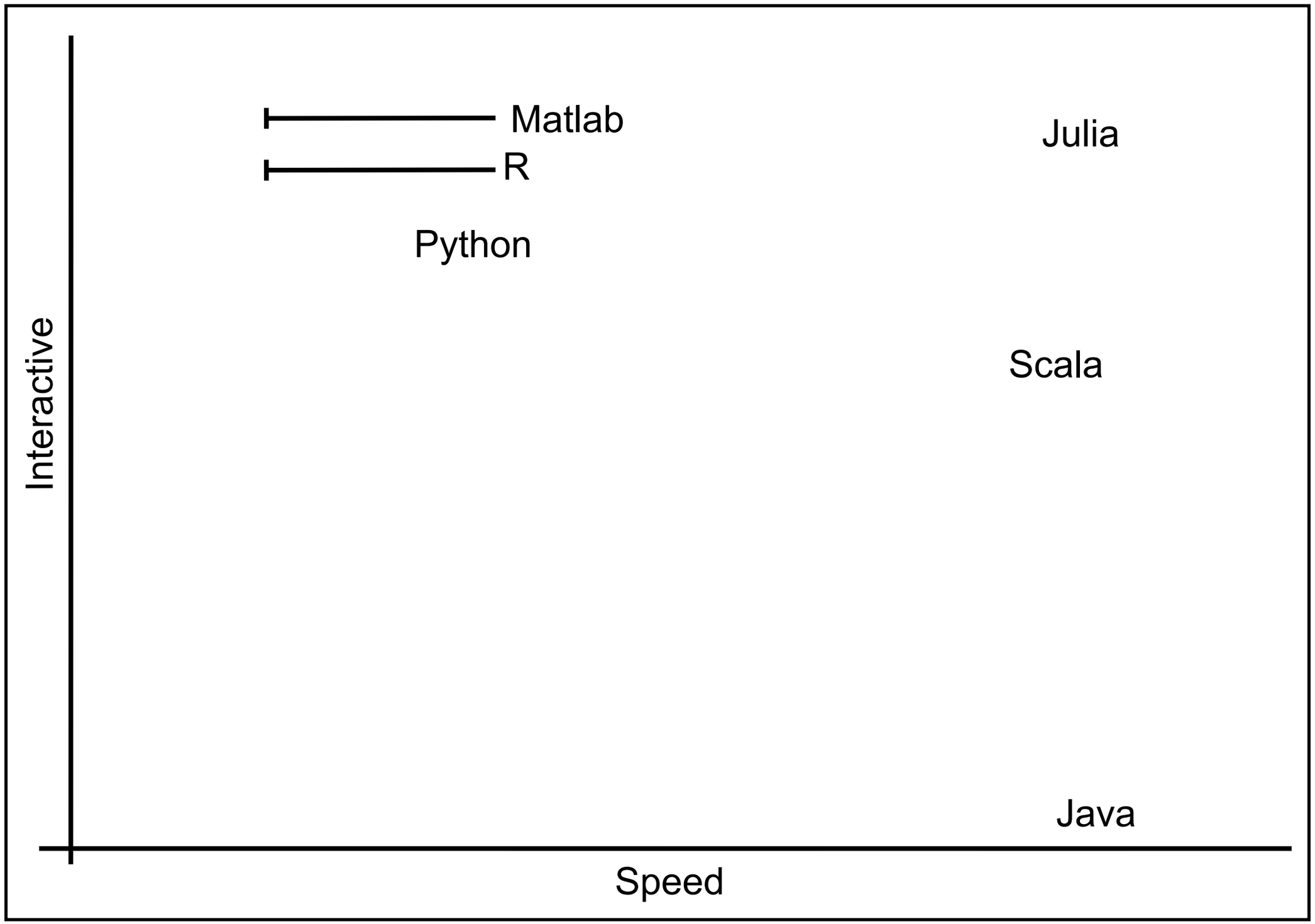
CS 696 Intro to Big Data: Tools and Methods
Fall Semester, 2016
Doc 2 Julia Introduction
Aug 30, 2016

Why Julia



Why Julia

Designed by and for people doing computational programming using current technloogy

- Interactive
- Fast
- Simple syntax
- Can call C/Fortran/Java/R/Python code
- Libaries
  - Statistics
  - ML
  - Web
  - Graphics, etc
- Lisp-like macros

Ways of Running Julia

Command line

Juno

Jupyter

Juliabox (Jupyter in the cloud)

Demo

Numerical Types

Integers

Int8 Int16 Int32 Int64 Int128 BigInt

UInt8 UInt16 UInt32 UInt64 UInt128

Floats

Float16 Float32 Float64 BigFloat

Rational

2//3

Complex

4 + 3im

```
y = 123
x = 123_345
z = 12_21
binaryX = 0b0101
octalX = 0o123
hexX = 0x1ab

typeof(y)    # Int64
```

```
z = 2.34e4
w = 1.2f2

typeof(z)   # Float64
typeof(w)  # Float32
```


+x	unary plus
-x	unary minus
x + y	binary plus
x - y	binary minus
x * y	times
x / y	divide
x \ y	inverse divide
x ^ y	power
x % y	remainder

Rational Numbers

```
3//4
5//2
1//3 + 1//5     == 8//15
1//3 + 1/5      == 0.533333333
float(2//3)     == 0.666666666
rationalize(0.5)  == 1//2
ationalize(1//3 + 1/5)  == 8//15
```

Complex Numbers

1 + 2im

(1 + 2im) + (1 - 2im) == 2 + 0im

(1 + 2im)*(2 - 3im) == 8 + 1im

real(1 + 2im) == 1

imag(1 + 2im) == 2

sqrt(1 + 2im)

sin(2 + 3im)

Converting & Rounding

```
convert(Float32, 12)            #12.0f0
convert(Rational, .333333)      # 3002396749180579//9007199254740992
convert(Complex64, 12)          # 12.0f0 + 0.0f0im
```

```
x = Int8(123)
x = BigInt(12)
y = Float16(12)
z = Int64(12.3)    #  InexactError()
```

```
round(12.3456)          123.0
```

Char

‘a’
‘\U0041’
‘\U00000097’

Int(’a’)
Char(97)
string(’a’)
isvalid(’a’)
isvalid(’\uFFFF’)

Standard C special characters

‘\t’	tab
‘\n’	newline
‘\r’	carriage return
‘\b’	backspace
‘\f’	form feed

‘A’ < ‘a’
‘A <= ‘X’ < ‘Z’
‘z’ - ‘a’ # 25

Strings

Immutable

String (abstract)
- ASCIIString
- UTF8String
- UTF16String
- UTF32String

```
"cat”
"cat\tdog”
"contains \" quote"
"""contains " quote"""

"cat" * "dog"                    # "catdog”
string("cat", "dog", 16, ‘a’) # "catdog16a"
```

```
typeof("cat")          # ASCIIString
typeof("c\u0041t")     # ASCIIString
typeof("c\uf041t")     # UTF8String
```

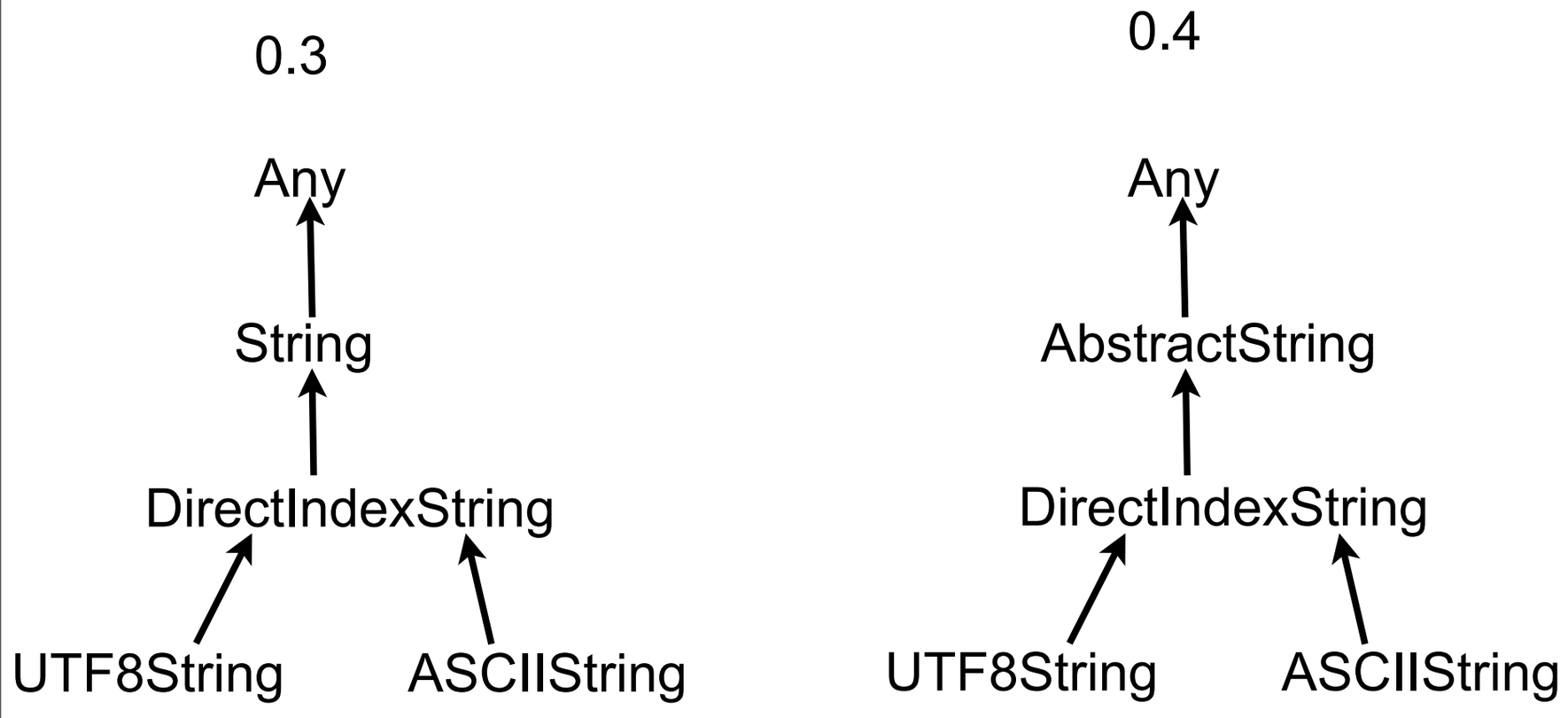
```
x = 3
y = 5
"X = $x"                 # "X = 3 "
"X + Y = $(x + y)"       # "X + Y = 8"
```

String - Julia 0.3, 0.4 & 0.5

Julia 0.3 - String was abstract type

Julia 0.4 - String was deprecated
Replaced with AbstractString

Julia 0.5 - String becomes concrete type



Converting

```
string(123)         == "123"
string(12.34)       == "12.34"
int("123")          == 123
int("12.3")         Error
float("12")         12.0

bin(3)                  == "11"
bin(3,5)            == "00011"
oct(11)             == "13"
hex(12)             == "c"
```

```
x = 123
"$x"                == "123"
```

```
parse(Int,"100",2)          == 4
parse(Int,"100",16)         == 256
```

```
@printf("%3.2f\n", 12.4)
@printf "%3.2f\n" 12.4
```

String Indexing & Functions

```
str = "Hello World"

str[1]                  # Char H

str[4]                  # Char l

str[end]            # Char d

str[end-1]          # Char l

str[2:5]            # “ello”

str[2:2]            # “e”

str[2]                  # Char e

length(str)         # 11

endof(str)          # 11
```

Unicode

పన్నెండు = 12

Δ = 2*π

s = "\u2200 x \u2203 y" # “∀ x ∃ y”

s[1] # ‘∀’
s[2] # UnicodeError: invalid character index
s[3] # UnicodeError: invalid character index
s[4] # ‘ ‘
s[5] # ‘x’
length(s) # 7 there are 7 unicode characters in the string
endof(s) # 11 But the 7 characters take 11 char locations
s[11] # ‘y’

```
for c in s
  println(c)
end
```

∀

x

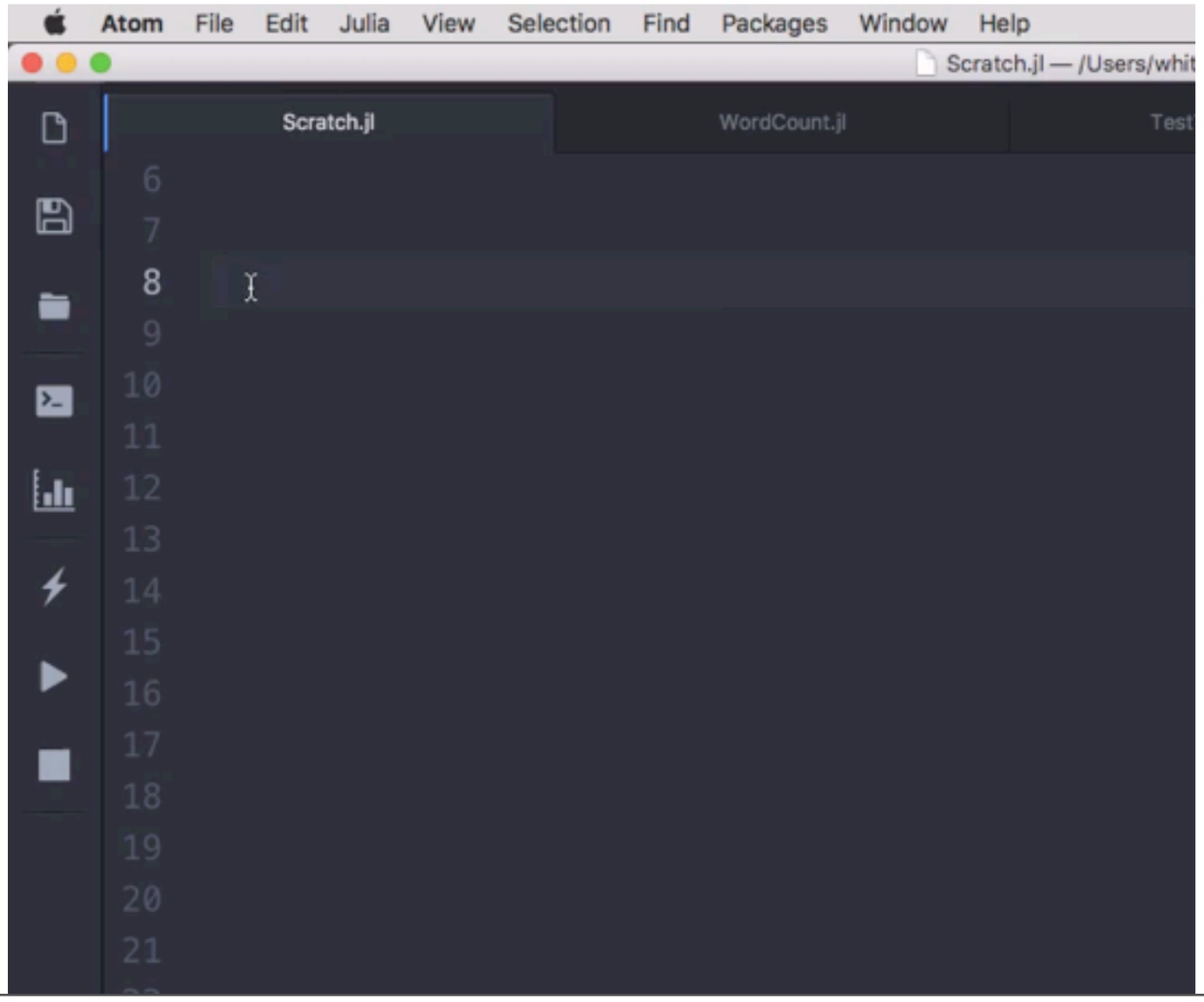
∃

y

How to Enter Unicode Characters

OS and Editor dependent

Atom/Juno autocompletes some standard math symbols
Type \ then english word for symbol



Non-Standard String Literals

Versions	v”0.4.6”
Regular Expressions	r"^\s*(?:#|$)"
Byte Literals	b"Aa\xff"

```
if VERSION < v"0.4.6"
  print("old")
else
  print("current")
end
```

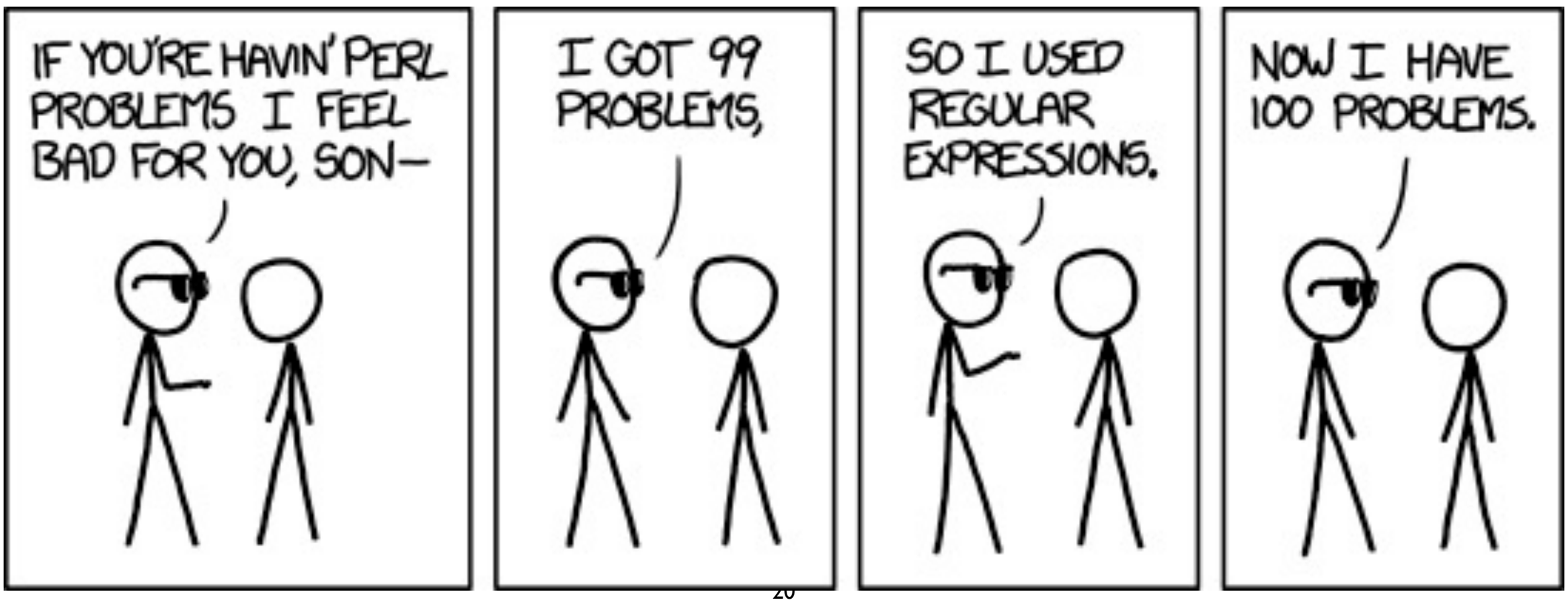
Regular Expressions

Perl-compatible regular expressions

PCRE library

Syntax

http://www.pcre.org/current/doc/html/pcre2syntax.html



```
ismatch(r"^\s*(?:#|$)", "not a comment")
ismatch(r"^\s*(?:#|$)", "# a comment")
ismatch(r"^\s*(?:#|$)", "    ")
```

```
m = match(r"^\s*(?:#|$)", "not a comment")
if m == nothing
    println("not a comment")
else
    println("blank or comment")
end
```

nothing means no value

```
m = match(r"(a|b)(c)?(d)", "yacdx")
m.match                    # “acd”
m.captures                 # [“a”, “c”, “d]
m.offset                   # 2
m.offsets                  # [2, 3, 4]
```

Tuples

Immutable ordered list of values

```
a = (1,2)
b = tuple(3,4,5)
c = (6,"cat",sqrt)
d = 1, 2

b[1]                # 3
first(b)            # 3
last(b)             # 5
length(b)       # 3

b[1] = 9        # MethodError

c[3](100)       # 10
```

Destructuring

```
(x,y) = (4,5)
x                    # 4
y                    # 5

x,y = y,x
x                    # 5
y                    # 4
```