Språkide - Codename Songic

Ett DSL för att snabbt kunna skapa sångstruktur med hjälp av motiv. För motiven finns funktionalitet för att lätt ändra enstaka element samt helt kunna ändra hela motivet samtidigt. Exempelvis genom att höja alla noter i motivet ett steg för varje iteration motivet spelas.

Den minsta byggstenen är en not som består av:

1. frekvens (oktav + not)
2. takt_längd (ex 1/4 dels takt)

Noter kan läggas ihop för att skapa motiv. Motiven tillsammans bildar ett songstycke och flera songstycken bildar en låt.

4C3 - En fjärdedel lång tonhöjd C i tredje oktaven.
2C#2 - En halv lång tonhöjd C höjt ett halvt steg i andra oktaven.
D - En hel tonhöjd D i första oktaven (eller något användardefinierat default värde).

```
1    motiv = 4C3.repeat(2)  # Spelar noten två gånger.
2    vers = motiv.transpose(1).repeat(3)  # Höjer tonen 1 steg och spelar motivet tre gånger.
3    C == B#   # Kommer ge sant.
```

Vi tänker oss initialt separat grammatik för noter respektive ackord då även om ackord består av noter så beter de sig lite annorlunda. En egen grammatik för ackorden kommer först skapas när vi är klara med den grundläggande grammatiken för noterna då vi vill abstrahera bort noterna i ackorden.

BNF ::::::::::

<multi_line_comment> ::=
%
.*
%

<song> ::= |<var>| <tempo> <scale> <motif_block> <segment_block> <structure_block> |end <var>|

<single_line_comment> ::= // .*

<tempo> ::= tempo : \d

<scale> ::= scale : <tone> (major | minor)?

<function_definition> : def <var> <parameters> <block>

<parameters> ::= <parameters> <parameter>

<parameter> ::= <var>

<motif_block> ::= motifs <block>

<segment_block> ::= segments <block>

<structure_block> ::= structure <block>

<block> ::= { (<variable_assignment>+ | <loop>)+ | <function>+ }

<function> ::= <var> <parameters>

<loop> ::= repeat \d <block>

<if_statement> ::= if <boolean_expression> then <block>

<boolean_expression> ::= <expr> <comparator> <expr> | <boolean>

<boolean> ::= true | false

<comparator> ::= = | > | < | >= | <=

<variable_assignment> ::= <var> = <motif>

<var> ::= \w+

<motif> ::= (<note>)+

<note> ::= <length>? <tone> <octave>?

<octave> ::= -2 -1 ”+1 ”+2”

<length> ::= ”2 ”3 ”4 ”8 ”16”

<tone> ::= [a-g] (” b”)? | z

- Statisk vs dynamisk typning?

  Preliminärt anses statisk typning mer intuitivt än dynamisk. Dels då önskan inte är att variabler ska kunna byta typ och dels för ökad läsbarhet vilket är ett av huvudsyftena med språket.

- När en funktion anropas med komplexa datatyper så som strängar eller listor, blir det en kopia eller referens?

  Vi tänker att en funktion ska verka på en kopia av den data den anropas med. Detta för att de motivsom definieras inte ska ändras från ena funktionsanropet till nästa.

- Om språket är objektorienterat, finns det: Public/protected/private? Arv? Osv.

  Vi undrar om arv är rätt väg att gå då alla basklasserna i programmet har samma eller liknande datamedlemmar.

- Hur funkar scopet: Vid anrop av funktion? I blocket tillhörande en for-loop? Delar funktioner och variabler scope? Osv (OBS: scope tas upp på senare föreläsning, så ni vill kanske vänta lite med att beskriva scopet).

  Hela programmet är tänkt att ha en nästlad scopeorienterad struktur då alla klasser hör ihop med varandra.