
Python

quansight

May 14, 2021

CONTENTS

1	projects	3
2	Syllabus	5
2.1	writers workshop content	5
2.2	developer notes	12
2.3	github practices for the workshops	13

the quansight writers workshop is an enrichment program focused on the creative side of writing literate programs.

PROJECTS

SYLLABUS

the syllabus is a living thing that will change over time. the current version is available at [[github/readme.md](#)]

2.1 writers workshop content

2.1.1 writing

our first short literate programs

What is array?

This blog post is motivated from a question what is a 0-dimensional array, it's relation to array values, and how to represent such arrays in a programming language. Here the answer will be given only to the first parts of the question (sorry..).

While almost all programming languages implement multidimensional array objects and various operations on these, implementing indexing operations that result in an array object with reduced dimensionality require handling the degenerate case where the resulting dimensionality is 0: should the result be a 0-dimensional array object (of array type) or should the result be an array value object (of value type, e.g. scalar type)? This question has been discussed a lot but there appears to be no definite answer. I think one reason for this originates from “practicality beats purity” based decisions that array implementations are often based on.

In this blog post I'll deliberately distance the discussion from implementation details and try to answer the question what is an array in general. I hope that it will support the discussions on the implementation details as well.

Array definition

Let me define:

An array is a collection of values that can be referenced.

and discuss this as follows.

Array, as a collection, is different from other collections, such as sets or lists, in two aspects:

1. all array values have the same type
2. array values are referenced

The type of an array values does not necessarily need to be a scalar type. The values could be anything, including arrays, as long as they have the same type. This property allows various optimizations at the implementation level, for instance, the common value type information can be stored as an array metadata instead of storing the type information together with each value.

The “referencing of values” means that one can perform array manipulations with references to values rather than with values directly. For instance, to take a slice of an array (that is a specific subset of array) one can avoid (possibly expensive) retrieval of the array values from storage, and instead, one can just restrict the manipulations on the set of references (e.g. via strides manipulations).

In general, a reference to an array value can be any object that represents a unique label (within the given array definition) that is attached to each array value. Hence, *array is a bijective mapping* from a *set of references* to a *set of array values*. (Replacing the adjective “bijective” with “surjective” would apply to sparse array storage formats that use the so-called fill-value which is a value that that most array values are equal to.)

An empty array is an array with no values.

Multi-dimensional array

A multi-dimensional array is an array that reference set is a set of tuples with the same length, called *dimensionality* of the array.

From this definition follows:

a 0-dimensional array is a non-empty array that reference set consists of a single element: a zero-length tuple.

and

the set of 0-dimensional arrays is isomorphic to a set of array values.

That’s it.

Bonus part

Multi-dimensional array with integer indices

Restricting the set of references to tuples of (a compact set of) integers, called *indices*, allows many optimizations. For instance, one does not need to store the reference set when the array values are stored in memory contiguously: for a given tuple of integers one can compute the location of array value in memory storage very efficiently; and vice-versa, given the location of the array value in memory, one can compute the corresponding index as efficiently. In addition, the operation of slicing an array boils down to an operation on shape/strides only.

Running pytest in notebooks

Notebooks are an awesome way to tell stories about code. But sometimes the story that you want to tell is not about the code itself, but rather about the tools handling it. One such example is trying to showcase the popular test framework `pytest` or possibly a plugin that you wrote for it. Fortunately, the Python community got you covered:

In this notebook we take a quick look at `ipytest` and how it is used. It is hosted on `PyPI` so installing it is as easy as `pip install ipytest`:

```
%%capture
!pip install ipytest
```

Within each notebook, there is only a minimal setup. `ipytest.autoconfig()` sets some sensible defaults, but each parameter can also be overwritten. Alternatively, you can use `ipytest.config()` to start with an empty configuration if most of the defaults do not fit your use case.

```
import ipytest
ipytest.autoconfig()
```

Afterwards, we have access to the `run_pytest` cell-magic, which lets us run `pytest` and reports back the results:

```
%%run_pytest [clean]

class TestFoo:
    def test_foo(self):
        assert True

def test_baz():
    assert "baz" == "bar"
```

```
.F                                                                    [100%]
===== FAILURES =====
_____ test_baz _____

    def test_baz():
>         assert "baz" == "bar"
E         AssertionError: assert 'baz' == 'bar'
E             - bar
E             + baz

<ipython-input-3-4d838d46c768>:6: AssertionError
===== short test summary info =====
FAILED tmpth9tsj87.py::test_baz - AssertionError: assert 'baz' == 'bar'
1 failed, 1 passed in 0.04s
```

You may have noticed the additional `[clean]` option. If this is omitted, `run_pytest` will also pick up all previously defined tests:

```
%%run_pytest

import pytest

@pytest.mark.skip
def test_spam():
    pass
```

```
.Fs                                                                    [100%]
===== FAILURES =====
_____ test_baz _____

    def test_baz():
>         assert "baz" == "bar"
E         AssertionError: assert 'baz' == 'bar'
E             - bar
E             + baz

<ipython-input-3-4d838d46c768>:6: AssertionError
===== short test summary info =====
FAILED tmpckwhlzed.py::test_baz - AssertionError: assert 'baz' == 'bar'
1 failed, 1 passed, 1 skipped in 0.01s
```

All additional arguments passed to `run_pytest` will be directly passed to `pytest`:

```
%%run_pytest --collect-only  
  
pass
```

```
tmp17d6l386.py::TestFoo::test_foo  
tmp17d6l386.py::test_baz  
tmp17d6l386.py::test_spam  
  
3 tests collected in 0.00s
```

With the `addopts` parameter you can configure `ipytest` to add some `pytest` flags to all calls of `run_pytest`:

```
ipytest.autoconfig(addopts=("--quiet", "--collect-only"))
```

```
%%run_pytest  
  
pass
```

```
tmp0ctpkc2i.py::TestFoo::test_foo  
tmp0ctpkc2i.py::test_baz  
tmp0ctpkc2i.py::test_spam  
  
3 tests collected in 0.00s
```

If you have a `pytest` configuration file, you can use `addopts` to register it with the `-c` flag:

```
ipytest.autoconfig(addopts=("-c", "pytest.ini"))
```

There are many more options that we didn't cover here, so make sure to checkout `ipytest`'s [README](#).

The content for this Markdown document is copied from `template.tex` and processed with `watch_latex_md.py` script.

Basic Template

Introduction

Consider the two points and . Section *distance* computes the distance between these two points. Section *linear fit* computes a linear equation through the two points, and Section *exponential fit* fits a exponential equation through the two points.

Distance

We can use the distance formula

to determine the distance between any two points and in . For our example, and , so plugging these values into the distance formula *distance* tell us the distance between the two points is

Linear Fit

Consider a linear equation through the two points. We will first determine the slope of the line in Section [slope](#), and we will then determine the -intercept of the line in Section [intercept](#).

Slope

The slope of the line passing through the two points is given by the formula

Plugging in our two points, we find the slope of the line between them is

Intercept

To find the -intercept of the line, we start with the point-slope form of the line of slope through the point :

We plug in the point and the slope we found previously *slope* to obtain the equation

Solving for , we find the slope-intercept form of the line:

Therefore, the -intercept is , and the equation describes the line through the two points.

Exponential Fit

Let us consider the exponential function . For this function to pass through both points, we must find constants and that satisfy both equations and . To solve these two simultaneous equations, we first take the ratio of the two equations, which gives us a single equation involving only :

We can take the natural logarithm of this equation to solve for :

which means .

We can then use this value of , along with either of the two points to solve for . Let us consider the point :

Solving for , we find , and the exponential equation through both points is

Writer's Workshop #3- This Little Piggy...

docs/people/marsbarlee/attachment:image.png

The written Japanese language has three different 'scripts'. Usually kanji doesn't have any pronunciation- you are expected to know what it means and how it's pronounced.

However, for children or new language learners, pronunciation aid can help a lot.

There is a HTML markup to add these pronunciation aids. It is called 'ruby' (confusingly enough, no, it's not the programming language Ruby).

Let's try 'ruby' out.

First, we need to import HTML and display it, since JupyterLab uses Python.

```
from IPython.core.display import display, HTML
```

Next, let's display some kanji

```
display(HTML('<h1></h1>'))
```

```
<IPython.core.display.HTML object>
```

As you can see, the first character is kanji, and the other three are hiragana. If I can only read hiragana, I have no idea what the full word means. :(

So for me, I can read 'something-ki-so-ba'

If someone were to add pronunciation aid by 'ruby' HTML markup, I can read the full word.

```
display(HTML('<h1><ruby> <rp></rp><rt> </rt><rp></rp></ruby></h1>'))
```

```
<IPython.core.display.HTML object>
```

docs/people/marsbarlee/attachment:image.png

Oh! The word is... ya-ki-so-ba!! Delicious!

Now's let's play around with 'ruby' for other uses! Let's try using English text!

```
display(HTML('<h1><ruby> <rp></rp><rt>abdcfg </rt><rp></rp></ruby></h1>'))
```

```
display(HTML('<h1><ruby>This little piggy went to the market! <rp></rp><rt> </rt><rp>↪</rp></ruby></h1>'))
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

Now, what if we changed it to emoji, the next frontier language?

```
display(HTML('<h1><ruby>This little piggy went to the market! <rp></rp><rt> </rt><rp>
↳</rp></ruby></h1>'))
```

```
<IPython.core.display.HTML object>
```

Yay! Let's continue and finish the nursery rhyme!

```
display(HTML('<h1><ruby>This little piggy went to the market! <rp></rp><rt> </rt><rp>
↳</rp></ruby></h1>'))
```

```
<IPython.core.display.HTML object>
```

```
display(HTML('<h1><ruby>This little piggy stayed home, <rp></rp><rt> </rt><rp></rp></
↳ruby></h1>'))
```

```
<IPython.core.display.HTML object>
```

```
display(HTML('<h1><ruby>This little piggy had roast beef, <rp></rp><rt> </rt><rp></rp>
↳</ruby></h1>'))
```

```
<IPython.core.display.HTML object>
```

```
display(HTML('<h1><ruby>This little piggy had none. <rp></rp><rt> </rt><rp></rp></
↳ruby></h1>'))
```

```
<IPython.core.display.HTML object>
```

```
display(HTML('<h1><ruby>This little piggy went ... <rp></rp><rt> </rt><rp></rp></ruby>
↳</h1>'))
```

```
<IPython.core.display.HTML object>
```

```
display(HTML('<h1><ruby>Wee, wee, wee, <rp></rp><rt> </rt><rp></rp></ruby></h1>'))
```

```
<IPython.core.display.HTML object>
```

```
display(HTML('<h1><ruby>all the way home! <rp></rp><rt> </rt><rp></rp></ruby></h1>'))
```

```
<IPython.core.display.HTML object>
```

Thank you piggies and thank you for reading!

the readme challenge

technical essays

long form essays about literate programming, literate computing, and notebooks.

2.1.2 about this project layout

to date, the tools in this project are designed to work with a docs layouts.

```
_build
.github
binder
conf.py
dodo.py
noxfile.py
jupyter_nbconvert_config.py
jupyter_server_config.json
readme.md
.readthedocs.yml
```

2.2 developer notes

the documentation is build with jupyter book and sphinx

2.2.1 run in an isolated environment

this option requires nox for session management.

build the html docs

```
!nox -s docs
```

build the pdf docs

```
!nox -s docs -- pdf
```

2.2.2 running tasks in your environment

nox runs doit, doit manages files tasks, and you can use it directly.

list the tasks available.

```
!doit list
```

Build the html content


```
!doit html
```

2.3 github practices for the workshops

we've provided some github templates for [issues](#) and [pull requests](#) that should help contribute to the workshop. please submit a new template if you think one is missing.

- [github](#)
- [nbviewer](#)
- [binder](#)

2.3.1 how discussions work

we use [discussions](#) for open ended issues that don't close, for example classes and discussions

2.3.2 how submitting issues works

we use [issues](#) for tasks that can be triaged and resolved eventually, (eg new ideas, publishing improvements, submitting a document)

2.3.3 how publishing works

we aren't saying everything passes, we have standards!

1. the issue is triaged by an editor within the first **24 business hours**
2. issue is tagged
3. reviewers manual assigned
 - countdown begins 7531
 - reviewers assign themselves
 - review required
 - restart countdown
 - no review required after 7531 days
 - close and accept
1. tests pass
2. there are sufficient reviews, or reactions

issue templates

describe the topic you'd like to read about

“i'd like to read about using pandas to explore data.”

please share some relevant links that you have looked prior to submitting this idea.

- <https://pandas.pydata.org/>

please name some modules you would like to see used together

```
pandas, requests, requests_cache
```

use this template if you are inexperienced in adding work to the writers-workshop, otherwise we prefer you submit new content by making a [pull request].

describe the topic you wrote about in tweet form using #@ for attribution

i just published a program about an example that is fun because examples are fun. thanks to @example for the . #example

share links that are derived from this work (eg nbviewer, binder, github)

- <https://github.com/>
- <https://gist.github.com/>
- <https://nbviewer.jupyter.org>

please share links from other resources relevant to your work.

- <https://example.com/content>

please name some modules projects or tools used in this work

```
pandas, requests, requests_cache
```

tell us why you liked creating this work.

this work was fun because i always wanted to work with example, and i didn't know example generates examples.

i liked this work because it addresses an idea from the issues.

addresses #-1

Stages

- [] confirm the tweet content
- [] assign reviewers to triage, and comment on the document
 - [] make changes from the triage review
- [] make a pr
 - [] request changes
 - * [] address changes until no review is required
 - [] pull requested accepted and closed

pull request templates

if you are unsure about how to submit content through a pull requests, please [create an issue instead](#).

describe the topic you wrote about in tweet form using #@ for attribution

i just published a program about an example that is fun because examples are fun. thanks to @example for the . #example

share links that are derived from this work (eg nbviewer, binder, github)

- <https://github.com/>
- <https://gist.github.com/>
- <https://nbviewer.jupyter.org>

please share links from other resources relevant to your work.

- <https://example.com/content>

please name some modules projects or tools used in this work

`pandas, requests, requests_cache`

tell us why you liked creating this work.

this work was fun because i always wanted to work with example, and i didn't know example generates examples.

i liked this work because it addresses an idea from the issues.

addresses #-1

Stages

- [] confirm the tweet content
- [] assign reviewers to triage, and comment on the document
 - [] make changes from the triage review
- [] request changes
 - [] address changes until no review is required
- [] pull requested accepted and closed