Introduction to Julia

based on
https://stanford.edu/class/engr108/julia.html
https://syl1.gitbook.io/julia-language-a-concise-tutorial/language-core/func

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

Data Types

- Int64, Float64, Char, Bool...: *e.g.*, 1.23, -135, 3.77e-7, 'a', true
- String: length(s), string(123), s1 * s2, split(s), join([s1,s2])
- Vector: [1,2,3], Int64[1,2,3], [1,'a',"hi"]
  - length(a); push!(a,'6'); pop!(a);
  - pushfirst!(a,0); popfirst!(a)
  - deleteat!(a,2); append!(a1,a2);
  - b = collect(1:2:10)
  - a[1] = 8;
- Matrix:[1 2; 3 4]
  - size(m), transpose(m)
  - a[1,2] = 6
- Tuple: (1,2,3)
  - t[1] = 6, [t...]

Data Types (Cont.)

- Dict:
  - d = Dict('a'=>1, 'b'=>2, 'c'=>3)
  - println(d[‘a’])
  - println(get(d,’e’,0))
  - keys(d)
  - values(d)
- Set:
  - s = Set([1,2,3])
  - push!(s,4)
  - pop!(s)

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 an 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

Defining Functions

```
function foo(n)
  f = y -> n + x + y
  x = 1
  f(2)
end

julia> foo(10)
13
```

Some important functions

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

Higher-Order Functions

- x -> x^2
- map(x -> x^2, [1:5])
- map(/, [16, 9, 4], [8, 3, 2])
- filter(isprime, [1:50])
- reduce(/, 1:4)
- mapreduce(x -> x^2, +, [1:5])

Other resources

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