Português (Portugal) → Inglês

Informatics for Musicology (IPM) 2024/25

Jupyter Notebooks

Teacher: JN Oliveira

Department of IT at U. Minho, in collaboration with ENSICO

Class of 22-Oct:

Continued exploration of IPM (Haskell) libraries for 'Computer Aided Musicology'. The operations `take` , `drop` and the corresponding temporal versions `dtake` and `ddrop` .

The role of *imitation* in music. Case study: construction of the *Canon per 3 Violini e Basso* by Johann Pachelbel (1653-1706) from the first violin part and the infinite bass sequence ('ostinato').

Manipulation of infinite sequences and their importance in music.

⚠️ **Important** : run without moving the next cells.

In [ ]:

```
: opt  no – lint
: m  Data . Char
: m  Date . List
: m  Date . List . Split
: m  Data . Ratio
```

Modules developed for the discipline:

In [ ]:

```
: l  ../ src / Cp . hs
: l  ../ src / Reducer . hs
: l  ../ src / Ipm . hs
: l  ../ src / Abc . hs
```

Data ("case studies"):

In [ ]:

```
: l  ../ src / CS . hs
```

🔴The functions `take` and `drop`

6.1 - Back to the names of all the students in this class,

In [ ]:

```
class   =  [
    "Ana Bárbara Francisco Gabriel" ,
    "Dinis Cunha Andrade" ,
    "Inês Beatriz Martins Neves" ,
    "João Jorge Soares Moreira" ,
    "João Henrique Mestre Conceição Inácio" ,
    "João Miguel Pereira de Oliveira" ,
    "Matilde Sampaio Teixeira " ,
    "Mohammad Najib Angar" ,
    "Miguel Pires Santiago"
    ]
```

Evaluate the following expressions and draw conclusions:

In [ ]:

```
take  3  class
```

In [ ]:

```
drop  4  class
```

In [ ]:

```
map  ( take  2 )  class
```

In [ ]:

```
map  ( take  2  .  drop  2 )  class
```

6.2 - Write an expression in the cell below that gives the following result:

```
[[ "Ana" , "Bárbara" ],[ "Dinis" , "Cunha" ],[ "Inês" , "Beatriz" ],[ "João" , "Jorge" ],[ "João"
, "Henrique" ] ,[ "João" , "Miguel" ],[ "Matilde" , "Sampaio" ],[ "Mohammad" , "Najib" ],[
"Miguel" , "Pires" ]]
```

In [ ]:

```

```

In short :

Designation	Meaning	Detailed description
`take i`	get prefix	gives the first i -elements of the sequence
`drop i`	get suffix	eliminates the first i -elements of the sequence

6.3 - As we have just seen, functions `drop` and `take` are complementary but not inverse to each other. Anticipate the result of applying the functions

```
f  =  ( drop  3 )  .  ( take  3 )
g  =  ( take  3 )  .  ( drop  3 )
```

to the list

```
x  =  [ 1..10 ]
```

Confirm your predictions by running tests on the next cell.

In [ ]:

```

```

6.4 - *(Consolidation)* Use `take` and/or `drop` to select the last 10 notes from `carnaval_serrano` :

In [ ]:

```

```

🔴The functions `dtake` and `ddrop`

Compare what was said above with:

Designation	Meaning	Detailed description
`dtake`	get prefix by duration	`dtake d m` will fetch as many notes as possible and `m` even predict the duration `d`
`ddrop`	get suffix by duration	`ddrop d m` Search for notes that `dtake d m` you did not select

6.5 - Check the differences by running the cells:

In [ ]:

```
take 2 carnival_serrano
```

In [ ]:

```
dtake 2 carnival_serrano
```

6.6 - The next cell shows us the first 10 bars of the 1st violin part of the *Canon per 3 Violini e Basso* by Johann Pachelbel (1653-1706).

In [ ]:

```
abcPlayM "D" "C" ( dtake 10 v1 )
```

Create cells to calculate the following results:

- the total number of banknotes `v1`
- the total duration of `v1`
- the first 10 bars in retrograde motion
- bars 7 to 9 (inclusive) of `v1`

(Use `abcPlay` etc where applicable.)

6.7 - Remembering the previous class, what should we write in the next cell to obtain the *motifs* of the first 200 notes of this melody?

In [ ]:

```

```

6.8 - ➡ Construction of the canon: Knowing that the second violin responds with a delay of 2 bars, define `v2` in the next cell (only 12 bars to *save* Jupyter...)

In [ ]:

```
v2 = undefined
( abcPlayM "D" "C" . dtake 12 ) ( v1 # v2 )
```

Using `abcShow` (etc) to listen:

In [ ]:

```
abcShow
```

6.9 - Now add the third violin, knowing that it also responds to the second with a delay of 2 measures:

In [ ]:

```

```

6.10 - After viewing the bass *motif from the same Canon per 3 Violini e Basso* ,

In [ ]:

```
abcPlayM "D" "C" low
```

evaluate the next cell and draw conclusions:

In [ ]:

```
v4 = low ++ v4
abcPlayM "D" "C" ( take 24 v4 )
```

6.11 - Perform the following expressions:

In [ ]:

```
take 12 v4
```

In [ ]:

```
take 120 v4
```

In [ ]:

```
take 1200 v4
```

What can you say about `v4` ?

6.12 - Since the bass `v4` , as we saw above, is the *ad eternum* repetition of the 8 notes of `bass` , we have to use it `dtake` to indicate how many bars we want from the entire canon. Based on the number of measures calculated above, what is the value of `n` writing it in the cell below to be `v1` complete?

In [ ]:

```
n = undefined
final = P [ v1 , v2 , v3 , v4 ]
( abcPlayM "D" "C" . dtake n ) final
abcShow
```

🔴Infinite sequences

`v4` above is an example of an *infinite* sequence .

How to define such sequences and manipulate them? We can't show them because they are... infinite, they never finish being shown.

Let's start by remembering

Designation	Meaning	Detailed description
`(++)`	junction	`x ++ y` joins the two sequences `x` into `y` one
`(:)`	affix	`a:x` It's the same thing as joining `[a] ++ x`

6.13 - Write expressions that capture the following situation:

- `x` is the sequence `[10..12]` (define in the cell below)
- `y` is the sequence in which the number is **affixed** (idem) `0` `x`
- `z` is the sequence in which the number is **affixed** (idem) `1` `y`

In [ ]:

```
x  =  undefined
y  =  undefined
z  =  undefined
---
x
y
z
```

6.14 - As above, write expressions that capture the following situation:

- `x` is the rhythmic sequence `[3%4,3%4,3%4]`
- `y` is the sequence that sets the duration `1%4` to be `x`
- `z` is the sequence that sets the duration `1%2` to be `y`

In [ ]:

```
x  =  undefined
y  =  undefined
z  =  undefined
---
x
y
z
```

6.15 - Finally, write expressions that capture the following situation:

- `tern` is the rhythmic sequence that sets the duration `3%4` to be `tern`

In [ ]:

```
tern  =  3 % 4  :  tern
```

Calculate `take 10 tern` and `take 1000` . Any other 'takes' will give results, as `tern` it is the infinite sequence `[3%4,3%4,3%4,3%4,3%4,...]`

6.16 - Analyze the following definitions:

```
x  =  [ 1 , 2 ]  ++  y
y  =  [ 2 , 1 ]  ++  x
```

What can you say about the sequences `x` and `y` ?

In [ ]:

```
```