Programming constructs for beatmaking

What is a beat?

– – – – – – – – – – – – – –

What is a beat?

→ A regular repeating pulse i.e foot tap

- - - - - - - - - - - - - -

What is a beat?

→ A regular repeating pulse i.e foot tap

→ A pattern (or rhythm) played by drums

- - - - - - - - - - - - - -

What is a beat?

→ A regular repeating pulse i.e foot tap

→ A pattern (or rhythm) played by drums

- - - - - - - - - - - - - -

What is a beat?

→ A regular repeating pulse i.e foot tap

→ A pattern (or rhythm) played by drums

- - - - - - - - - - - - - -

→ **Tempo ~ Speed of the pattern**

Tempo

```
# 50 80 160

use_bpm 80

live_loop :amen do
  sample :loop_amen, beat_stretch: 3
  sleep 3
end
```

→ Hip-hop: 60-100 bpm

→ House: 115-130 bpm

→ Techno/trance: 120-140 bpm

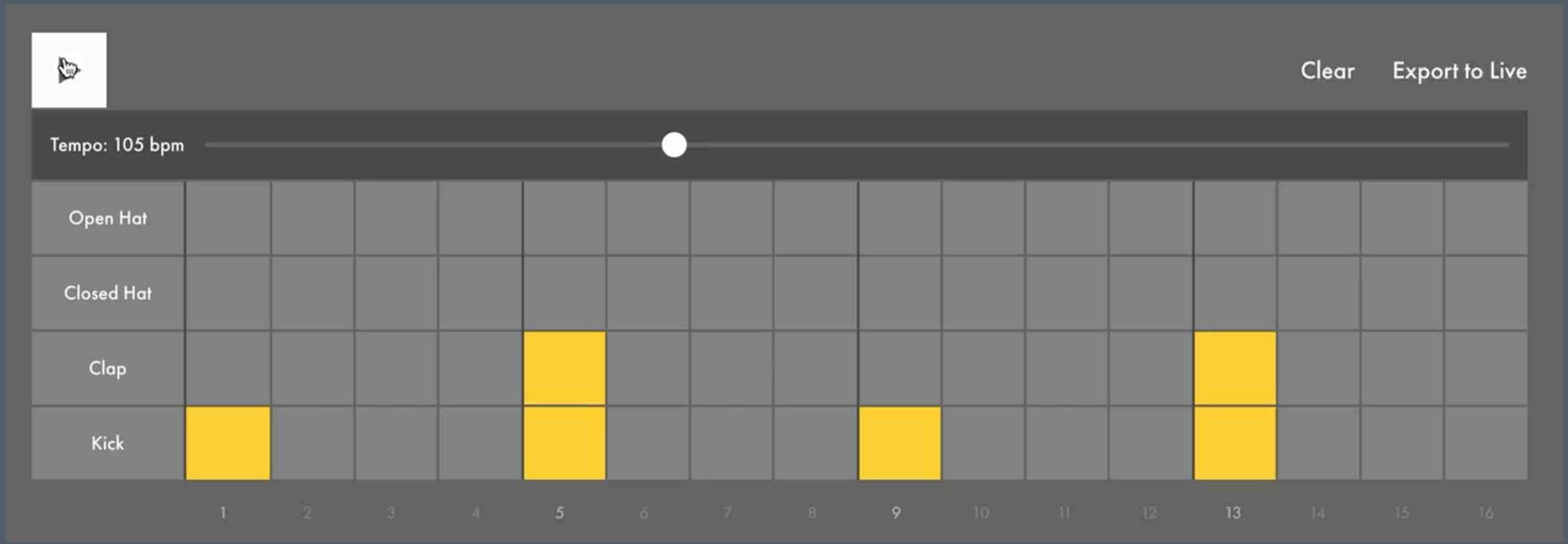
→ Dubstep: 135-145 bpm

→ Drum & bass: 160-180 bpm

A Basic Beat

```
use_bpm 110

live_loop :kick do
  ##| stop
  sample k
  sleep 1
end

live_loop :clap, sync: :kick do
  ##| stop
  sample c
  sleep 2
end
```

On the Grid


Clear
Export to Live
Tempo: 105 bpm
Open Hat
Closed Hat
Clap
Kick
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Coding the Grid

```
grid1= [ 1,0,0,0,  2,0,0,0 ]

live_loop :drum do
  8.times do |index|
    sample kick if [1, 2].include? grid1[index]
    sample clap if grid1[index] == 2
    sleep 0.25
  end
end
```

Coding the Grid

```
grid1= [ 1,0,0,0,  2,0,0,0 ]

live_loop :drum do
  8.times do |index|
    sample kick if [1, 2].include? grid1[index]
    sample clap if grid1[index] == 2
    sleep 0.25
  end
end
```

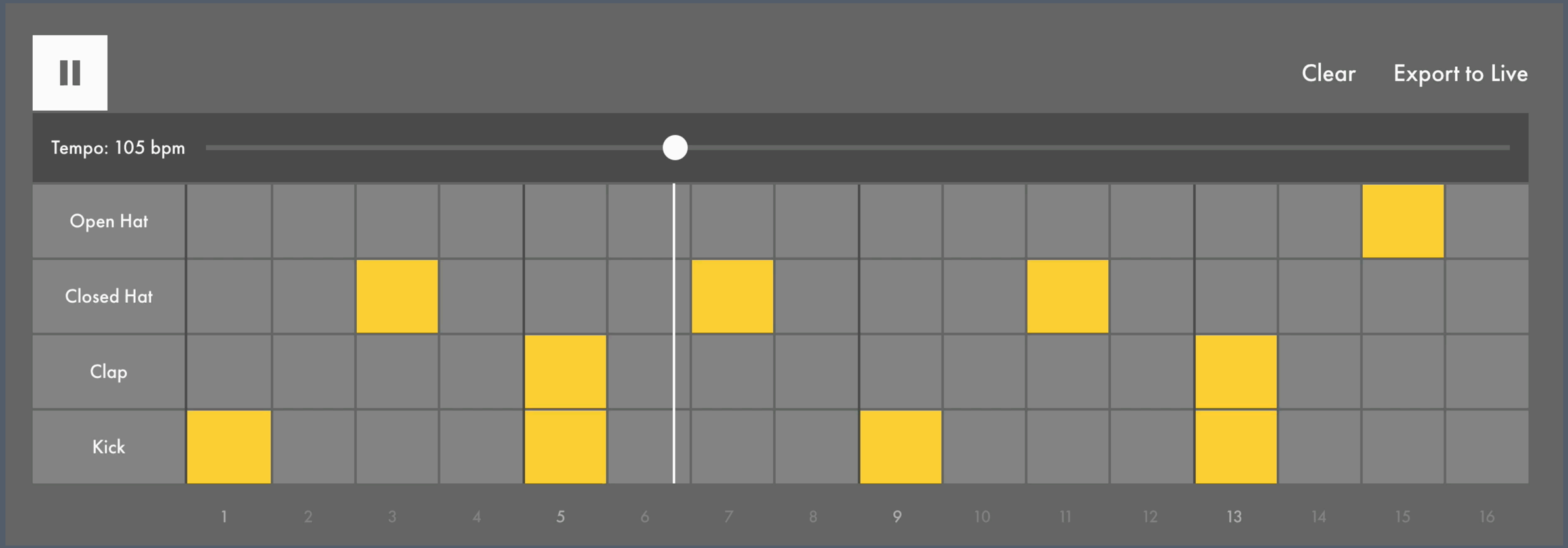
Coding the Grid

```
grid1= [ 1,0,0,0,  2,0,0,0 ]

live_loop :drum do
  8.times do |index|
    sample kick if [1, 2].include? grid1[index]
    sample clap if grid1[index] == 2
    sleep 0.25
  end
end
```

Coding the Grid

```
grid1= [ 1,0,0,0,  2,0,0,0 ]

live_loop :drum do
  8.times do |index|
    sample kick if [1, 2].include? grid1[index]
    sample clap if grid1[index] == 2
    sleep 0.25
  end
end
```

Yassify the Grid



```
use_bpm 100

grid1= [
  1,0,3,0, 2,0,3,0,
  1,0,3,0, 2,0,4,0,
]

live_loop :drum do
  16.times do |index|
    sample kick, amp: 1.5 if [1, 2].include? grid1[index]
    sample clap if grid1[index] == 2
    sample hat1 if grid1[index] == 3
    sample hat2 if grid1[index] == 4
    sleep 0.25
  end
end
```

🥺 No grid

```
use_bpm 110

live_loop :kick do
  ##| stop
  sample k
  sleep 1
end

live_loop :clap, sync: :kick do
  ##| stop
  sample c
  sleep 2
end

live_loop :hat, delay: 0.5 do
  ##| stop
  sample ch, amp: 0.5
  sleep 1
end

live_loop :cymbal, sync: :kick do
  ##| stop
  sleep 2.5
  sample oh
  sleep 1.5
end
```

Create new grids for Complex beats

```
kick_snare= [
  1,0,0,0, 0,0,0,0,
  0,1,0,0, 2,0,0,0,
  0,0,1,0, 0,0,0,0,
  0,0,0,0, 0,0,1,0,
  0,0,0,0, 2,0,0,0,
  0,0,1,0, 0,0,0,0,
]

hihat= [
  3,0,0,3, 0,0,3,0,
  0,3,0,0, 3,3,3,0,
  0,0,3,0, 0,3,0,0,
  3,0,0,3, 0,0,3,0,
  0,3,0,0, 3,0,0,3,
  0,0,3,0, 0,3,0,0,
]
```

Conditional Statements

```
# If <Statement> is True Do <Operation>

live_loop :kick do
  at [1, 3, 5] do
    sample kick
  end
  sleep 8
end
```

Sequencing Patterns

```
# With conditionals we can sequence and play with patterns

define :drum_pattern do |pattern|
  v = pattern.tick(:pattern)
  if v == "x"
    return sample k
  elsif v == "o"
    return sample c, amp: 0.5
  elsif v == "-"
    return sample oh, amp: 0.5
  end
end
```

Make it Bounce

```
use_bpm 120

live_loop :closedHiHat do # Vary the amplitude
  pattern = "4--3--3--1234--3--3--3--".ring

  pattern.length.times do
    sample :drum_cymbal_closed,
      amp: (pattern[look].to_f),
      sustain: 0.2 if (pattern[tick] != "-")

    sleep 4/pattern.length.to_f

  end
end
```


THIS TALK is like
a burrito
because i'd
like to wrap
mine up
imgflip.com