All of Monolib

Container data types

The whole library is based on the `Mono` data type, which is simply a `NamedTuple` with three fields

- `data`: this field is reserved for a 1D array representing the values of a signal.
- `sample_rate`: the sample rate of said signal.
- `tags`: any relevant meta data that you wish to attach to said signal.

```
from monolib.containers import mono

mono([1, 2, 3], 8_000, {"type": "B"})
```

```
Mono(data=[1, 2, 3], sample_rate=8000, tags={'type': 'B'})
```

`Mono` containers can be grouped into `MonoCollections`, which are nothing but a `NamedTuple` which includes a sequence of `Mono` along with optional `tags` for collection-wide meta data.

```
from monolib.containers import collect

x1 = mono([1, 2, 3], 8_000, {"type": "A"})
x2 = mono([4, 5, 6], 8_000, {"type": "B"})
x3 = mono([7, 8, 9], 8_000, {"type": "C"})

collect(x1, x2, x3)
```

```
MonoCollection(
  entries=(
    Mono(data=[1, 2, 3], sample_rate=8000, tags={'type': 'A'}),
    Mono(data=[4, 5, 6], sample_rate=8000, tags={'type': 'B'}),
    Mono(data=[7, 8, 9], sample_rate=8000, tags={'type': 'C'}),
```

```
  ),
  tags={}
)
```

This abstraction allows us to define composable signal processing transformations which in a functional programming style, which allows us to do some cool and useful stuff.

Chainable transformations

Thats it