# Informatics for Musicology (IPM) 2024/25

## Jupyter Notebooks

## Teacher: JN Oliveira

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

---

### **8-Oct Class** : Haskell in Jupyter Notebook

Sequence manipulation in the Haskell language.

From numbers and words to musical phrases. The operation of gluing ("zipping") two sequences. Pair sequences.

Sequences that represent music: representation of sound events by pairs (pitch, duration). Interaction with Abc notation.

Case study: pentatonic melody 'Carnaval serrano' (Peru-Bolivia), source: Flutes from the Andes, by Guillermo De La Roca. Epm Music B000027YVO.

Chain of operations (compositionality).

---

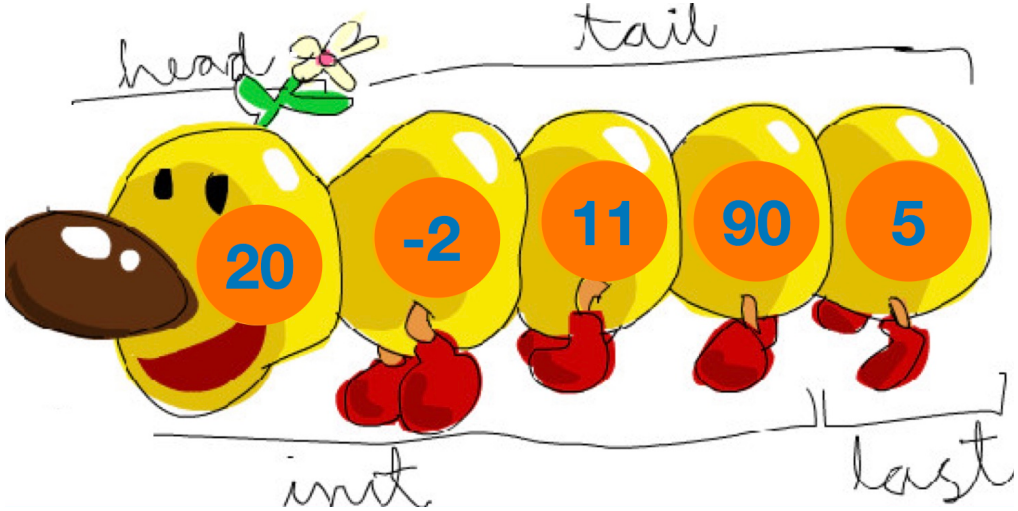
⚠️ **Important** : run without moving the next cells.

In [1]:
```
: opt  no - lint
: m  Data . Ratio
: m  Data . Char
: m  Date . List
```

In [2]:
```
: l  ../ src / Cp . hs
: l  ../ src / Reducer . hs
: l  ../ src / Ipm . hs
: l  ../ src / Abc . hs
: l  ../ src / CS . hs
```

---

## 🔴 Sequences in general




### ➡️ The functions of the " *centipede* "

**3.1** - After observing the figure, evaluate the following expressions and draw conclusions:

In [ ]:
```
centipede  =  [ 20 , - 2 , 11 , 90 , 5 ]
---
centipede
```

In [ ]:
```
head  centipede
```

In [ ]:
```
tail  centipede
```

In [ ]:
```
last  centipede
```

In [ ]:
```
init  centipede
```

---

**3.2** - Continue, evaluating the following expressions and drawing conclusions:

In [ ]:
```
 centipede length
```

In [ ]:
```
sum  centipede
```

In [ ]:
```
9  :  centipede
```

In [ ]:
```
centipede  ++  centipede
```

In [ ]:
```
rotr  centipede
```

In [ ]:
```
rotl  centipede
```

In [ ]:
```
reverse  centipede
```

In [ ]:
```
sort  centipede
```

---

**3.3** - Above, the sequence `centopeia` was ordered.

> What we could easily do without needing a computer...

But try sorting the following list "by hand":

In [ ]:
```
long_list  =  [ 21 , 421 , 28 , 166 , 83 , 371 , 334 , 265 , 110 , 257 , 198 , 438 , 59 , 16 , 329 , 450 , 3
        112 , 61 , 45 , 242 , 8 , 432 , 296 , 122 , 360 , 48 , 92 , 116 , 30 , 434 , 103 , 389 , 500 , 446 ,
        165 , 316 , 322 , 324 , 137 , 160 , 210 , 289 , 238 , 295 , 174 , 285 , 499 , 71 , 384 , 376 , 171 ,


        437 , 283 , 109 , 217 , 321 , 204 , 341 , 299 , 355 , 323 , 288 , 278 , 185 , 202 , 404 , 170 , 239
```

And compare with doing the same on a computer:

In [ ]:
```
sort  long_list
```

---

## 🔴Computing and programming

The above operations ( `reverse` , `sort` , `sum` etc) were **programmed** to work on a **computer** for any lists, of any size.

This is what **computers** are for:

> A **computer** is a machine that "does" very **difficult** things for us...
>
> ... **quickly** ...
>
> ...without getting **bored** ...
>
> ... don't even **make a mistake** !

👉Let's take advantage of this in **computer-assisted musicology** .

---

**3.4** - The following melodic line is given:

In [ ]:
```
mel  =  [ "A" , "^G" , "A" , "B" , "c" , "B" ]
```

Let's visualize it:

In [ ]:
```
abcplease  honey
```

> *Note that it is the basic melodic cell of the theme La Folia by Arcangelo Corelli (1653-1713), here transposed from D minor to A minor .*

---

**3.5** - Show in the cell below that `mel` there are only 4 (different) notes.

> *Suggestion: just eliminate the repeated ones and count the rest...*

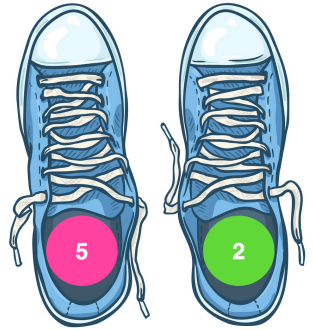
In [ ]:
```

```

---

## 🔴Pairs

We can group numbers into **pairs** , **triples** , etc. The next cell contains two pairs of numbers - run it.



In [ ]:
```
p  =  ( 5 , 2 )
q  =  ( 4 , 0 )
--
p
q
```

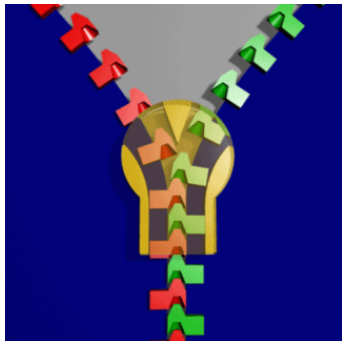
---

**3.6** - What do operations `fst` and `snd` the next cell do? Run it and respond according to what you observe as a result.

In [ ]:
```
fst  p
snd  q
```

---

🔴"Zipping"



**3.7** - Evaluate the following expressions and draw conclusions regarding the meaning of `zip` :

In [ ]:
```
zip  [ 1..5 ] centipede
```

In [ ]:
```
zip  []  centipede
```

In [ ]:
```
 centipede  zip [ 12 , 100 ]
```

In [ ]:
```
zip  [ "Maria" , "Manuel" ]  [ 1994 , 2002 ]
```

In [ ]:
```
zip  [ "Maria" , "Manuel" ]  [ 1994 , 2002 , 2020 , 1955 ]
```

---

**3.8** - Evaluate the next cell:

In [ ]:
```
r  =  [ 3 % 4 , 3 % 4 , 3 % 4 , 3 % 4 , 3 % 4 , 3 % 4 , 1 % 4 ]
m  =  zip  mel  r
---
m
```

Draw conclusions after "viewing" `m` :

In [ ]:
```
abcplease  m
```

---

**3.9** - Repeat what was done above in the next cell so that each note lasts for a minimum:

In [ ]:
```

```

---

**3.10** - What did you do `abcplease` ?

In [ ]:
```
abcShow
```

> 👉To listen: (a) copy the ABC code you generated above to the *clipboard* `abcShow` ; (b) enter the online ABC editor , paste and tap.

---

**3.11** - Evaluate the following cell and draw conclusions regarding the meaning of `unzip` :

In [ ]:
```
unzip  m
```

---

**3.12** - Back to

| Designation | Meaning | Detailed description |
|---|---|---|
| `fst` | the first | gives the first element of the pair `(a,b)` , that is `a` |
| `snd` | the second | gives the second element of the pair `(a,b)` , that is `b` |

note:

In [ ]:
```
n = ( "^G" , 3 % 4 )
```

Write expressions that calculate in the following cells:

- the number of characters in the first element of the note `n`
- the second element of this note
- double the latter.

In [ ]:

In [ ]:

In [ ]:

---

## 🔴Case study

`carnaval_serrano` A small Andean melody (Peru-Bolivia) is recorded in it . *[Source: Flutes from the Andes, by Guillermo De La Roca. Epm Music B000027YVO ]* .

Without inspecting `carnaval_serrano` , write expressions that let you know:

**3.13** - How many notes and how many different notes does this song have?

In [ ]:
```
....
```

In [ ]:
```
...
```

---

**3.14** - What is the first note?

In [ ]:
```
...
```

---

**3.15** - How long is the last note?

In [ ]:
```
...
```

---

**3.16** - And the respective sound height ( *pitch* )?

In [ ]:
```
...
```

---

**3.17** - Let's check it out by viewing (and listening) `carnaval_serrano` :

In [ ]:
```
abcplease  carnival_serrano
```

In [ ]:
```
abcShow
```

---

**3.18** - Now let's use it `unzip` to break it down `carnaval_serrano` into its melodic and rhythmic part:

In [ ]:
```
( mel , rhythm )  =  unzip  carnaval_serrano
```

---

**3.19** - What is the total duration of `carnaval_serrano` ?

In [ ]:
```
...
```

---

**3.20** - How many different notes do you have, `carnaval_serrano` ignoring the rhythmic part?

In [ ]:
```
...
```

---

**3.21** - What notes are these? (To replace `undefined` .)

In [ ]:
```
...
```

View them:

In [ ]:
```
abcplease  ....
```

Conclude that the melody is in a pentatonic scale . What type?

---

### 🔴 Operation chaining

| Designation | Meaning | Detailed description |
|---|---|---|
| `f . g` | after | apply `f` after `g` , that is: `(f . g) a = f (g a)` |
| `id` | do nothing | `id x = x` whatever `x` |

(Functional composition). For example, instead of

```
abcplease (reverse (rotl notas))
```

to write

```
(abcplease . reverse . rotl) notas
```

It's more practical.

---

**3.22** - To appreciate the chain of operations, let's start by executing the following cell, which **summarizes** everything we did above up to the visualization in score of the scale underlying `carnaval_serrano` :

In [ ]:
```
( abcplease . reverse . rotl . nub . fst . unzip ) carnaval_serrano
```

---

**3.23** - Now let's run all the steps of this expression one at a time and comment on the results:

In [ ]:
```
unzip carnaval_serrano
```

In [ ]:
```
( fst . unzip ) carnaval_serrano
```

In [ ]:
```
( nub . fst . unzip ) carnaval_serrano
```

In [ ]:
```
( rotl . nub . fst . unzip ) carnaval_serrano
```

In [ ]:
```
( reverse . rotl . nub . fst . unzip ) carnaval_serrano
```

In [ ]:
```
( abcplease . reverse . rotl . nub . fst . unzip ) carnaval_serrano
```

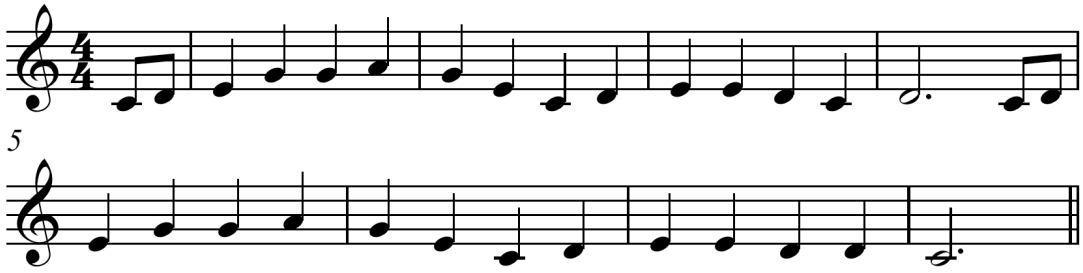
In [ ]:
```
scale = abcplease . reverse . rotl . nub . fst . unzip
```

In [ ]:
```
carnival_serrano scale
```

---

**3.24** - Consolidation:

> (a) Define the sequence of musical events of the well-known melody Oh! Susanna shown below.
>
> (b) Derive from it the scale in which it is composed, as was done for `carnaval_serrano` .




In [ ]:
```
susana = undefined
---
abcplease susana
```

---