# peact Documentation

***Release 0.1***

**Matthew Spellings**

**Oct 28, 2018**

# Contents

**Peact** is a library for reactive programming in python.

CHAPTER 1

# Installation

Installation works using distutils, for example:

```
python setup.py install --user
```

peact uses cython to build a C extension. If you update *peact/_peact.pyx*, you can trigger the cython code to rebuild with *–cython*:

```
python setup.py install --user --cython
```

CHAPTER 2

# Introduction

As an analogy for peact, consider the process of building software. The predominant build method on UNIX systems involves *Makefiles*, which specify files which can be created and "recipes" to create each file. Each file has a number of dependencies, which the make system will ensure have been created before the recipe for the file is run.

Peact is a library which enables a similar method of programming inside python instead of on the filesystem. Rather than the *make* program, peact is the orchestrator of activity. Instead of files, peact deals with "quantities," each with a particular name. The recipes and file contents of *make* are replaced with python functions and python objects, respectively.

In other words, peact allows you to string together python functions which consume and produce quantities. As input values change, nodes in the graph are updated in response to these changes, potentially updating other nodes as well.

# CHAPTER 3

# Reactive Python API

To use peact, create a `peact.CallGraph` object and `peact.CallGraph.register()` `peact.CallNode` objects (representing functions) on it. Input values can come from nodes which themselves have no inputs or by calling `peact.CallGraph.inject()` to immediately set values.

After the `peact.CallGraph` has been prepared, `peact.CallGraph.pump()` can be used to step through the graph and call each registered function which needs to be updated. Values are stored in the *scope* member of a `peact.CallGraph`.

CHAPTER 4

# Indices and tables

- genindex
- modindex
- search