# Euterpea Signal-Level Quick Reference

Donya Quick     22-Dec-2016

**Wave Tables and Oscillators**

`t = tableSinesN numSamples partialsList`     example: `tableSinesN 4096 [1]`

`t = tableLinear y0 syPairs`     example: `tableLinear 0 [(0.5, 1.0), (0.5, -1.0)].`

Note: `syPairs` represents pairs of *segment lengths* (note absolute x-coords) and y values (amplitudes).

See also: `tableExponN, tablesSines3N` (takes triples of partial number, strength, & phase offset)

`y <- osc tableName phaseOffset -< frequency`     basic oscillator syntax

See also: `oscI` (linear interpolation version of `osc`)

**Basic Signal Syntax**

General Format

```
sigName :: AudSF InType OutType
sigName = proc inSig -> do
  outSig <- anotherSigFun -< inSig
  returnA -< outSig
```

Example: 440Hz Sine Wave

```
sine440 :: AudSF () Double
sine440 = proc _ -> do
  y <- osc sineTable 0 -< 440
  returnA -< y
main = outFile "x.wav" 2.0 sine440
```

**Commonly Used Signal Functions**

White noise generator: `n <- noiseWhite intSeed -< ()`

Delay line: `outSignal <- delay sec -< inSignal`

Variable delay line: `outSig <- delay maxDel -< (inSig, delAmt)` where `delAmt` ≤ `maxDel`

Low-pass filter: `outSignal <- filterLowPass -< (inSignal, halfPowerHz)`

High-pass filter: `outSignal <- filterHighPass -< (inSignal, halfPowerHz)`

Butterworth low-pass: `outSig <- filterLowPassBW -< (inSig, cutoffFreq)`

Butterworth high-pass: `outSig <- filterHighPassBW -< (inSig, cutoffFreq)`

Butterworth band-pass: `outSig <- filterBandPassBW -< (inSig, cutoffFreq, bandWidth)`

Butterworth band-stop: `outSig <- filterBandStopBW -< (inSig, cutoffFreq, bandWidth)`

Linear envelope: e <- envLineSeg [$y_0$, $y_1$, ..., $y_n$] [$d_1$, ..., $d_n$] -< ()

where $y_i$`::Double` is an amplitude and $d_i$`::Double` is a duration in seconds. The list of amplitudes should always contain *one more value* than the list of durations. See also: `envExponSeg`.

**Virtual Instrument Creation and Usage**

Mono Instrument Format

```
instr1 :: Instr (Mono AudRate)
instr1 dur pch vol params =
  let freq = apToHz pch
  in  proc _ -> do
    …
    returnA -< outSignal
```

Stereo Instrument Format

```
instr2 :: Instr (Stereo AudRate)
instr2 dur pch vol params =
  let freq = apToHz pch
  in  proc _ -> do
    …
    returnA -< (leftSig, rightSig)
```

Using Your Instruments

`myName = CustomInstrument "Foo"`

`instrMap :: InstrMap (Mono AudRate)`     instruments used must be all mono or all stereo, not mixed

`instrMap = [(myName, instr1), …]`

`myMel = instrument myName musicVal`     musicVal must use only instrument names in instrMap

`writeIt = writeWav "m.wav" instrMap myMel`

See also: `writeWavNorm` (normalizes amplitudes to [-1.0, 1.0])