στο αριστερό panel, να δείτε και να επεξεργαστείτε τον κώδικα στο κεντρικό panel, να τον εκτελέσετε με το κουμπί play πάνω δεξιά και να δείτε τι εκτυπώνετε στην έξοδο από το terminal στο panel κάτω από τον κώδικα.

Προσοχή: Όπως και στο Google Colaboratory, έτσι και εδώ θα πρέπει να χρησιμοποιήσουμε το terminal για να εγκαταστήσουμε τις απαραίτητες βιβλιοθήκες της Python,όπως τη mido, π.χ.:

pip install mido

3.6.1 Ανάγνωση μηνυμάτων που καταφθάνουν από ελεγκτή

Ο παρακάτω κώδικας χρησιμοποιεί τη βιβλιοθήκη mido για να διαβάσει και να εκτυπώσει τα μηνύματα MIDI που προέρχονται από το πρώτο MIDI input port:

```
import mido


# print all available MIDI IN ports
input_ports = mido.get_input_names()
print('INPUTS ', input_ports)
if len(input_ports) > 0:
    port = mido.open_input(input_ports[0])  # open the 1st input port
    #OR use ur prt name, e.g. mido.open_input('USB Axiom 25 Port 1')


## print arriving messages
while True:
    for msg in port.iter_pending():
        print(msg)

# you can call this when done
port.close() # this will never be called in this program
```

4 Ηχητική απόδοση μηνυμάτων MIDI

Αφού έχει επιτευχθεί η επιτυχής λήψη και ερμηνεία μηνυμάτων MIDI από εξωτερική θύρα, το επόμενο βήμα είναι η μετατροπή αυτών των μηνυμάτων σε πραγματικό ήχο. Τα μηνύματα MIDI από μόνα τους δεν περιέχουν ηχητική πληροφορία, αλλά αποτελούν εντολές (π.χ. Note On, Note Off, Velocity, Control Change), οι οποίες πρέπει να ερμηνευτούν από μια γεννήτρια ήχου (synthesizer).
Η διαδικασία αυτή απαιτεί:

- Μια πηγή ηχητικών δειγμάτων (sound font)
- Έναν μηχανισμό σύνθεσης (synth engine)
- Διασύνδεση των MIDI μηνυμάτων με τη γεννήτρια ήχου

Στο κεφάλαιο αυτό εξετάζεται η χρήση αρχείων SoundFont (.sf2) και η υλοποίηση μιας γεννήτριας ήχου μέσω της βιβλιοθήκης FluidSynth.

4.1 Αναζήτηση sound font

Τα αρχεία SoundFont (.sf2) περιέχουν ηχητικά δείγματα (samples) και πληροφορίες για το πώς αυτά πρέπει να αναπαραχθούν για διαφορετικές νότες και όργανα.
Κριτήρια επιλογής SoundFont:

- Ποιότητα δειγμάτων (sample rate, bit depth)
- Πληρότητα οργάνων (General MIDI compatibility)
- Μέγεθος αρχείου (ισορροπία ποιότητας και απόδοσης)
- Άδεια χρήσης (δωρεάν ή εμπορική)

Δημοφιλείς πηγές:

- MuseScore SoundFonts
- Polyphone

Ένα General MIDI SoundFont επιτρέπει την αντιστοίχιση των MIDI καναλιών και προγραμμάτων (program change) σε συγκεκριμένα όργανα (π.χ. piano, violin, drums).
Μία καλή επιλογή είναι το αρχείο που θα βρείτε στο εδώ
https://musical-artifacts.com/artifacts/738

4.2 Περιεχομένα αρχείου sf2

Το αρχείο SoundFont (.sf2) έχει συγκεκριμένη εσωτερική δομή που οργανώνει τα ηχητικά δεδομένα και τις παραμέτρους αναπαραγωγής.
Βασικά δομικά στοιχεία:

- **Samples:** Ακατέργαστα ηχητικά δείγματα (PCM δεδομένα)
- **Instruments:** Ομάδες samples που αντιστοιχούν σε ένα μουσικό όργανο
- **Presets:** Ανώτερο επίπεδο που συνδέει instruments με MIDI προγράμματα

Επιπλέον παράμετροι:

- Envelope (ADSR)
- Loop points
- Filter settings
- Modulation (LFOs)

Παράδειγμα λειτουργίας:
Όταν λαμβάνεται ένα MIDI μήνυμα:

- Note On (π.χ. C4)
- Το preset εντοπίζει το κατάλληλο instrument
- Το instrument επιλέγει το σωστό sample
- Το sample αναπαράγεται με βάση velocity και envelope

4.3 Ανάγνωση αρχείου sf2 με sf2utils

Για την ανάλυση και εξερεύνηση ενός αρχείου SoundFont σε επίπεδο κώδικα, μπορεί να χρησιμοποιηθεί το πακέτο sf2utils, το οποίο επιτρέπει την πρόσβαση στη δομή του αρχείου. Για την εγκατάσταση του πακέτου εκτελέστε την ακόλουθη εντολή στο terminal:

```
pip install sf2utils
```

Για την επισκόπηση του αρχείου sf2, εκτελέστε τον ακόλουθο κώδικα της Python:

```
from sf2utils.sf2parse import Sf2File

# Open the SoundFont file
with open('FluidR3_GM.sf2', 'rb') as f:
    sf2 = Sf2File(f)

print(f"SoundFont Name: {sf2.info.bank_name}")
print("-" * 40)
print(f"{'Bank':<6} | {'Preset':<8} | {'Instrument Name'}")
print("-" * 40)

# Iterate through all presets in the file
for preset in sf2.presets:
    # We filter out 'Global' presets which aren't playable instruments
    if preset.name != 'EOP':
        print(f"{preset.bank:<6} | {preset.preset:<8} | {preset.name}")
```

Άσκηση: εκτελέστε τον παραπάνω κώδικα και ερμηνεύστε το αποτέλεσμα

4.4 Υλοποίηση γεννήτριας ήχου με FluidSynth

Η βιβλιοθήκη FluidSynth χρησιμοποιείται ευρέως για την ηχητική απόδοση MIDI αρχείων/μηνυμάτων με χρήση SoundFonts. Η FluidSynth είναι υλοποιημένη στη γλώσσα προγραμματισμού C, γεγονός που επιτρέπει υψηλή απόδοση και χαμηλή καθυστέρηση κατά τη σύνθεση ήχου σε πραγματικό χρόνο.
Βασικά χαρακτηριστικά:

- Υποστήριξη SoundFont (.sf2)
- Real-time σύνθεση
- MIDI input/output
- Χαμηλή καθυστέρηση (low latency)

Για να εγκαταστήσουμε τη FluidSynth στον υπολογιστή μας πρέπει, ανάλογα με το περιβάλλον μας να ακολουθήσουμε τις εξής οδηγίες:

- Windows/Android FluidSynth Releases

- MacOS/Linux Distributions
- Building from Source

Για χρήση σε υψηλού επιπέδου γλώσσες, όπως η Python, παρέχονται κατάλληλα bindings που επιτρέπουν την πρόσβαση στη λειτουργικότητα της βιβλιοθήκης. Το python binding που χρησιμοποιούμε εδώ είναι το PyFluidSynth. Για την εγκατάσταση του εκτελέστε την ακόλουθη εντολή στο terminal:

```
pip install pyfluidsynth
```

Έπειτα μπορείτε να εκτελέσετε το ακόλουθο παράδειγμα κώδικα:

```
import time
import fluidsynth

fs = fluidsynth.Synth()
fs.start()

sfid = fs.sfload("FluidR3_GM.sf2")
fs.program_select(0, sfid, 0, 0)

fs.noteon(0, 60, 127)
fs.noteon(0, 67, 127)
fs.noteon(0, 76, 127)

time.sleep(1.0) # wait for 1 sec

fs.noteoff(0, 60)
fs.noteoff(0, 67)
fs.noteoff(0, 76)

time.sleep(1.0)

fs.delete()
```

5 Music21 - A toolkit for computer-aided musicology

5.1 Γενική περιγραφή

Το music21 είναι ένα πακέτο της Python για επεξεργασία και σύνθεση μουσικής. Αναπτύχθηκε από ερευνητές του MIT και κυκλοφόρησε για πρώτη φορά το 2008. Χρησιμοποιείται για την ανάλυση δομών δεδομένων, την έρευνα και τη διδασκαλία γύρω από την μουσική.

Συνοπτικά, το music21 υποστηρίζει τις ακόλουθες λειτουργίες:

- Ανάγνωση και εγγραφή αρχείων μουσικού περιεχομένου (Midi, Musicxml, **kern, ABC).
- Διαθέτει μεγάλη συλλογή αρχείων (corpus) που μπορούν να χρησιμοποιηθούν για μουσικολογική έρευνα.
- Δημιουργία μουσικών παραδειγμάτων (μελωδίες, συγχορδίες).
- Σύνθεση μουσικής.
- Επεξεργασία και απόδοση παρτιτούρας.

Το music21 μπορεί να εγκατασταθεί στον υπολογιστή σας ως πακέτο τη Python. Για να αποφύγουμε προβλήματα που σχετίζονται με το λειτουργικό σας σύστημα, στο μάθημα θα υλοποιήσουμε όλα τα παραδείγματα στο Google Colaboratory, όπως περιγράφεται στην ακόλουθη ενότητα.

5.2 Οδηγίες Χρήσης

5.2.1.1 Επιπρόσθετα για τη χρήση του music21

Για την λειτουργία του music21 αμιγώς σε περιβάλλον Google Colaboratory πρέπει πρώτα να εγκατασταθούν δυο πακέτα που θα μας βοηθήσουν στη γραφική απόδοση και την ηχητική απόδοση του μουσικού περιεχομένου. Ειδικότερα για γραφική απόδοση παρτιτούρας (SVG Rendering) θα χρησιμοποιήσουμε το πακέτο lilypond, ενώ για ηχητική απόδοση θα χρησιμοποιηθεί το πακέτο fluidsynth. Για την εγκατάσταση των δύο πακέτων θα πρέπει να εισάγετε τις ακόλουθες εντολές σε ένα code block του Google Colaboratory:

```
# install lilypond for score svg rendering
!apt-get install lilypond > /dev/null

# install fluidsynth for sound rendering
!apt-get install fluidsynth > /dev/null

# copy sound bank to use for sound rendering
!cp /usr/share/sounds/sf2/FluidR3_GM.sf2 ./font.sf2
```

Για να εισαχθούν στο Google Colaboratory τα πακέτα της Python με τα οποία θα δουλέψουμε θα πρέπει επιπρόσθετα, να εισάγετε τις ακόλουθες γραμμές κώδικα:

```
import music21
from music21 import *
import numpy as np
from IPython.display import Image, Audio
```

Έχοντας εγκαταστήσει τα παραπάνω πακέτα, θα πρέπει να συμπεριληφθούν δύο συναρτήσεις, η πρώτη για γραφική και η δεύτερη για ηχητική απόδοση της παρτιτούρας:

```
# Define rendering functions

def renderNotation(score):
  display(Image(str(score.write('lily.png'))))
```

```
def renderSound(score):
  filename = score.write('mid')
  !fluidsynth -ni font.sf2 $filename -F $filename\.wav -r 44100 > /dev/null
  display(Audio(filename + '.wav'))
```

Τέλος για να έχουμε πρόσβαση από το αρχείο της Python σε άλλα αρχεία του Google Drive, θα πρέπει να εισάγουμε το εξής απόσπασμα κώδικα:

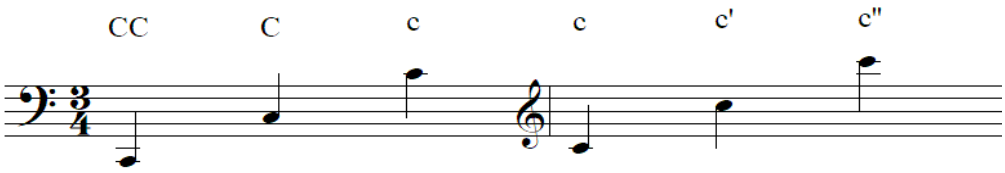
```
from google.colab import drive
drive.mount('/content/gdrive')
%cd /content/gdrive/MyDrive/DMR
```

5.3 Σύνθεση Παρτιτούρας

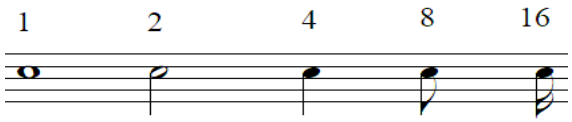
Υπάρχουν διάφοροι τρόποι για τη δημιουργία παρτιτούρας στο music21. Ενδεικτικά αναφέρονται παρακάτω η δημιουργία παρτιτούρας με TinyNotation, μέσω stream, καθώς και με χρήση συγχορδιών.

5.3.1 TinyNotation

Το TinyNotation είναι ένας εύκολος και γρήγορος τρόπος να συνθέσει κανείς μουσική μέσω της music21. Χρησιμοποιεί ειδικούς κανόνες και η εντολή μπορεί να δοθεί σε μία μόνο γραμμή.
Συγκεκριμένα, ορίζουμε πρώτα το μέτρο και έπειτα το τονικό ύψος και τη διάρκεια (αξία) της κάθε νότας. Ο χωρισμός σε μέτρα καθώς και το κλειδί εισάγονται αυτόματα.



Εικόνα 4-1: Προσδιορισμός τονικού ύψους στο TinyNotation.

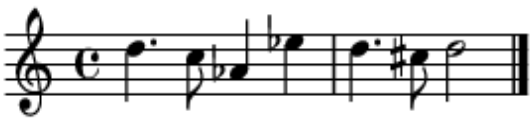


Εικόνα 4-2: Προσδιορισμός χρονικής αξίας στο TinyNotation.

Παράδειγμα:

```
s = converter.parse("tinyNotation: 4/4 d'.4 c'8 a-4 e'- d'.4 c'#8 d'2")
renderNotation(s)
renderSound(s)
```

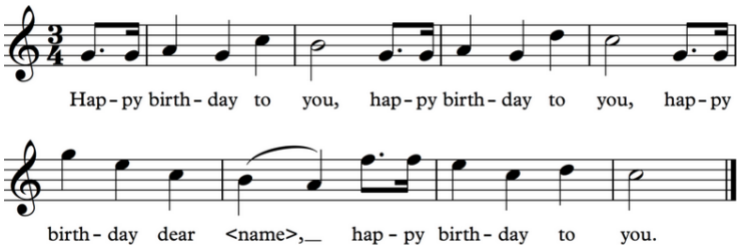
Αποτέλεσμα:



Ειδικά σύμβολα:

‘ . ’ = dotted note (παρεστιγμένη νότα)
‘ # ‘ = sharp (δίεση)
‘ – ‘ = flat (ύφεση)
‘~’ = slur (σύζευξη)

Εργασία Αυτό-εξάσκησης:
Να γράψετε σε TinyNotation τη μελωδία του Happy Birthday:



Εικόνα 4-3: Η μελωδία του Happy Birthday.

5.3.2 Απόδοση Μουσικού Περιεχομένου (Rendering)

Πέρα από τη γραφική αναπαράσταση και την ηχητική απόδοση μουσικού περιεχομένου, στο πακέτο music21 υπάρχει η δυνατότητα απόδοσης σε κείμενο και pianoroll. Για παράδειγμα δοκιμάστε τα εξής:

```
s = converter.parse("tinyNotation: 4/4 d'.4 c'8 a-4 e'- d'.4 c'#8 d'2")

# Σε σημειογραφία (SVG Rendering)
renderNotation(s)

# Σε ήχο (Sound Rendering)
renderSound(s)

# Σε κείμενο
s.show('text')

# Σε pianoroll
s.plot('pianoroll')
```

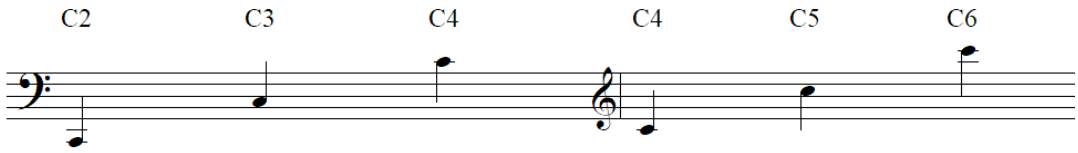
5.3.3 Stream

Εναλλακτικά μπορεί κανείς να συνθέσει μια σειρά από νότες εισάγοντας τα στοιχεία μιας παρτιτούρας ένα προς ένα, σε μία κενή ροή (stream). Η λειτουργία των stream στο music21 περιγράφεται στο κεφάλαιο 4 και το κεφάλαιο 6 των οδηγιών χρήσης (documentation) του music21.

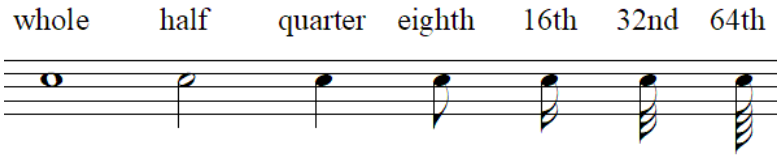
Στην αρχή ορίζονται τα κλειδιά (clefs), έπειτα η κλίμακα (key signature), μετά το μέτρο (time signature) και τέλος όλες οι νότες, οι συγχορδίες ή οι παύσεις.

Αναφορικά με τα κλειδιά, η κλίση **`clef.TrebleClef()`** αντιστοιχεί στο κλειδί του Σολ, ενώ **`clef.BassClef()`** αντιστοιχεί στο κλειδί του Φα. Η κλίμακα καθορίζεται με βάση την τονική μουσική και το πλήθος διέσεων/υφέσεων. Για παράδειγμα **`key.KeySignature(1)`** αντιστοιχεί σε κλίμακα της τονικής μουσικής με μια δίεση (κλίμακα Σολ) με βάση τη μουσική θεωρία. Ενώ το μέτρο (**meter.TimeSignature**) δίνεται ως κλάσμα όπως στο ακόλουθο παράδειγμα.

Σε αντίθεση με το tinyNotation, στις μεμονωμένες νότες και τις συγχορδίες τα τονικά ύψη ορίζονται με αριθμό οκτάβας και οι αξίες με την διάρκειά τους, όπως και στην μουσική θεωρία.



Εικόνα 4–4: Προσδιορισμός τονικού ύψους στην ροή.

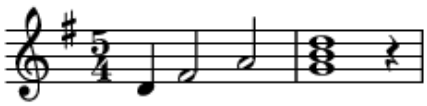


Εικόνα 4-5: Προσδιορισμός χρονικής αξίας στην ροή.

Περισσότερες λεπτομέρειες για τα τονικά ύψη και τις διάρκειες στα κεφάλαια 2 και 3 του των οδηγιών.

Παράδειγμα:

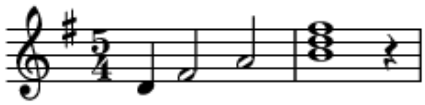
```
# Ορίζουμε κλειδί
c1 = clef.TrebleClef()

# Ορίζουμε κλίμακα
k1 = key.KeySignature(1)

# Ορίζουμε μέτρο
t1 = meter.TimeSignature('5/4')

# Ορίζουμε νότες/συγχορδίες/παύσεις
n1 = note.Note('D4', type='quarter')
n2 = note.Note('F#4', type='half')
n3 = note.Note('A4', type='half')
n4 = chord.Chord('G4 B4 D5', type='whole')
n5 = note.Rest(type='quarter')

# Φτιάχνουμε το stream παραθέτοντας τα παραπάνω στοιχεία
s = stream.Stream([c1, k1, t1, n1, n2, n3, n4, n5])

renderNotation(s)
renderSound(s)
```



5.3.3.1 Συγχορδίες (Chords) και επεξεργασία τους

Αντίστοιχα με τις νότες και τις παύσεις, σε μία ροή (stream) υπάρχει η δυνατότητα προσθήκης συγχορδιών, όπως φαίνεται στο στοιχείο n4 του προηγούμενου παραδείγματος. Σε μια συγχορδία μπορούν να προστεθούν και να αφαιρεθούν νότες, όπως στο ακόλουθο παράδειγμα. Προσέξτε ότι η συγχορδία στο δεύτερο μέτρο είναι διαφορετική από ότι στο προηγούμενο παράδειγμα.

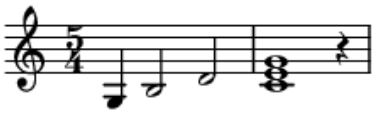
```
n4.add('F#5')
n4.remove('G4')
renderNotation(score)
```



5.3.3.2 Μετατροπία (Transposition)

Μια ροή μπορεί να υποστεί μετατροπία δηλαδή να μετακινηθεί σε άλλη κλίμακα. Αυτό σημαίνει ότι θα αλλάξει τόσο ο οπλισμός (key signature), όσο και οι νότες. Ορίζουμε την μετατροπία ανάλογα με το πόσα ημιτόνια θα μετακινηθούμε προς τα πάνω ή προς τα κάτω από την αρχική θέση. Υπάρχει δυνατότητα να γίνει μετατόνιση όχι μόνο σε ολόκληρη τη σύνθεση αλλα και σε κάθε στοιχείο (νοτα, συγχορδία, κλίμακα) ξεχωριστά.

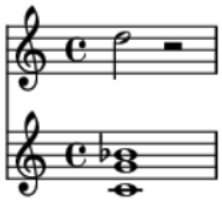
```
score2 = score.transpose(-7)
renderNotation(score2)
```



5.3.4 Score

Στη μουσική ως παρτιτούρα (score) αναφέρεται η συμβολική γραφή ενός μουσικού έργου που διαθέτει πολλές πάρτες (parts), δηλαδή πολλά μουσικά όργανα. Στο ακόλουθο παράδειγμα αποτυπώνεται η χρήση μιας παρτιτούρας με δύο πάρτες, κι επομένως δύο πεντάγραμμα.

```
score = stream.Score() #create an empty score


p1 = stream.Part()
p1.append(clef.TrebleClef())
p1.append(note.Note('D5', type='half'))  # half note
p1.append(note.Rest(quarterLength=2.0))  # half rest
p1.id = "Flute"

p2 = stream.Part()
p2.append(clef.TrebleClef())
p2.append(chord.Chord('C4 G4 B-4', type= 'whole'))
p2.id = 'Piano'

score.append([p1, p2])


renderNotation(score)
renderSound(score)
```



5.4 Διάθεση Αρχείων από το music21. corpus

Το πακέτο music21 διαθέτει μεγάλο αρχείο από έργα συνθετών σε μορφές musicxml, kern, abc τα οποία μπορούν να χρησιμοποιηθούν για μουσικολογική ανάλυση και έρευνα. Τα ονόματα των συνθετών που μπορούμε να αναζητήσουμε έργα τους βρίσκονται στον παρακάτω σύνδεσμο (link).
Με το επόμενο κομμάτι κώδικα μπορούμε να δούμε ποιά έργα και σε ποιά μορφή υπάρχουν για τον εκάστοτε συνθέτη.

```
paths = corpus.getComposer('beethoven')
paths
```

Το προηγούμενο κομμάτι κώδικα θα δώσει μία λίστα αρχείων μουσικής σημειογραφίας, για παράδειγμα:

```
PosixPath('/usr/local/lib/python3.7/dist-packages/music21/corpus/beethoven/opus132.mxl'),
 PosixPath('/usr/local/lib/python3.7/dist-packages/music21/corpus/beethoven/opus133.mxl'),
 PosixPath('/usr/local/lib/python3.7/dist-packages/music21/corpus/beethoven/opus18no1/movement1.krn'),
 PosixPath('/usr/local/lib/python3.7/dist-packages/music21/corpus/beethoven/opus18no1/movement1.mxl'),
 PosixPath('/usr/local/lib/python3.7/dist-packages/music21/corpus/beethoven/opus18no1/movement2.krn'),
```

Κ.λπ.

5.4.1 Φόρτωση αρχείου

Για να φορτώσουμε κάποιο αρχείο από το corpus εισάγουμε την εντολή:

```
#read the opus132 beethoven work
s = corpus.parse('beethoven/opus132.mxl')
```

Προσέξτε ότι η διαδρομή (path) για το συγκεκριμένο αρχείο δίνεται από τη λέξη corpus κι έπειτα.
Στο ακόλουθο παράδειγμα κώδικα, εξάγουμε πληροφορίες για το περιεχόμενο του αρχείου. Ειδικότερα, για κάθε πάρτη εκτυπώνεται το όνομά της καθώς και το πόσα μέτρα περιλαμβάνει. Στη συνέχεια, αποδίδεται γραφικά και ηχητικά ένα απόσπασμα τεσσάρων μέτρων της παρτιτούρας, ειδικότερα από το μέτρο 200 έως και το 203.

```
for p in s.parts:
  print("Part ", p.id, " has ", len(p.measures(0, None)), " measures")

#s.show('text')
#s.plot('pianoroll')

#render four measures
renderNotation(s.measures(200, 203))
renderSound(s.measures(200, 203))
```

Οι δύο πρώτες γραμμές δίνουν το εξής αποτέλεσμα:

```
Part  Violin I  has  2300  measures
Part  Violin II  has  2117  measures
Part  Viola  has  2122  measures
Part  Violoncello  has  2069  measures
```

Που σημαίνει ότι πρόκειται για ένα κουαρτέτο εγχόρδων.

5.4.2 Επιλογή αποσπάσματος και επεξεργασία

Επειδή το έργο που επιλέχθηκε για διάβασμα στο music21 είναι αρκετά μεγάλο, θα επιλέξουμε ένα απόσπασμα, στο οποίο θα εφαρμόσουμε κάποιες επεμβάσεις:

```
#choose excerpt
excerpt = s. measures(200, 203)
#transpose the first note of the second part
excerpt.parts[1].pitches[0].transpose(3, inPlace=True)
#transpose the entire excerpt by one fifth down
excerpt.transpose(-7, inPlace=True)
renderNotation(excerpt)
renderSound(excerpt)
```

Στο παραπάνω απόσπασμα αλλάζει ο τόνος της πρώτης νότας της δεύτερης πάρτης κι έπειτα γίνεται μια μετατροπία σε όλο το απόσπασμα, μια 5η καθαρή (7 ημιτόνια) κάτω

```
p.transpose(4, inPlace=True)
print(p)
```

5.5 Εγγραφή και ανάγνωση αρχείων στη music21

Μέσω του music21 μπορεί να γίνει εγγραφή και ανάγνωση αρχείων διαφόρων τύπων όπως MIDI και musicxml. Περισσότερες πληροφορίες για τους τύπους αρχείων που είναι συμβατοί για την εισαγωγή είτε την εξαγωγή βρίσκονται εδώ.

Για παράδειγμα, το ακόλουθο κομμάτι κώδικα αποθηκεύει το απόσπασμα της προηγούμενης ενότητας σε αρχείο MIDI:

```
excerpt.write('midi', 'composition.mid')
```

Το πρώτο όρισμα της συνάρτησης write αφορά στον τύπο του αρχείου (π.χ. musicxml, midi), ενώ το δεύτερο το όνομα του αρχείου. Για την ανάγνωση ενός αρχείου, μπορεί να χρησιμοποιηθεί η εντολή converter.parse() όπως στα προηγούμενα.

6 Ασκήσεις

6.1 Εισαγωγή στο πακέτο Mido

Να δημιουργήσετε ένα αρχείο MIDI με τον online midi editor το οποίο να περιέχει την κλίμακα Ντό Μείζονα (C-major).
Διαβάστε το αρχείο αυτό μέσω του πακέτου της Python Mido.

1. Τι παρατηρείτε αναφορικά με το tempo και το ρυθμο;
2. Δημιουργήστε ένα ίδιο αρχείο με το MuseScore. Τι διαφορές παρατηρείτε;

6.2 Δημιουργία και επεξεργασία απλού αρχείου MIDI

1. Να δημιουργήσετε ένα αρχείο Midi με το πακέτο mido που να περιέχει την κλίμακα ντό μείζονα.
2. Επεξεργαστείτε το αρχείο αυτό αλλάζοντας του το tempo σε 80bpm και τις νότες ώστε να περιλαμβάνουν την Ντό ελάσσονα.
3. Κάντε μία μετατροπία στη Ρε ελάσσονα και σώστε σε νέο αρχείο.

Και τα τρία παραπάνω αρχεία να αποθηκευθούν στο google drive, να κατέβουν στο τοπικό σύστημα αρχείων και να επιθεωρηθούν το με το MuseScore και τον Online Midi Editor.

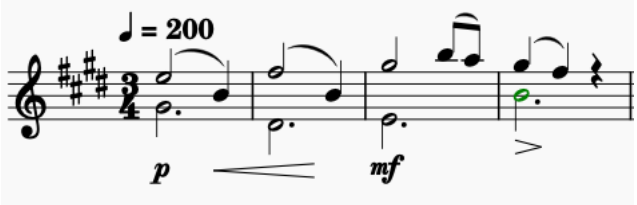
6.3 Μετρονόμος

1. Να υλοποιήσετε έναν μετρονόμο με το πακέτο mido, ο οποίος θα ζητάει από το χρήστη τέμπο, ρυθμό και αριθμό μέτρων και θα παράγει ένα click_track με τους ήχους του μετρονόμου.
2. Να προσθέσετε στο αρχείο μια μελωδία που θα αντιστοιχεί στην κλίμακα ντό μείζονα
3. Να αλλάξετε το ρυθμό σε 3/4 με το πρώτο beat να διαρκεί δύο τέταρτα και το τρίτο ένα τέταρτο
4. Να κάνετε μία μετατροπία κατά ένα διάστημα τετάρτης προς τα κάτω.

Σε κάθε βήμα επιθεωρήστε το αποτέλεσμα στο MuseScore ή στον Online MIDI Editor.

6.4 Πολυφωνικό και πολυχρωματικό αρχείο

Με το πακέτο Mido, να δημιουργήσετε ένα αρχείο MIDI που να αποδίδει την ακόλουθη παρτιτούρα:



1. Χρησιμοποιήστε ένα MidiTrack για όλες τις νότες
2. Χρησιμοποιήστε ξεχωριστά tracks για κάθε φωνή
3. Χρησιμοποιήστε ξεχωριστά μουσικά όργανα για κάθε φωνή επιλέγοντας από τον Πίνακα των ηχοχρωμάτων του General Midi

6.5 Παράμετροι συνεχούς μεταβολής

Να δημιουργήσετε ένα αρχείο MIDI το οποίο να περιλαμβάνει μία νότα μεγάλης διάρκειας που αναπαράγεται από ένα μουσικό όργανο ελεγχόμενης διάρκειας, δηλαδή ένα μουσικό όργανο με ελεγχόμενο sustain (π.χ. βιολοντστέλο, τρομπέτα, κ.λ.π.)

Κατά τη διάρκεια αυτής της νότας να εφαρμόσετε ένα pitch bend από τη μικρότερη στην υψηλότερη τιμή του pitch bend καθώς και ένα ταυτόχρονο panning (contoller number 10) από τη θέση τέρμα αριστερά στη θέση τέρμα δεξιά.

6.6 Κατανομή τονικού ύψους

Χρησιμοποιήστε κάποιο από τα αρχεία MIDI που θα βρείτε στο Έγγραφα->MIDI Files ή και άλλα που θα βρείτε στο 8notes.com, ώστε:

- να υπολογίσετε το πλήθος των νοτών, διαχωρίζοντας το όνομα τους (Pitch Class) από την οκτάβα στην οποία βρίσκονται
- Να υπολογιστεί και να αναπαρασταθεί γραφικά το **Pitch Class Profile (PCP)** ενός αρχείου MIDI. Τι μας λέει το PCP για την τονικότητα του κομματιού?

Ορισμοί:

- **Pitch Class ή Chroma:** Στην ανάλυση μουσικού περιεχομένου ο όρος αυτός αναφέρεται στη νότα του συγκερασμένου συστήματος όταν δε λαμβάνεται υπόψη η οκτάβα. Έτσι στη δυτική μουσική υπάρχουν 12 pith classes,

```
chroma = ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B']
```

- **Pitch Class Profile**: Ο όρος αναφέρεται στην κατανομή τιμών χρώματος για ένα μουσικό κομμάτι. Δηλάδή σε ποιό ποσοστό οι νότες του αντιστοιχούν στο χρώμα C, σε ποιό ποσοστό C# κ.ο.κ