

# Getting started with Julia

Michiel Stock Bram De Jaegher Daan?

December 2019

## 1 Basic computing

```
1 + 2
```

```
1.0 + 2.0
```

```
2 / 4
```

```
div(2, 4)
```

```
35 \ 7
```

```
1 // 3
```

```
'c'
```

```
:symbol
```

```
Error: type QuoteNode has no field head
```

```
x = 2
```

```
 $\tau = 1 / 37$  # fine structure constant
```

```
0.02702702702702703
```

```
3x
```

```
x += 2 # inplace update of x
```

```
4
```

```
mystery = "life, the universe and everything"
```

```
"life, the universe and everything"
```

```
println("The answer to $mystery is $(3*2*7)")
```

```
The answer to life, the universe and everything is 42
```

## 2 Boolean operators

```

# Boolean operators
!true # => false
!false # => true
1 == 1 # => true
2 == 1 # => false
1 != 1 # => false
2 != 1 # => true
1 < 10 # => true
1 > 10 # => false
2 <= 2 # => true
2 >= 2 # => true
# Comparisons can be chained
1 < 2 < 3 # => true
2 < 3 < 2 # => false

```

```
false
```

### 3 Control flow

```

if 4 > 3
  println("A")
elseif 3 > 4
  println("B")
else
  println("C")
end

```

```
A
```

```
y = condition ? valueiftrue : valueiffalse
```

### 4 Looping

```
characters = ["Harry", "Ron", "Hermione"]
```

```

for char in characters
  println("Character $char")
end

```

```

Character Harry
Character Ron
Character Hermione

```

```

for (i, char) in enumerate(characters)
  println("$i. $char")
end

```

```

1. Harry
2. Ron
3. Hermione

```

```
pets = ["Hedwig", "Pig", "Crookhanks"]
```

```

for (char, pet) in zip(characters, pets)
  println("$char has $pet as a pet")
end

```

Harry has Hedwig as a pet  
Ron has Pig as a pet  
Hermione has Crookhanks as a pet

1675767616  
837883808  
418941904  
209470952  
104735476  
52367738  
26183869  
78551608  
39275804  
19637902  
9818951  
29456854  
14728427  
44185282  
22092641  
66277924  
33138962  
16569481  
49708444  
24854222  
12427111  
37281334  
18640667  
55922002  
27961001  
83883004  
41941502  
20970751  
62912254  
31456127  
94368382  
47184191  
141552574  
70776287  
212328862  
106164431  
318493294  
159246647  
477739942  
238869971  
716609914  
358304957  
1074914872  
537457436  
268728718  
134364359  
403093078  
201546539  
604639618  
302319809  
906959428  
453479714  
226739857  
680219572  
340109786  
170054893

510164680  
255082340  
127541170  
63770585  
191311756  
95655878  
47827939  
143483818  
71741909  
215225728  
107612864  
53806432  
26903216  
13451608  
6725804  
3362902  
1681451  
5044354  
2522177  
7566532  
3783266  
1891633  
5674900  
2837450  
1418725  
4256176  
2128088  
1064044  
532022  
266011  
798034  
399017  
1197052  
598526  
299263  
897790  
448895  
1346686  
673343  
2020030  
1010015  
3030046  
1515023  
4545070  
2272535  
6817606  
3408803  
10226410  
5113205  
15339616  
7669808  
3834904  
1917452  
958726  
479363  
1438090  
719045  
2157136  
1078568

539284  
269642  
134821  
404464  
202232  
101116  
50558  
25279  
75838  
37919  
113758  
56879  
170638  
85319  
255958  
127979  
383938  
191969  
575908  
287954  
143977  
431932  
215966  
107983  
323950  
161975  
485926  
242963  
728890  
364445  
1093336  
546668  
273334  
136667  
410002  
205001  
615004  
307502  
153751  
461254  
230627  
691882  
345941  
1037824  
518912  
259456  
129728  
64864  
32432  
16216  
8108  
4054  
2027  
6082  
3041  
9124  
4562  
2281  
6844

3422  
1711  
5134  
2567  
7702  
3851  
11554  
5777  
17332  
8666  
4333  
13000  
6500  
3250  
1625  
4876  
2438  
1219  
3658  
1829  
5488  
2744  
1372  
686  
343  
1030  
515  
1546  
773  
2320  
1160  
580  
290  
145  
436  
218  
109  
328  
164  
82  
41  
124  
62  
31  
94  
47  
142  
71  
214  
107  
322  
161  
484  
242  
121  
364  
182  
91  
274

137  
412  
206  
103  
310  
155  
466  
233  
700  
350  
175  
526  
263  
790  
395  
1186  
593  
1780  
890  
445  
1336  
668  
334  
167  
502  
251  
754  
377  
1132  
566  
283  
850  
425  
1276  
638  
319  
958  
479  
1438  
719  
2158  
1079  
3238  
1619  
4858  
2429  
7288  
3644  
1822  
911  
2734  
1367  
4102  
2051  
6154  
3077  
9232  
4616  
2308

1154  
577  
1732  
866  
433  
1300  
650  
325  
976  
488  
244  
122  
61  
184  
92  
46  
23  
70  
35  
106  
53  
160  
80  
40  
20  
10  
5  
16  
8  
4  
2

## 5 Functions

```
function square(x)
    result = x * x
    return result
end
```

```
square(2)
```

```
square(2.0)
```

```
square("ni")
```

```
"nini"
```

```
s(x) = x * x
```

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

```
s([1, 2, 3, 4, 5])
```

```
s.([1, 2, 3, 4, 5])
```

```
safelog(x, offset=0.1; base=10) = log(x + offset) / log(base)
```

```
safelog(0)
```



```

safelog(0, 0.01)

safelog(0, 0.01, base=2)

-6.643856189774724

?sort

my_unsorted_list = [4, 5, 9, 7, 1, 9]

sort(my_unsorted_list)

my_unsorted_list

6-element Array{Int64,1}:
 4
 5
 9
 7
 1
 9

sort!(my_unsorted_list)

my_unsorted_list

6-element Array{Int64,1}:
 1
 4
 5
 7
 9
 9

```

## 6 Plotting

```

using Plots

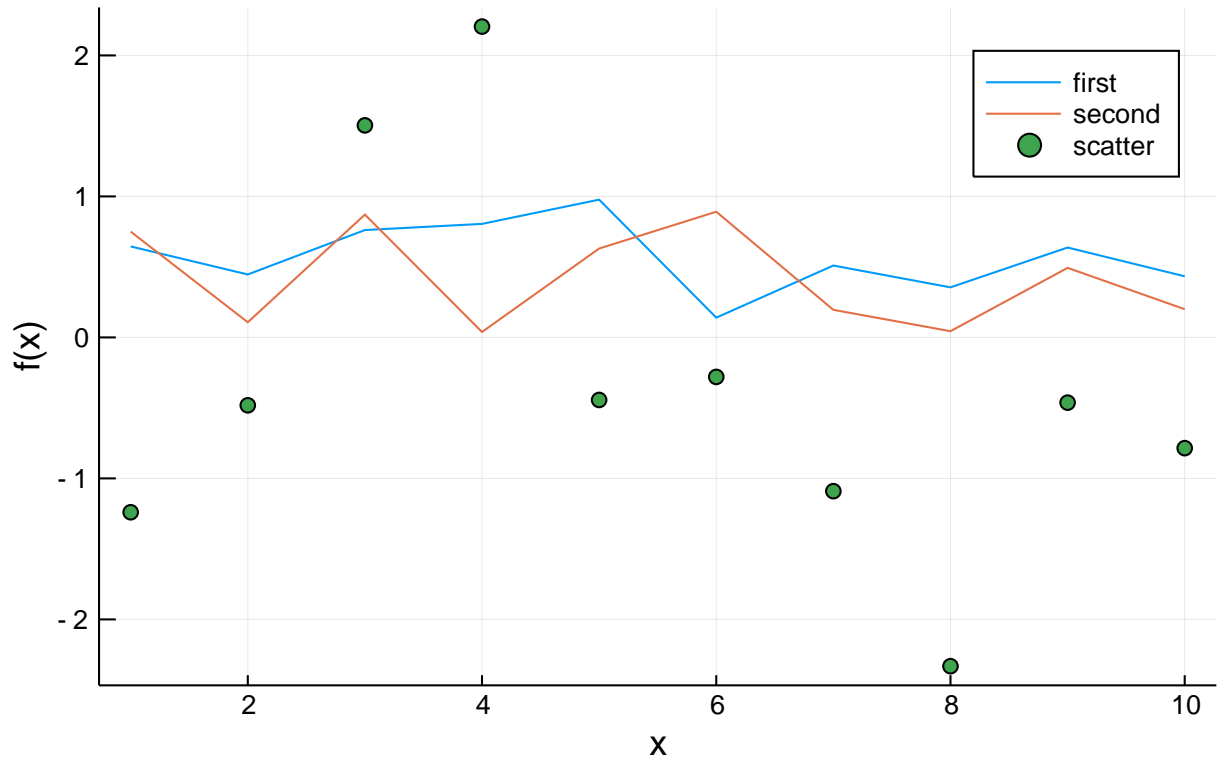
plot(1:10, rand(10), label="first")
plot!(1:10, rand(10), label="second")

scatter!([1:10], randn(10), label="scatter")

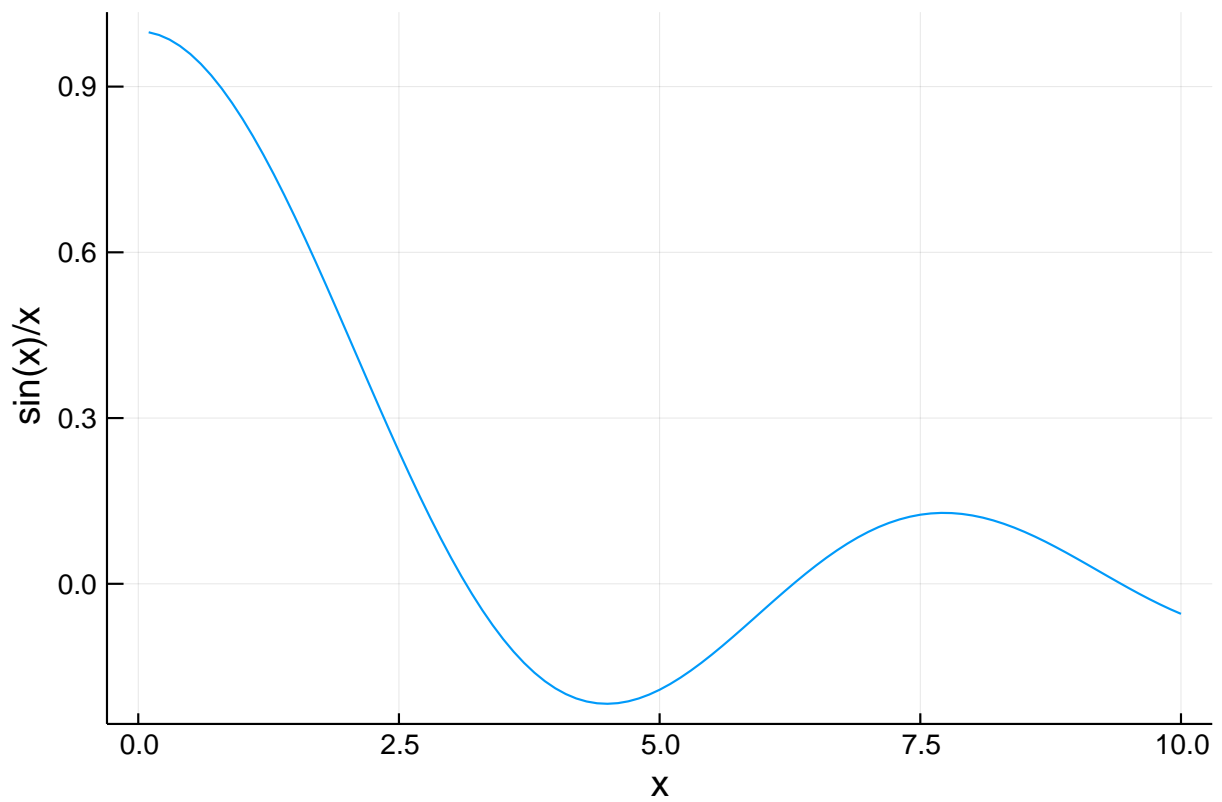
xlabel!("x")
ylabel!("f(x)")
title!("My pretty Julia plot")

```

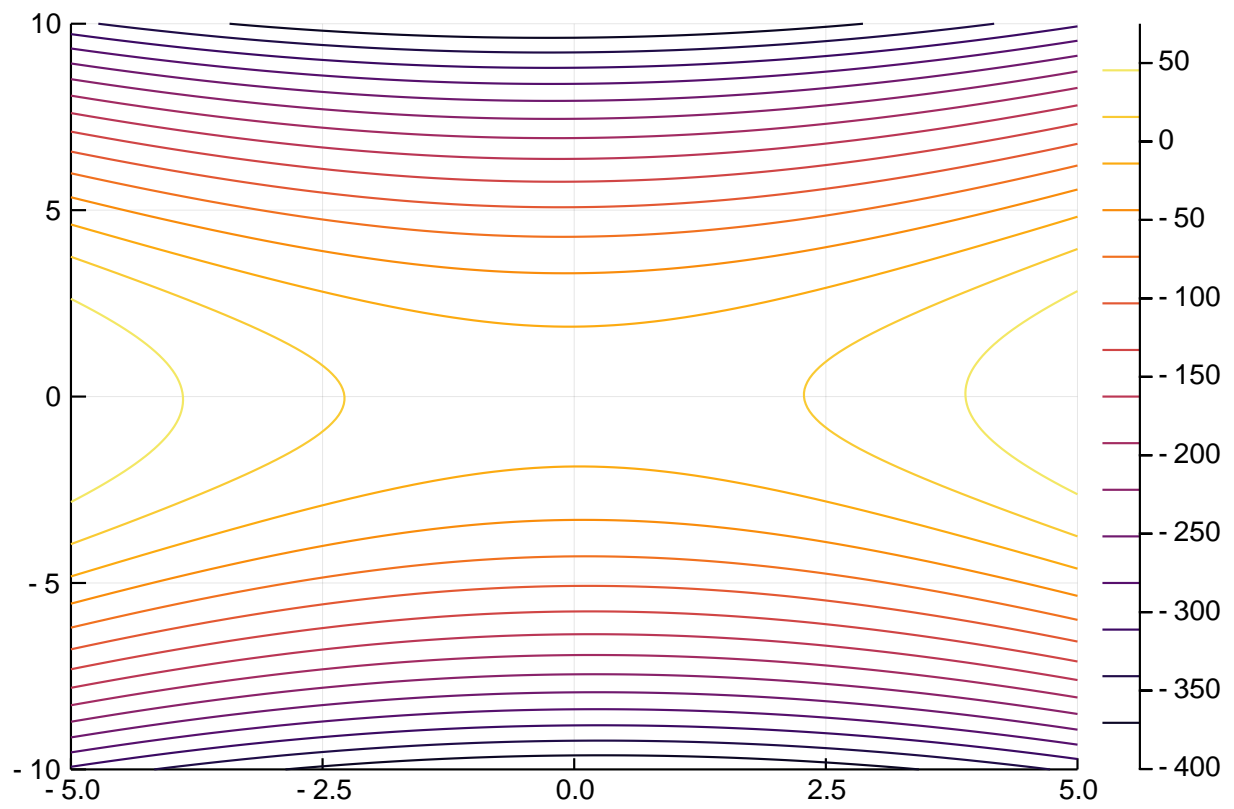
## My pretty Julia plot



```
plot(0:0.1:10, x -> sin(x) / x, xlabel="x", ylabel="sin(x)/x", legend=:none)
```



```
contour(-5:0.1:5, -10:0.1:10, (x, y) -> 3x^2-4y^2 + x*y/6)
```



## 7 Exercise