

Introduction to Julia

ENGR108
Stanford University

April 3, 2023

What is Julia?

a new programming language for scientific computing

- ▶ developed by a group mostly from MIT
- ▶ fully open source, *i.e.*, free
- ▶ convenient syntax for building math constructs like vectors, matrices, etc.
- ▶ super fast

Setting up Julia

we support two options for coding in Julia for the class

- ▶ run Julia within Ed
 - automatically works
- ▶ run Julia locally
 - faster and more flexible

Data in Julia

- ▶ scalars and numbers: *e.g.*, 1.23, -135, 3.77e-7
- ▶ booleans: true or false
- ▶ strings: *e.g.*, "Hello, world!"

Setting variables

- ▶ assignments use the = operator

```
value = 5.0
```

```
name = "Bob"
```

Basic arithmetic and mathematical functions

- ▶ +, -, *, / operators

a = 4.0

b = 7.0

c = (b - a) * (b + a) / a

- ▶ exponentiate using ^

a = 2 ^ 10

- ▶ all the usual math functions, like exp, sin, ...

result = exp(-2.33) * cos(22 * 180 / pi)

Boolean expressions

- ▶ evaluate to true or false
- ▶ use the ==, !=, <, >, <=, >= operators

```
value = 4
```

```
value == 4
```

```
value == "4"
```

```
value > 9.0
```

```
value <= 5.3
```

- ▶ flip the value of a boolean expression using !

```
!(value == "4")
```
- ▶ combine boolean expressions using && and ||

```
(value == 4) && (value == "4")
```

```
(value == 4) || (value == "4")
```

If/else statements

- ▶ test if a boolean expression is true or false and run different code in each case

```
if (value < 5)
  value = 10
else
  value = 20
end
```

- ▶ can split the code into more than two cases

```
if (value < 5)
  value = 10
elseif (value == 5)
  value = 15
else
  value = 20
end
```


Ranges

- ▶ create a sequence of numbers using :
 - ▶ the sequence includes the endpoints
1:5
 - ▶ optional middle argument gives increment (default is 1)
1:0.1:10
 - ▶ to view a range, call `collect(1:0.1:5)`

Lists

- ▶ create a numbered list of objects of different types using `[]`, e.g.,
`my_list = ["a", 1, -0.76]`
- ▶ can access the i th element of the list using `[i]`
`my_list[2] + my_list[3]`
- ▶ unlike many other programming languages, Julia indexes start at 1
`my_list[1]` # first element of the list
`my_list[0]` # issues an error
- ▶ access from the end of a list using `end`
`my_list[end]` # last element of the list
`my_list[end - 1]` # second to last element
- ▶ use `length` to find how long the list is
`length(my_list)`

For loops

- ▶ executes a code block multiple times
- ▶ most common construction involves looping over a range

```
value = 0
for i in 1:10
    value += i # short for value = value + i
end
```

- ▶ or you can loop over a list

```
value = 0
my_list = [1, 2, 3, 4, 5]
for i in my_list
    value += i
end
```

Functions

- ▶ a chunk of code that can be run over and over, e.g.,

```
println("Hello, World!")  
println("How are you doing?")  
println(49876)
```
- ▶ functions can take arguments, e.g., `println` prints its argument
- ▶ functions can return a value, which can be stored in a variable

```
length_of_list = length(my_list)
```
- ▶ functions can have a side effect (*i.e.*, do something), e.g., `println` prints something to the screen

Some important functions

- ▶ quit Julia: `quit()`
- ▶ generate a random number between 0 and 1: `rand()`

Other resources

- ▶ these slides only scratch the surface of the features of Julia
- ▶ more tutorials can be found here:
<http://julialang.org/teaching/>