

“Looks like Python, runs like C”

- a quick introduction to Julia for Python users



#SISTEM2021 Conference

MIT julia

Spring 2021 | MIT
18.S191/6.S083/22.S092

Introduction to Computational Thinking

by Alan Edelman, David P. Sanders & Charles E. Leiserson

Welcome

Logistics

Software Installation

Cheatsheets

COURSE CONTENT

Introduction to Computational Thinking

Welcome to MIT 18.S191 aka 6.S083 aka 22.S092, Spring 2021 edition!

This is *Spring 2021*. For *Fall 2020*, see our [old website](#).

This is a *preview* of the new course material. Come back soon for the first lecture!

This is an introductory course on Computational Thinking. We use the [Julia programming language](#) to approach real-world problems in varied areas applying data analysis and computational and mathematical modeling. In this class you will learn computer science, software, algorithms, applications, and mathematics as an integrated whole.

Topics include:

- Image analysis



YouTube

Search

Optimization Methods for Machine Learning and Engineering (KIT Winter Term 20/21)

33 videos • 1,865 views • Updated 7 days ago

1.1 Optimization Methods - Motivation and Historical Perspective
Julius Pfrommer
27:58

1.2 Optimization Methods - Course Organization
Julius Pfrommer
6:20

1.3 Optimization Methods - Notation and Analysis Refresher
Julius Pfrommer
9:49

1.4 Optimization Methods - Convexity
Julius Pfrommer
12:49

1.5 Optimization Methods - Gradient Descent

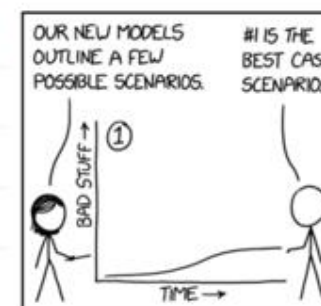
Introduction to Computational Thinking with Julia, with Applications to Modeling the COVID-19 Pandemic

COURSE HOME

SYLLABUS

COURSE MATERIALS

ASSIGNMENTS



It's important to consider a variety of possible scenarios when looking at data to potentially predict the future. (Courtesy of [Randall Munroe](#). License: CC-BY-NC.)

Instructor(s)

Prof. Alan Edelman

Prof. David P. Sanders

MIT Course Number
18.S191 / 6.S083

As Taught In
Spring 2020

Level

Undergraduate

[CITE THIS COURSE](#)

- Since v.1.0 in 2018 an increasing number of courses, books and institutions using Julia.



Mosè Giordano @MoseGiordano · Jan 23, 2021

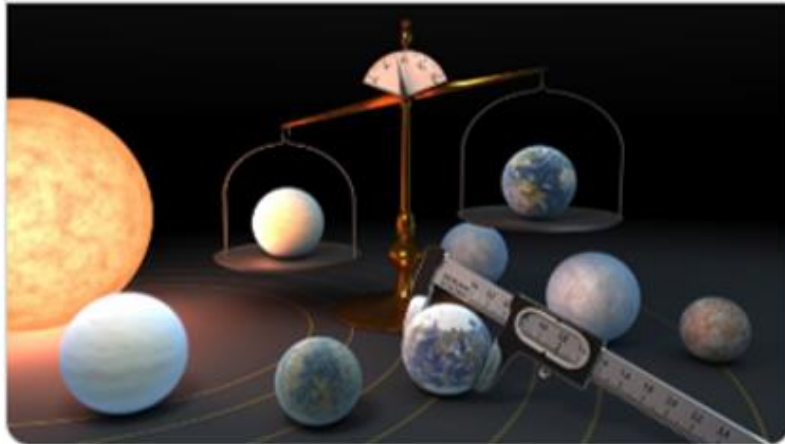
And the analysis was done in #JuliaLang 😊



NASA Exoplanets @NASAExoplanets

The TRAPPIST-1 star is home to the largest batch of roughly Earth-size planets ever found. Now we know even more about the densities of these seven rocky worlds.

And the composition of the TRAPPIST-1 planets could be notably different than Earth's! go.nasa.gov/3c78qST



Eric Agol
@AgolEric

Yes - I don't think I could have written & debugged the code so quickly in FORTRAN or C, or debugged it so quickly in IDL or Python.

4:29 AM · Jan 23, 2021



♥ 21 💬 5 🔗 Copy link to Tweet

My favorite slide...

A quick warning:

- Julia does not have classes, strictly speaking.
- Compared to Python, this means more use of functions that are not built as part of a class/object.
- E.g. some operations on strings: Julia vs Python

```
test_string = "Julia is a Cool Language"  
  
findfirst("i", test_string)  
  
lowercase(test_string)  
  
occursin("Julia", test_string)
```

```
test_string = "Python is a Cool Language"  
  
test_string.find("y")  
  
test_string.lower()  
  
print("Python" in test_string)
```

Some primitive operations: Julia vs Python

```
parse{Int64, "42"}  
parse{Float64, "123.345"}  
string(123)
```

```
int("42")  
float("123.345")  
str(1210)
```

```
println(typeof('a'))  
println(typeof("a"))
```

```
Char  
String
```

- Converting Strings to Integers or Floats is a little different to Python. We use `parse()` and pass the type we'd like to convert our string to. Conversion to string is much the same as Python.
- **Note:** Single quotation marks are not strings in Julia, instead they are individual characters (or "chars").

Arrays and Dictionaries:

```
arr1 = [3, 5, 9]
length(arr1)
maximum(arr1)
```

```
arr2 = ["Hello", 21, 4.2]
push!(arr2, "34")
println("Newly appended array: ", arr2)

arr3 = [3, 5, -1]
sort!(arr3)
println("Sorted array: ", arr3)
```

```
Newly appended array: Any["Hello", 21, 4.2, "34"]
Sorted array: [-1, 3, 5]
```

```
phonebook = Dict{"John" => "555", "Alice" => "111", "Bob" => "333"}

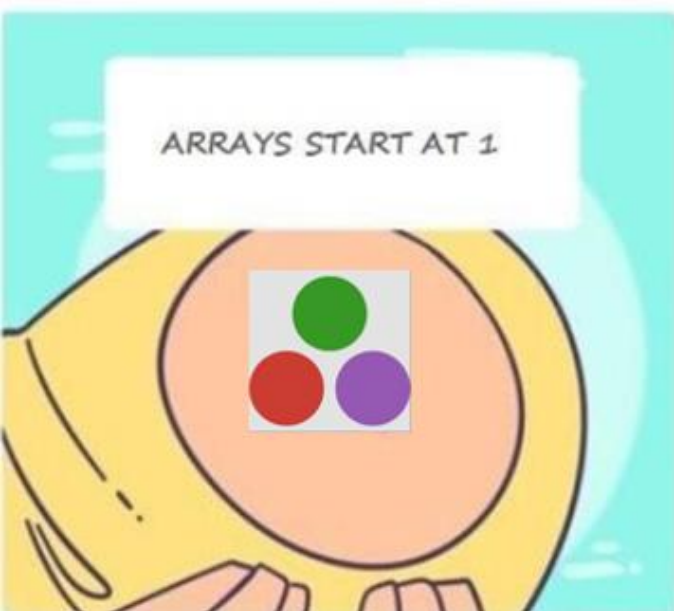
# We can retrieve keys using the keys() function. Ditto for values
println("Keys: ", keys(phonebook))
println("Values: ", values(phonebook))
```

- Some operations are much the same as in Python, i.e. max, len etc.
- However, lists are added to using `push!`. Remember: there are no built-in methods like `list.append()`. Sorting behaves similarly: sorting the list in-situ.
- Tuples and dictionaries behave similarly, again using functions over methods.
- In some ways, the behavior is more consistently applied than in Python.



But most importantly....

Arrays start at 1!!!



```
letters = ["a", "b", "c", "d"]  
println(letters[1:end])  
println(letters[2:3])
```

```
["a", "b", "c", "d"]  
["b", "c"]
```

Functions & Loops:

```
function welcome(name)
    println("Hello $name")
end
welcome("Katya")

f(x, y) = println(x * y)
f(3, 5)
```

Hello Katya
15

```
for i = 1:5, j = 1:5
    println(i, " : ", j)
end

i = 1
while i <= 5
    println(i)
    i = i + 1
end
```

- Julia allows for very minimal one liner functions. More complex ones are similar to Python, but with an end block.
- Loops follow a similar convention of minimal syntax.

A study in Julia vs Python: a part of a statistical spelling corrector

```
words(text::String) = collect(m.match for m in eachmatch(r"\w+", text))

WORDS = countmap([word for word in words(x)])

"""Probability of a given 'word'."""
P(word::String, N::Float64 = sum(values(WORDS))) = WORDS[word] / N
```

- Julia

```
def words(text): return re.findall(r'\w+', text.lower())

WORDS = Counter(words(open(x).read()))

def P(word, N=sum(WORDS.values())):
    """Probability of `word`."""
    return WORDS[word] / N
```

- Python

References:

- Getting started with Julia: <https://julialang.org/learning/>
- Optimisation methods for Machine Learning and Engineering:
<https://www.youtube.com/playlist?list=PLdkTDauaUnQpzuOCZyUUZc0lxf4-PXNR5>
- MIT Introduction to Computational Thinking with Julia:
<https://computationalthinking.mit.edu/Fall20/>
- GitHub repo for slides and some code:
[https://github.com/RobinsonLuzo/Talks/tree/master/SISTEM2021 Intro to Julia](https://github.com/RobinsonLuzo/Talks/tree/master/SISTEM2021%20Intro%20to%20Julia)
- Dublin Julia Users Group: <https://www.meetup.com/Dublin-Julia-Users-Group/>