Julia Programming Tips

Tom Kwong

January 19, 2019

1 Introduction

This book contains various tips for the Julia programming language. The purpose is to introduce common techniques to new Julia users so they can learn the language quickly without having to resort to help in Slack or Discourse sites.

The scope of this book is very broad, and it is largely dependent on my own programming experience and the type of use cases that I am exposed to. The subjects may range from general programming technique, parallel programming, statistical computing, GPU computing, etc etc. I also cannot claim that every idea originates from me because it's more likely that I have learned such technique from others.

1.1 Audience

The audience may be:

1. Users just started learning the Julia language
2. Experience users hoping to find new tips to add to their own toolbox
3. People curious about the Julia langage but haven't otherwise started trying it out

1.2 Credits

This book is devoted to the Julia community. I learned a lot from the experts/veterans that participate in either the JuliaLang Slack or the Discourse forum site. These people are the kindest ones that I have ever seen in other programming communities.

I would like to thank my wife, May, for keeping me relatively free of house chores so I can focus on writing this book. Likewise, I thank my kids for being independent and not having to trouble me about any of their school work.

2 Control Flows

2.1 Early Break

It's a fairly common pattern that the program breaks out of a loop when a specific condition is satisified.

```
In [3]: # Find the first value where it is greater than a threshold
        function greater_than(vec, threshold)
            idx = 1
            value = nothing
            for v ∈ vec
                if v > threshold
                    value = v
                    break
                end
                idx += 1
            end
            return value == nothing ? nothing : (idx, value)
        end;
```

```
In [5]: @show greater_than(rand(1000), 0.999)
        @show greater_than(rand(1000), 0.999)
        @show greater_than(rand(1000), 0.999)
        ;
```

```
greater_than(rand(1000), 0.999) = (17, 0.9993878165046459)
greater_than(rand(1000), 0.999) = (151, 0.9991774899756094)
greater_than(rand(1000), 0.999) = nothing
```

It is, however, more *Julian* to use the short-circuit `&&` operator to return early. Doing so makes the code more concise and easier to read.

```
In [6]: # Find the first value where it is greater than a threshold
        function greater_than(vec, threshold)
            idx = 1
            for v ∈ vec
                v > threshold && break
                idx += 1
            end
            return idx > length(vec) ? nothing : vec[idx]
        end;
```

```
In [8]: @show greater_than(rand(1000), 0.999)
        @show greater_than(rand(1000), 0.999)
        @show greater_than(rand(1000), 0.999)
        ;
```

```
greater_than(rand(1000), 0.999) = 0.9998223326994309
greater_than(rand(1000), 0.999) = nothing
greater_than(rand(1000), 0.999) = 0.9994499054436097
```

2.2 Enumerating a Collection with the Index

In the above example Early Break, we are scanning all elements of a vector and keeping tracking of an index on our own. It actually comes for free in Julia using the `enumerate` function. Here, we will also take advantage of an early return.

```
In [9]: # Find the first value where it is greater than a threshold
        function greater_than(vec, threshold)
            for (idx, v) ∈ enumerate(vec)
                v > threshold && return (idx, v)
            end
            return nothing
        end;
```

```
In [10]: @show greater_than(rand(1000), 0.999)
         @show greater_than(rand(1000), 0.999)
         @show greater_than(rand(1000), 0.999)
         ;
```

```
greater_than(rand(1000), 0.999) = (468, 0.9994336793768159)
greater_than(rand(1000), 0.999) = nothing
greater_than(rand(1000), 0.999) = (971, 0.9991035932033405)
```

2.3 Finding the First Element with a Condition

It turns out that Julia already has a function that can be used to find the first element that satisfies a condition.

```
In [11]: function greater_than(vec, threshold)
             idx = findfirst(x -> x > threshold, vec)
             return idx == nothing ? nothing : vec[idx]
         end;
```

```
In [12]: @show greater_than(rand(1000), 0.999)
         @show greater_than(rand(1000), 0.999)
         @show greater_than(rand(1000), 0.999)
         ;
```

```
greater_than(rand(1000), 0.999) = nothing
greater_than(rand(1000), 0.999) = 0.999040825523396
greater_than(rand(1000), 0.999) = nothing
```

Related Functions

The `find*` functions all take a predicate function:

- `findall`: find all elements that satify a predicate condition
- `findlast`: find the last element that satify a predicate condition

3 Testing

This notebook is used for testing purpose only

3.1 Equations

Equations are cool in jupyter cells. How do they render in PDF?

3.1.1 Example 1

$$\frac{d}{dx}\left(\int_0^x f(u)\,du\right) = f(x) \tag{2.1}$$

See equation (2.1)
See reference 2.1

3.1.2 Example 2

$$\frac{1}{\left(\sqrt{\phi\sqrt{5}}-\phi\right)e^{\frac{2}{5}\pi}} = 1+\frac{e^{-2\pi}}{1+\frac{e^{-4\pi}}{1+\frac{e^{-6\pi}}{1+\frac{e^{-8\pi}}{1+\ldots}}}}$$

See (3.1.2)

3.2 References

See eq1 (2.1)
See eq2 (3.1.2)

$$a = 1 \tag{1}$$

See (1)
See (1)