Arithmetic

Tools to perform arithmetic operations on audio signals.

Sin

class **Sin**(*input*, *mul=1*, *add=0*) [source]

Performs a sine function on audio signal.

Returns the sine of audio signal as input.

Parent: `PyoObject`

Args: input: PyoObject

Input signal, angle in radians.

```
>>> s = Server().boot()
>>> s.start()
>>> import math
>>> a = Phasor(500, mul=math.pi*2)
>>> b = Sin(a, mul=.3).mix(2).out()
```

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

input

PyoObject. Input signal to process.

Cos

class **Cos**(*input*, *mul=1*, *add=0*) [source]

Performs a cosine function on audio signal.

Returns the cosine of audio signal as input.

Parent: `PyoObject`

Args: input: PyoObject

Input signal, angle in radians.

```
>>> s = Server().boot()
>>> s.start()
>>> import math
>>> a = Phasor(500, mul=math.pi*2)
>>> b = Cos(a, mul=.3).mix(2).out()
```

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

input

PyoObject. Input signal to process.

Tan

class **Tan**(*input*, *mul=1*, *add=0*) [source]

Performs a tangent function on audio signal.

Returns the tangent of audio signal as input.

Parent: `PyoObject`

Args: input: PyoObject

Input signal, angle in radians.

```
>>> s = Server().boot()
>>> s.start()
>>> # Tangent panning function
>>> import math
>>> src = Sine(mul=.3)
>>> a = Phasor(freq=1, mul=90, add=-45)
>>> b = Tan(Abs(a*math.pi/180))
>>> b1 = 1.0 - b
>>> oL = src * b
>>> oR = src * b1
>>> oL.out()
>>> oR.out(1)
```

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

`input`

PyoObject. Input signal to process.

Tanh

class **`Tanh`**(*input*, *mul=1*, *add=0*) [source]

Performs a hyperbolic tangent function on audio signal.

Returns the hyperbolic tangent of audio signal as input.

Parent: `PyoObject`

Args: input: PyoObject

Input signal, angle in radians.

```
>>> s = Server().boot()
>>> s.start()
>>> import math
>>> a = Phasor(250, mul=math.pi*2)
>>> b = Tanh(Sin(a, mul=10), mul=0.3).mix(2).out()
```

`setInput`(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

`input`

PyoObject. Input signal to process.

Abs

class **`Abs`**(*input*, *mul=1*, *add=0*) [source]

Performs an absolute function on audio signal.

Returns the absolute value of audio signal as input.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

```
>>> s = Server().boot()
>>> s.start()
>>> # Back-and-Forth playback
>>> t = SndTable(SNDS_PATH + "/transparent.aif")
>>> a = Phasor(freq=t.getRate()*0.5, mul=2, add=-1)
>>> b = Pointer(table=t, index=Abs(a), mul=0.5).mix(2).out()
```

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

input

PyoObject. Input signal to process.

Sqrt

class **Sqrt**(*input*, *mul=1*, *add=0*) [source]

Performs a square-root function on audio signal.

Returns the square-root value of audio signal as input.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

```
>>> s = Server().boot()
>>> s.start()
>>> # Equal-power panning function
>>> src = Sine(mul=.3)
>>> a = Abs(Phasor(freq=1, mul=2, add=-1))
>>> left = Sqrt(1.0 - a)
>>> right = Sqrt(a)
>>> oL = src * left
>>> oR = src * right
>>> oL.out()
>>> oR.out(1)
```

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

`input`

PyoObject. Input signal to process.

Log

class **`Log`**(*input*, *mul=1*, *add=0*) [source]

Performs a natural log function on audio signal.

Returns the natural log value of of audio signal as input. Values less than 0.0 return 0.0.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

```
>>> s = Server().boot()
>>> s.start()
# Logarithmic amplitude envelope
>>> a = LFO(freq=1, type=3, mul=0.2, add=1.2) # triangle
>>> b = Log(a)
>>> c = SineLoop(freq=[300,301], feedback=0.05, mul=b).out()
```

`setInput`(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

`input`

PyoObject. Input signal to process.

Log2

class **`Log2`**(*input*, *mul=1*, *add=0*) [source]

Performs a base 2 log function on audio signal.

Returns the base 2 log value of audio signal as input. Values less than 0.0 return 0.0.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

```
>>> s = Server().boot()
>>> s.start()
# Logarithmic amplitude envelope
>>> a = LFO(freq=1, type=3, mul=0.1, add=1.1) # triangle
>>> b = Log2(a)
>>> c = SineLoop(freq=[300,301], feedback=0.05, mul=b).out()
```

`setInput`(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

`input`

PyoObject. Input signal to process.

Log10

class `Log10`(*input*, *mul=1*, *add=0*) [source]

Performs a base 10 log function on audio signal.

Returns the base 10 log value of audio signal as input. Values less than 0.0 return 0.0.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

```
>>> s = Server().boot()
>>> s.start()
# Logarithmic amplitude envelope
>>> a = LFO(freq=1, type=3, mul=0.4, add=1.4) # triangle
>>> b = Log10(a)
>>> c = SineLoop(freq=[300,301], feedback=0.05, mul=b).out()
```

`setInput`(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

`input`

PyoObject. Input signal to process.

Atan2

class **`Atan2`**(*b=1*, *a=1*, *mul=1*, *add=0*) [source]

Computes the principal value of the arc tangent of b/a.

Computes the principal value of the arc tangent of b/a, using the signs of both arguments to determine the quadrant of the return value.

Parent: `PyoObject`

Args: b: float or PyoObject, optional

Numerator. Defaults to 1.

a: float or PyoObject, optional

Denominator. Defaults to 1.

```
>>> s = Server().boot()
>>> s.start()
>>> # Simple distortion
>>> a = Sine(freq=[200,200.3])
>>> lf = Sine(freq=1, mul=.2, add=.2)
>>> dist = Atan2(a, lf)
>>> lp = Tone(dist, freq=2000, mul=.1).out()
```

`setB`(*x*) [source]

Replace the *b* attribute.

Args: x: float or PyoObject

new *b* attribute.

`setA`(*x*) [source]

Replace the *a* attribute.

Args: x: float or PyoObject

new *a* attribute.

b

float or PyoObject. Numerator.

a

float or PyoObject. Denominator.

Floor

class **Floor**(*input*, *mul=1*, *add=0*) [source]

Rounds to largest integral value not greater than audio signal.

For each samples in the input signal, rounds to the largest integral value not greater than the sample value.

Parent: PyoObject

Args: input: PyoObject

Input signal to process.

```
>>> s = Server().boot()
>>> s.start()
>>> # Clipping frequencies
>>> sweep = Phasor(freq=[1,.67], mul=4)
>>> flo = Floor(sweep, mul=50, add=200)
>>> a = SineLoop(freq=flo, feedback=.1, mul=.3).out()
```

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

input

PyoObject. Input signal to process.

Ceil

class **Ceil**(*input*, *mul=1*, *add=0*) [source]

Rounds to smallest integral value greater than or equal to the input signal.

For each samples in the input signal, rounds to the smallest integral value greater than or equal to the sample value.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

```
>>> s = Server().boot()
>>> s.start()
>>> # Clipping frequencies
>>> sweep = Phasor(freq=[1,.67], mul=4)
>>> flo = Ceil(sweep, mul=50, add=200)
>>> a = SineLoop(freq=flo, feedback=.1, mul=.3).out()
```

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

input

PyoObject. Input signal to process.

Round

class **Round**(*input*, *mul=1*, *add=0*) [source]

Rounds to the nearest integer value in a floating-point format.

For each samples in the input signal, rounds to the nearest integer value of the sample value.

Parent: `PyoObject`

Args: input: PyoObject

Input signal to process.

```
>>> s = Server().boot()
>>> s.start()
>>> # Clipping frequencies
>>> sweep = Phasor(freq=[1,.67], mul=4)
>>> flo = Round(sweep, mul=50, add=200)
>>> a = SineLoop(freq=flo, feedback=.1, mul=.3).out()
```

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

`input`

PyoObject. Input signal to process.

Pow

class **`Pow`**(*base=10*, *exponent=1*, *mul=1*, *add=0*) [source]

Performs a power function on audio signal.

Parent: `PyoObject`

Args: base: float or PyoObject, optional

Base composant. Defaults to 10.

exponent: float or PyoObject, optional

Exponent composant. Defaults to 1.

```
>>> s = Server().boot()
>>> s.start()
>>> # Exponential amplitude envelope
>>> a = LFO(freq=1, type=3, mul=0.5, add=0.5)
>>> b = Pow(Clip(a, 0, 1), 4, mul=.3)
>>> c = SineLoop(freq=[300,301], feedback=0.05, mul=b).out()
```

`base`

float or PyoObject. Base composant.

`exponent`

float or PyoObject. Exponent composant.

`setBase`(*x*) [source]

Replace the *base* attribute.

Args: x: float or PyoObject

new *base* attribute.

`setExponent`(*x*) [source]

Replace the *exponent* attribute.

Args: x: float or PyoObject

new *exponent* attribute.

Exp

class **Exp**(*input*, *mul=1*, *add=0*) [source]

Calculates the value of e to the power of x.

Returns the value of e to the power of x, where e is the base of the natural logarithm, 2.718281828...

Parent: `PyoObject`

Args: input: PyoObject

Input signal, the exponent.

```
>>> s = Server().boot()
>>> s.start()
>>> a = Sine(freq=200)
>>> lf = Sine(freq=.5, mul=5, add=6)
>>> # Tanh style distortion
>>> t = Exp(2 * a * lf)
>>> th = (t - 1) / (t + 1)
>>> out = (th * 0.3).out()
```

setInput(*x*, *fadetime=0.05*) [source]

Replace the *input* attribute.

Args: x: PyoObject

New signal to process.

fadetime: float, optional

Crossfade time between old and new input. Default to 0.05.

input

PyoObject. Input signal to process.