Lessa

Peng Song - Project Manager
Diyue Xiao - Language Guru
Jiaqi Liu - System Architect
Mingfei Ge - System Integrator
Yuanyuan Kong - System Tester

Language Design

Easy to Learn

Experimental

Object-oriented

High-Level

Interpreted

FUN!

Hello World!

```
print(“Hello World!”);
```

Motivation

Music is fun to play with, But difficult to make:

No instrument at hand to test.
No enough knowledge in music theory.
…

Our Language tries to let users to write simple code with “C D E F G A B” to make music they want!

Language Properties

- 1. Simple: easy to learn and write
  - Straightforward syntax
  - Good Readability
- 2. Powerful
  - Loops, condition statements to use, etc.
- 3. Convenient to test
  - Interactive: execute line by line

Straight-forward Syntax

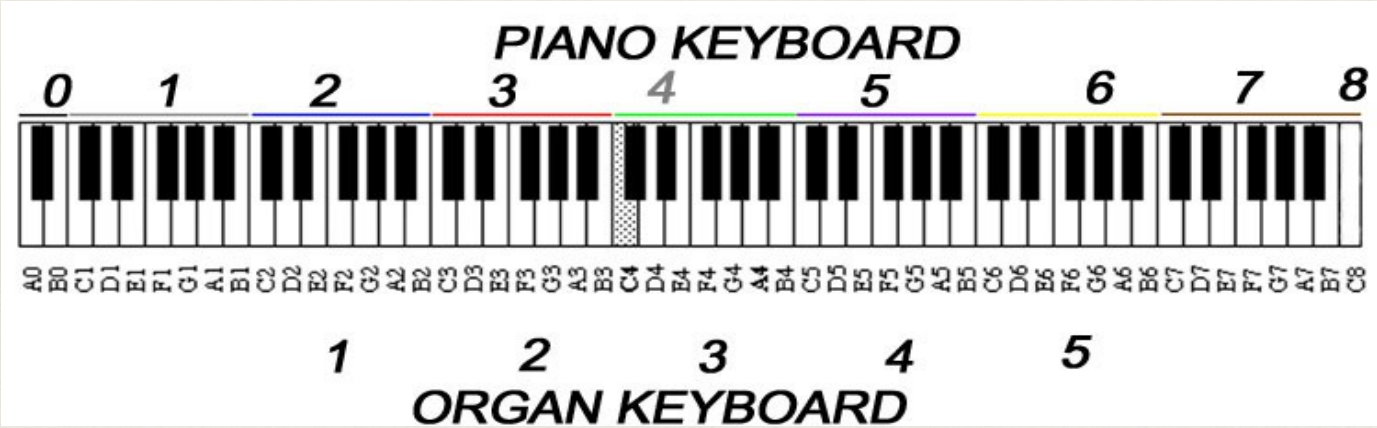
```
$$ file: test.le


myNote = ‘C4w’;

mySeq = ['C3q', 'C3q', 'G3q'];

mySong = {};

mySong.add(mySeq);

mySong.create_MIDI();

mySong.play();
```

Also Powerful…

```
myNote = ‘C4w’;

mySeq = [];

while (myNote < ‘D6w’) {

        mySeq.append(myNote);

        myNote += 1;

}

mySong = {};

mySong.add(mySeq);
```

Music-related Syntactic Constructs (1)

- **Note**: a token recognized by the lexer. We follow the piano keyboard standard. The last character denote the duration: w (whole note), h (half note), q (quarter note), e(eighth note), and s



Note

- Note can be increase or decrease.

```
myNote = ‘C3w’;

myNote += 1;

print(myNote);

>>> ‘D3w’
```

Note

- Flat and sharp is implemented as an operator.

```
myNote = ‘C3w’;

myNote  = #myNote;

print(myNote);

>>> #'C3w'
```

Music-related Syntactic Constructs (2)

- sequence: consists of a list of notes. Begin with ‘[’ and end with ‘]’. We can append note to a sequence by using function mySeq.append().

mySeq = [‘C3w’, ‘C3q’];

The grammar rule to analyze sequence:

atom - > ‘[‘ sequencemaker ‘]’

sequencemaker -> (NOTE | NAME) (',' (NOTE | NAME))*

Sequence

- Concatenation can be done by operator + and *.

```
seq1 = [‘C3w’];

seq2 = [‘D4q’];

seq1 = seq1 + seq2;

print(seq1);

>>> [‘C3w’, ‘D4q’]

seq1 = seq1 * 2;

Print(seq1);
```

Sequence

- We can set instrument for sequence.

mySeq.instrument = “Guitar”;

- We can append a note to a sequence.

mySeq.append(‘C4w’);

Music-related Syntactic Constructs (3)

- song: consists of several sequence, and is the object we use to play music. It begins with ‘{‘ and end with ‘}’. We add sequence to a song by using function add().

```
mySong = {};

mySong.add(seq1);

mySong.add(seq2);
```

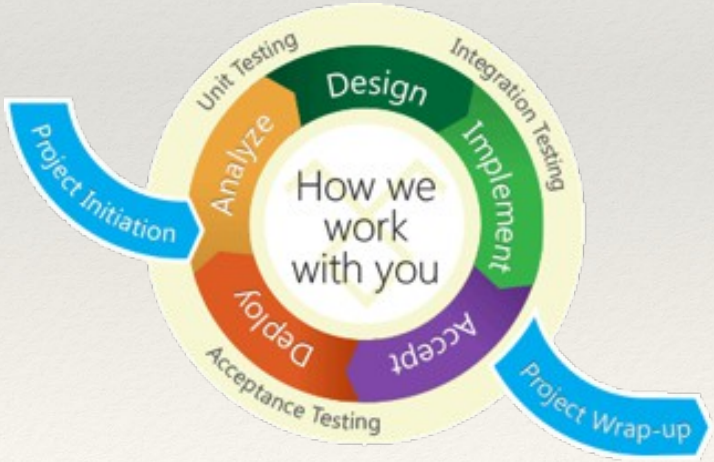
Project Management

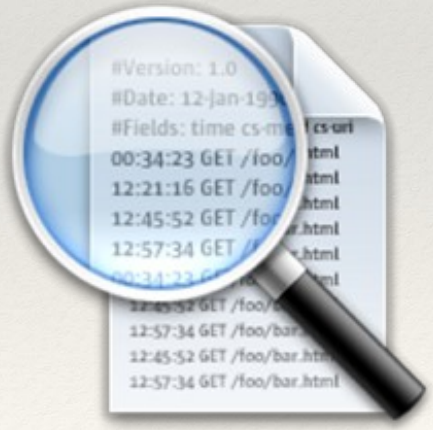
GitHub

slack

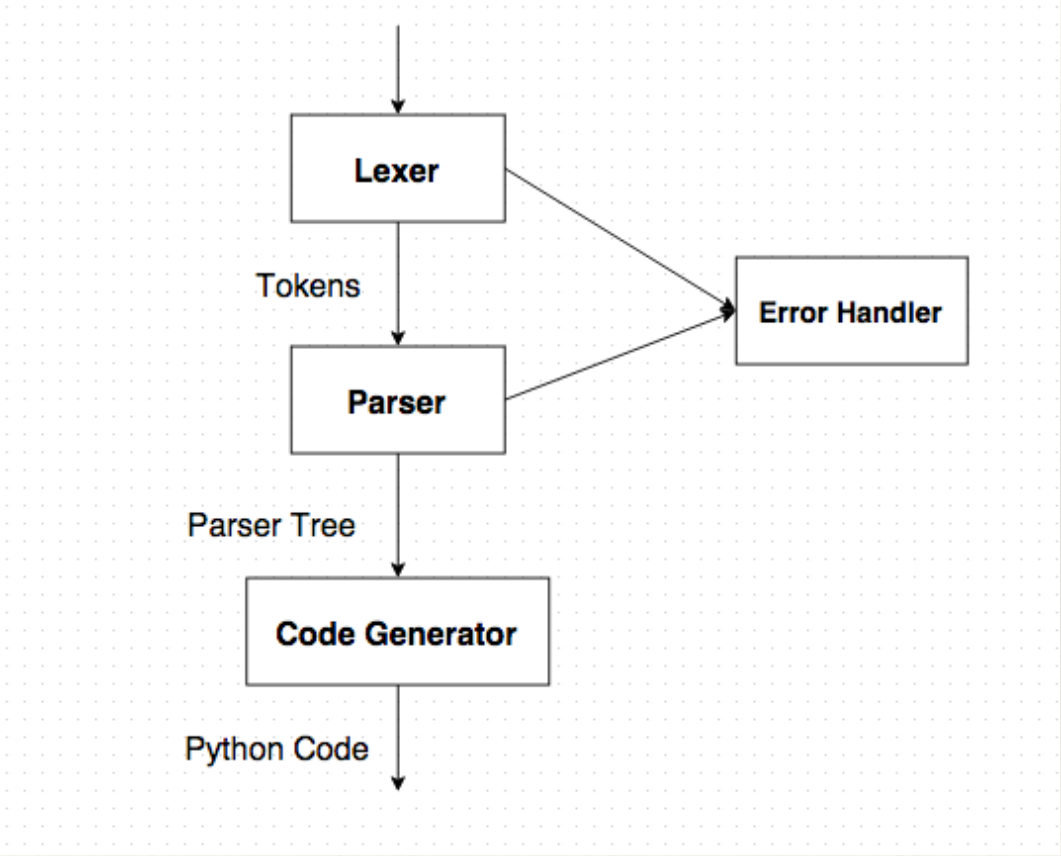
Google Drive

join me





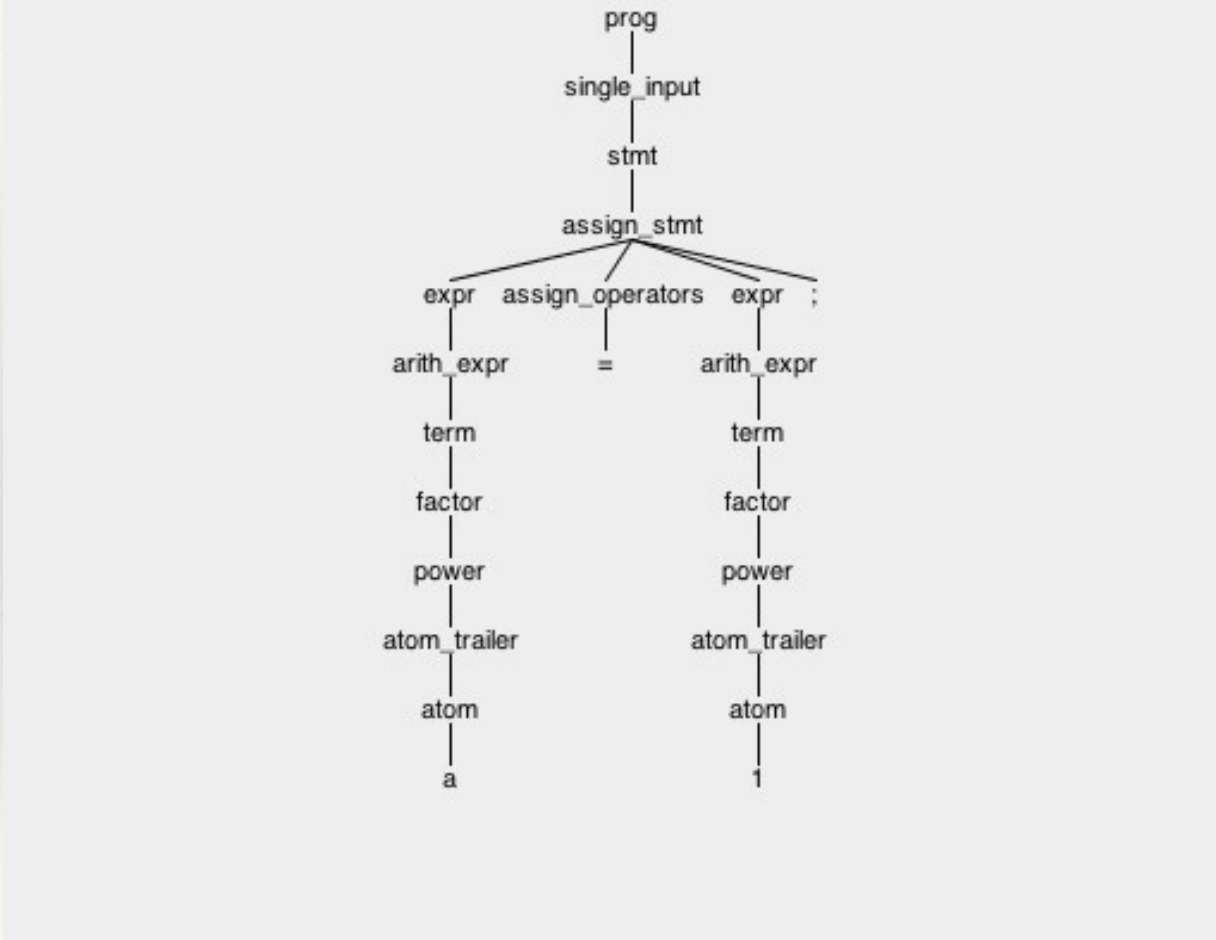
Architecture Design



Architecture Design

```
input = new ANTLRInputStream(stream);
//lexer
ExprLexer lexer = new ExprLexer(input);
lexer.removeErrorListeners();
lexer.addErrorListener(TokenErrorListener.INSTANCE);
CommonTokenStream tokens = new CommonTokenStream(lexer);

//parser
ExprParser parser = new ExprParser(tokens);
parser.removeErrorListeners();
parser.addErrorListener(DescriptiveErrorListener.INSTANCE);
ParseTree tree = parser.prog();
```

Architecture Design



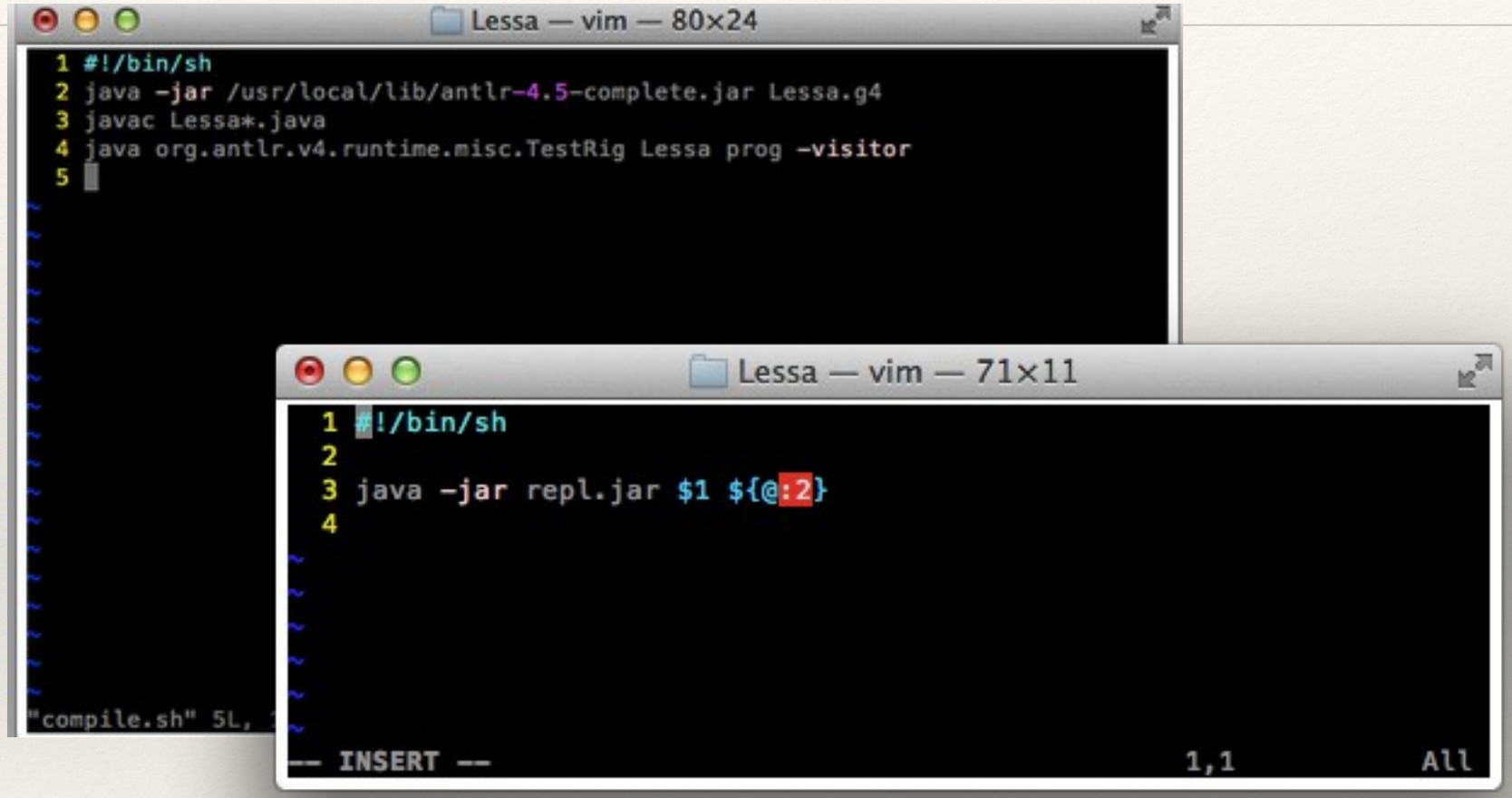
Architecture Design

```
def sort(array){
    less = ();
    equal = ();
    greater = ();
    if (len(array) > 1) {
        pivot = array[0];
        for (x in array) {
            if (x < pivot){
                less.append(x);
                }
            if (x == pivot){
                equal.append(x);
                }
            if (x > pivot){
                greater.append(x);
                }
        }
        return sort(less)+equal+sort(greater);
    }else{
        return array;
    }
}
print(sort((12,4,5,6,7,3,1,15)));
```

```
def sort(array):
  less = []
  equal = []
  greater = []
  if len(array) > 1:
    pivot = array[0]
    for x in array:
      if x < pivot:
        less.append(x)
      if x == pivot:
        equal.append(x)
      if x > pivot:
        greater.append(x)
    return sort(less) + equal + sort(greater)
  else:
    return array


print(sort([12, 4, 5, 6, 7, 3, 1, 15]))
```

Development Tools

Make File


Lessa — vim — 80×24
1 #!/bin/sh
2 java -jar /usr/local/lib/antlr-4.5-complete.jar Lessa.g4
3 javac Lessa*.java
4 java org.antlr.v4.runtime.misc.TestRig Lessa prog -visitor
5
"compile.sh" 5L,
Lessa — vim — 71×11
1 #!/bin/sh
2
3 java -jar repl.jar $1 ${@:2}
4
-- INSERT --
1,1
All

Runtime Environment

Interactive Mode

Variable Reference

Class

```
class C_Major(){
        def to_B_Major(sequence){
                for (i in range(len(sequence))){
                        sequence[i] -= 1;
                        if (sequence[i].tone == "E" or sequence[i].tone == "A"){
                                sequence[i] = ~sequence[i];
                        }
                }
                return sequence;
        }

        def to_G_Major(sequence){
                for (i in range(len(sequence))){
                        sequence[i] += 5;
                        if (sequence[i].tone == "F"){
                                sequence[i] = #sequence[i];
                        }
                }
                return sequence;
        }
}
```

Class

```
seq1 = ['C4q', 'C4q', 'G4q', 'G4q', 'A4q', 'A4q', 'G4w', 'F4q', 'F4q', 'E4q', 'E4q', 'D4q', 'D4q', 'C4w'];
print("original sequence: " + seq1);

C_Major_instance = C_Major();

seq_in_B = C_Major_instance.to_B_Major(seq1);
print("sequence in B Major: " + seq_in_B);

seq_in_G = C_Major_instance.to_G_Major(seq1);
print("sequence in G Major: " + seq_in_G);
```

Tools

-Junit

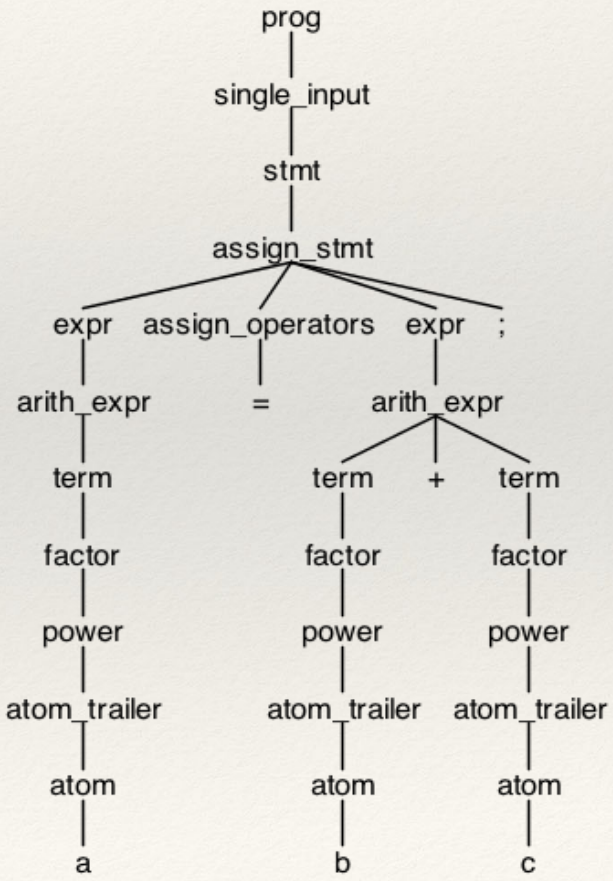
- Unit Test Framework
- The “white box” test

…

-Antlr

- TestRig

tokens, tree, gui



Test Cases

- 1. Lexer
  - Tokens
- 2. Parser
  - Definition of the grammar
- 3. Visitor
  - Generated python code

Bugs found

- Grammar Definition

  - semicolons, commas, parenthesis

    e.g. stmt, expr, …

  - parse trees

- Code Generation

  - indent

    if, while, …

  - music-related expressions

    e.g. note in Lessa: ‘C3w’

    in generated python code: note(‘C3w’)

Integration Test

- Basic functions
  - Quick Sort

```
def sort(array){
less = ();
  equal = ();
  greater = ();
  if (len(array) > 1) {
  pivot = array[0];
  for (x in array) {
    if (x < pivot){
                less.append(x);
    }
              if (x == pivot){
                equal.append(x);
```

More Examoples...

MinStack

```
class MinStack {
        def __init__(){
                this.data =
();
        this.min = ();
        }
    def push(x) {
        if (len(this.data) == 0) {
            this.data.append(x);
            this.min.append(x);
        } else {
            curr =
this.min[len(this.min)-1];
            this.data.append(x);
            if (x < curr) {

this.min.append(x);
            } else {

this.min.append(curr);
            }
        }
    }

    def pop() {
        this.min.remove(len(data)-1);
        this.data.remove(len(data)-1);
    }

    def top() {
        return this.data[len(this.data)-
1];
    }

    def getMin() {
        return this.min[len(this.data)-1];
    }
}

a = MinStack();
a.push(3);
a.push(1);
a.push(2);
print(a.top());
print(a.getMin());
```

About Music

- Focus:
  - -notes & sequences
  - -functions & attributes
  - -generating & playing MIDI file

```
scale = [];
chord = [];
start1 = 'C4w';
print("scale:");

while(start1 < 'D6w'){
        print (str(start1)+" ");
        scale.append(start1);
        start1 += 1;
}
print ("broken chord: ");

for (n in scale){
        if (n == 'A4w'){
                print(" ");
                break;
        }
        elif (n.pitch != "C" and n.pitch != "E" and
n.pitch != "G"){
                continue;
        }
        else{
                print(str(n) + " ");
        }
        chord.append(n);
}
```

Lesson Learned

Team work, Project Management

Design well before start implementing

Use version control tool wisely

Write good tests

Communication is educational, and fun

Leanrn how to use the tools

Twinkle Twinkle Little Sta

```
seq1 = ['C4q', 'C4q', 'G4q', 'G4q', 'A4q', 'A4q', 'G4w', 'F4q', 'F4q', 'E4q', 'E4q', 'D4q', 'D4q', 'C4w'];
seq2 = ['G4q', 'G4q', 'F4q', 'F4q', 'E4q', 'E4q', 'D4w'];

seq3 = [];
seq3 = seq1 + seq2 * 2 + seq1;
print(seq3);
seq3.instrument = "Piano";

seq4 = ['C3w', 'E3w', 'F3w', 'E3w', 'D3w', 'C3w', 'F3q', 'G3q', 'E3w'];
seq4[3].pitch_up();
seq5 = ['E3w', 'D3w', 'C3w', 'C3q', 'B2q'];

seq6 = [];
seq6 = seq4 + seq5 + seq4;
print(seq6);
seq6.instrument = "Piano";

twinkle = {};
twinkle.add(seq3);
twinkle.add(seq6);

twinkle.create_MIDI();
twinkle.play();
```