present

Lecture 1: Getting Started

Kevin Bonham, PhD

2021-06-08

```
Lessons = [1,2]
```

What is a computer program?

```
hello (generic function with 1 method)
```

```
function hello(x)
    return "Hello, $(x)!"
end
```

```
"Hello, world!"
```

```
hello("world")
```

What makes programming languages challenging to learn?

- Programming languages are **literal**

```
52
```

```
42 + 10
```

```
MethodError: no method matching +(::Int64, ::String)
Closest candidates are:
+(::Any, ::Any, !Matched::Any, !Matched::Any...) at operators.jl:560
+(::T, !Matched::T) where T<:Union{Int128, Int16, Int32, Int64, Int8, UInt128, UInt16,
+(::Union{Int16, Int32, Int64, Int8}, !Matched::BigInt) at gmp.jl:534
...

  1. top-level scope  @  Local: 1   [inlined]
```

```
5 + "2"
```

- Programming languages are **procedural**

```
11
```

```
let
    foo(x) = x + 1
    foo(10)
end
```

```
UndefVarError: bar not defined

  1. top-level scope  @  Local: 2
```

```
let
    bar(10)
    bar(y) = y + 2
end
```

Programs (algorithms) are just things and actions

- "Things" in computer code are data
- "Actions" in computer code are generally called "functions"
- Real life is filled with algorithms

Question: What are some algorithms you run in real life?

What are "essential" skills?

- How do I think about writing a computer program?
- When the code I've written has an error, what steps do I take to debug it?
- How do I keep track of the code that I've written?
- How do I get help when I'm stuck?

What are *not* essential skills:

- syntax specific to any programming language (even julia!)
- anything that you can google (though knowing *how* to google is!)

Who are you?

Kevin

PhD in Immunology, but now working as a computational biologist, studying the human microbiome and its effect on cognitive development in kids. Senior Research Scientist at Wellesley.

Married to **Rachel Rynick**, have a 2 year old son (Isaiah).

Course Components

- **Zulip chat**; all course communication will happen here.
- Free online textbook, ***Think Julia***
- **BISC 195 course website**
  - "Lessons" contain additional written content, and links to other components
- Scheduled course times (Tu/F) will be mix of "lecture" and "lab"
  - Lectures are what we're doing now!
  - Labs will be a mix of activities, pair-programming, and chances to work on assignments
- "Assignments" are due ~ 2 / week, and are the primary source of your grade
  - Submitted / auto graded through **github classroom**
- A "Final Project" will be designed and built in the last 2 weeks of class.

An example of what's coming

(you'll be able to do all of this in a few weeks)

What's the reverse complement of:

ATTCGGGAC

```
"GTCCCGAAT"
```

Lab 1 - Install stuff

In principle, you should have already done this, but life gets busy! Before we're done here, you should have:

- If you're a windows user, installed WSL2
- installed julia and VS Code
- (optional) Mac users, if you finish other stuff, **install git**

Utils (you can ignore the stuff below)

```
using PlutoUI
```

```
using Test
```

```
Test Summary: | Pass  Total
Some tests    |    2      2
```

```
with_terminal() do
    @testset "Some tests" begin
        @test 1+1 == 2
        @test_throws ErrorException error()
    end
end
```

```
dna_complements =  (a = 't',  c = 'g',  g = 'c',  t = 'a')
```

```
dna_complements = (
    a = 't',
    c = 'g',
    g = 'c',
    t = 'a')
```

```
complement (generic function with 1 method)
```

```
complement(base::Symbol) = uppercase(dna_complements[base])
```

```
complement (generic function with 2 methods)
```

```
complement(base::Char) = complement(Symbol(lowercase(base)))
```

```
complement (generic function with 3 methods)
```

```
complement(sequence::AbstractString) = string((complement(sequence[i]) for i in
eachindex(sequence))...)
```

```
reverse_complement (generic function with 1 method)
```

```
reverse_complement(sequence) = reverse(complement(sequence))
```