Disadvantages of Julia

Julia for ORNL Science Workshop

Monday, July 18, 2022

Gavin Wiggins

wigginsg@ornl.gov

https://gavinw.me

Memory usage

- A basic "hello world" example in Julia uses more memory than an equivalent Python example
- Idle memory usage for Julia = 90 MB, Python = 5.1 MB

Julia memory usage = 89 MB

```julia
sleep(5)
println("hello world")
```

Python memory usage = 5 MB

```python
import time

time.sleep(5)
print('hello world')
```

Results from a MacBook Pro with 2.6 GHz CPU with 32 GB RAM running macOS 12.4

Barebones testing

- Julia's built-in Test package is lacking features
- No third-party Julia packages like Python's pytest package
- Poor testing integration with GitHub/GitLab CI

Limited ecosystem

- Julia offers fewer scientific packages compared to other languages (Python, R)
- Machine learning packages for Julia are basically nonexistent or cumbersome to use
- Julia is still a young/immature language, only 4 years since Julia v1.0 was released

Compile time latency

- Invoking Julia from command line is slow due to code compilation, command line workflow not feasible
- Compile time lag is especially noticeable for plotting packages when running in REPL or command line
- Forced into REPL driven development

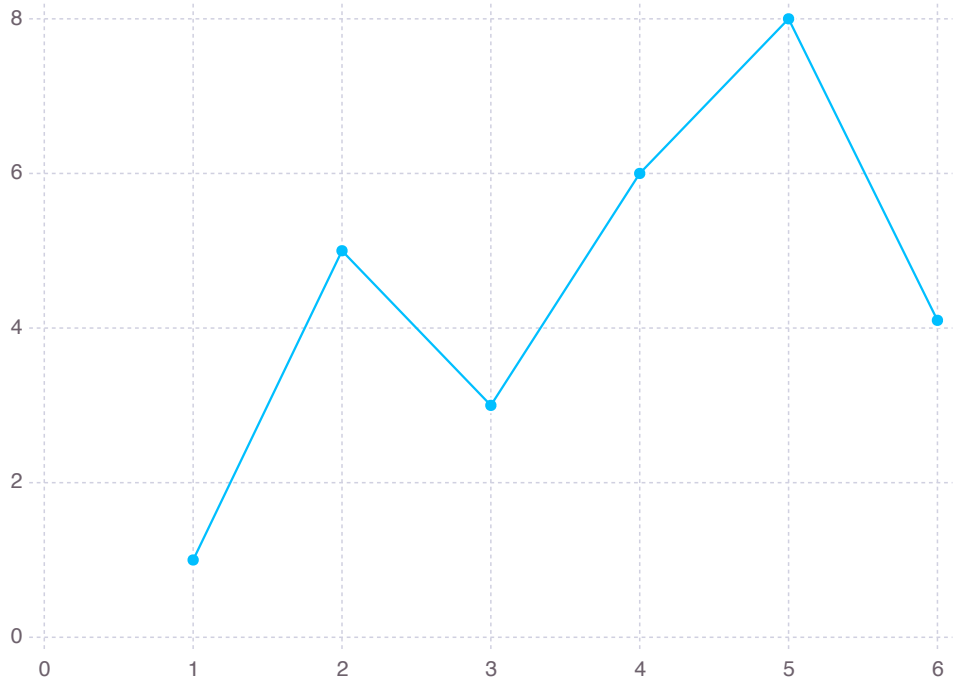
testing — testing: julia

```
~/Desktop/testing
[base ❯ julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.6.1 (2021-04-23)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> for i in 1:5
           println(i)
       end
1
2
3
4
5

julia>
```

Compile time latency

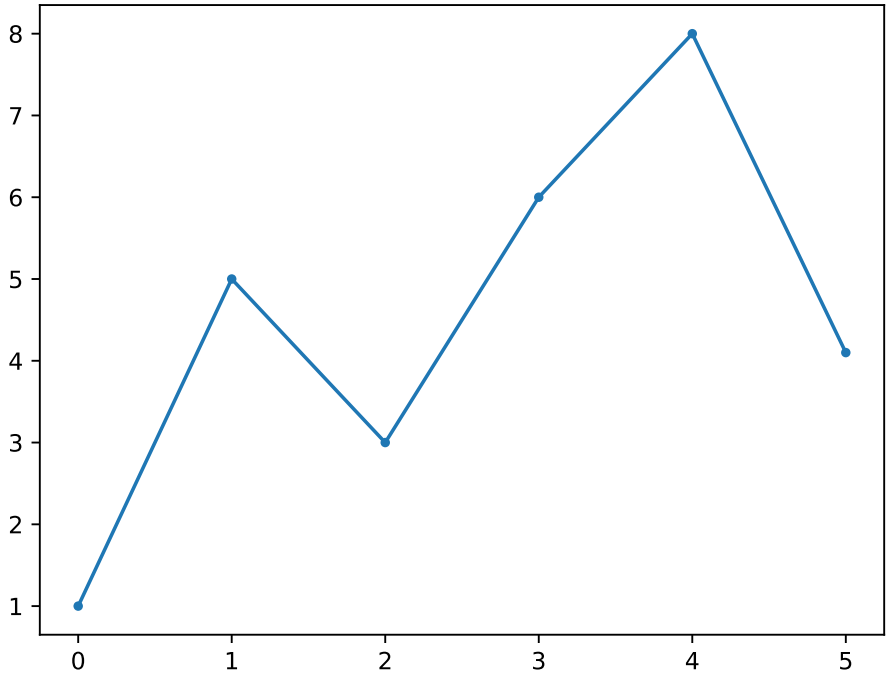
Julia execution time = 29 s

```julia
using Cairo
using Fontconfig
using Gadfly

p = plot(y=[1, 5, 3, 6, 8, 4.1], Geom.line, Geom.point)
draw(PDF("plotjulia.pdf"), p)
```



Python execution time = 1 s

```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot([1, 5, 3, 6, 8, 4.1], marker='.')

fig.savefig('plotpython.pdf')
```



Results from a MacBook Pro with 2.6 GHz CPU with 32 GB RAM running macOS 12.4

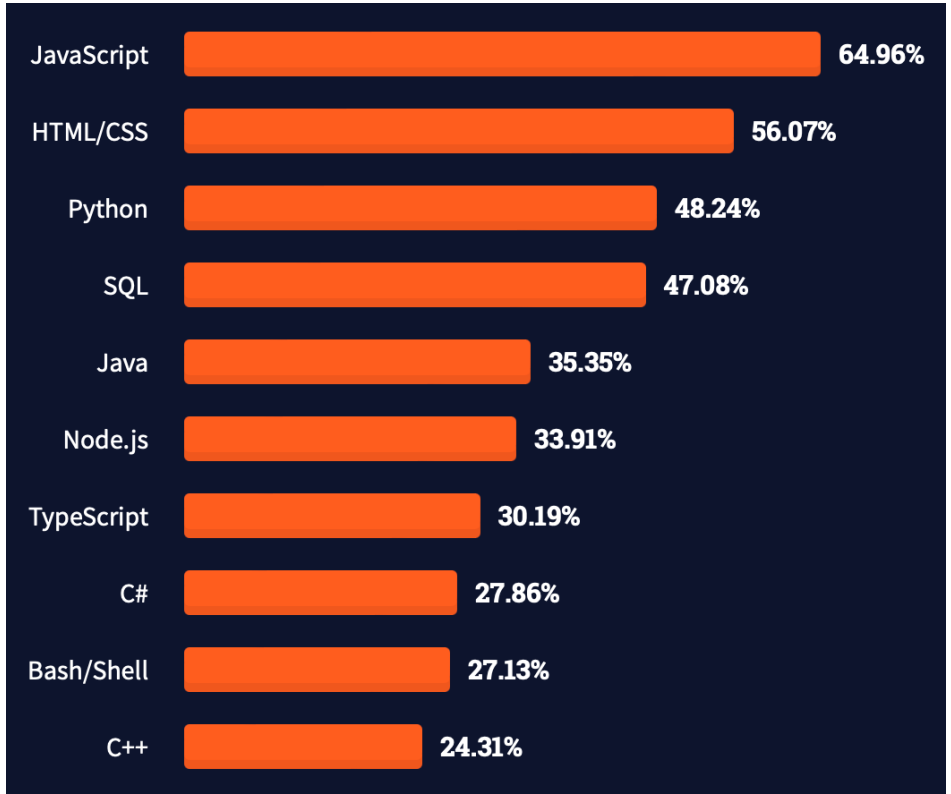
Final thoughts

- Julia is great for working with differential equations but it fails to "differentiate" itself from other languages
- Development environment is frustrating to work in

Top programming languages.
Source: TIOBE Index for July 2022.

Jul 2022	Jul 2021	Change	Programming Language	Ratings	Change
1	**3**	^	**Python**	**13.44%**	**+2.48%**
2	1	v	C	13.13%	+1.50%
3	2	v	Java	11.59%	+0.40%
4	4		C++	10.00%	+1.98%
5	5		C#	5.65%	+0.82%
6	6		Visual Basic	4.97%	+0.47%
7	7		JavaScript	1.78%	-0.93%
8	9	^	Assembly language	1.65%	-0.76%
9	10	^	SQL	1.64%	+0.11%
10	16	^^	Swift	1.27%	+0.20%

Most popular technologies.
Source: Stack Overflow 2021 Developer Survey.



Most wanted language.
Source: Stack Overflow 2021 Developer Survey.