Julia

Tehokas, uusi, moderni, korkean abstraktiotason, matemaattiseen laskentaan suunnattu, geneerinen ohjelmointikieli

Tehokkuus

Luotu korvaamaan hitaat kielet (Python ja R). Tehokkuudeltaan lähellä C-kieltä. Concurrent, parallel ja distributed computing -tuki. JIT -tyylinen käännös suoraan konekielelle.

Korkea abstraktiotaso

Multiple dispatch tärkein paradigma. Proseduraalinen. Funktionaalinen. Metaohjelmointi. Rationaaliluvut. Irrationaaliluvut. Roskienkeruu. Valinnainen (dynaaminen) tyypitys.

Matalan tason rakenteet ja hallinta

Monipuoliset sisäänrakennetut ja itse määriteltävissä olevat value-tyypit. Helppo käyttöjärjestelmäkomentojen kutsu. Prosessien hallinta.

Muuta

Sisäänrakennettu pakettienhallinta. Tehokas unicode-tuki. Sisäänrakennettu C-rajapintakutsu. Helpot Python ja Java -rajapintojen kutsut kirjastoilla.

Pääasiallinen käyttö matemaattisessa laskennassa dokumentteihin upotettuna (Jupyter). Yksi kolmesta Project Jupyter alkuperäiskielistä. Käytössä yli 1500 korkeakoulussa. Vuosittainen JuliaCon conferenssi. Sopii mainiosti myös esimerkiksi scriptikieleksi matalan tason resurssienhallinnan takia. Subjektiivinen huono puoli on viime vuosikymmeninä suosituksi tulleen pakotetun olioparadigman poissaolo. Vielä suhteellisen pieni kirjastovalikoima ja helposti versioiden välillä hajoavat kolmannen osapuolen kirjastot ihan oikeita ongelmia. Oikean staattisen tyypityksen puute rajoittanee käyttöä isoimmissa projekteissa. Veikkaukseni on että Julialla on loistava tulevaisuus ainakin laskennan alalla, ehkä myös geneerisenä ohjelmointikielenä.

Paradigma: Multi-paradigm: multiple dispatch, procedural, functional, meta, multistaged
Kehittäjät: Jeff Bezanson, Alan Edelman, Stefan Karpinski, Viral B. Shah ja muut
Julkaistu: 2012. (1.0 versio Elokuussa 2018)
Vakaa Versio: 1.3.0 (26. Marraskuuta 2019)
Tyypitys: Dynamic, nominative, parametric, optional
Toteutuskieli: Julia, C, C++, Scheme, LLVM
Alustat: x86-64, IA-32, CUDA, ARM, PowerPC
OS: Linux, macOS, Windows ja FreeBSD
Lisenssi: MIT(core), GPL v2(GPL libraries)

Tämä dokumentti kokonaisuudessaan: https://www.cs.helsinki.fi/u/jussiasp/csm14203/

Kielen rakenneosia **3**
Avainsanat 3
Primitiivit 3
Primitiivien tilanvaraus ja arvoalueet 4

Valittuja koodiesimerkkejä **5**
Multiple dispatch 5
Primitiivit ja yksinkertaiset rakenteet 6
Luokkarakennelma 10
Valinnainen tyypitys 11
Poikkeustenhallintaa 12
Käyttöjärjestelmäkomennon kutsuminen 13
C-kirjastojen kutsuminen 13
Javan kutsuminen (periaatteessa) 14
Uuden kirjaston asentaminen ja Pythonin kutsuminen 15

LÄHTEET **16**
Julia 16
Multiple Dispatch 16
Jupyter 16

Kielen rakenneosia

Avainsanat

Avainsanoissa on paljon tuttuja sanoja mm. C-kielestä.

```
abstract type
baremodule
begin
break
catch
const
continue
do
else
elseif
end
export
false
finally
for
function
global
if
import
let
local
macro
module
mutable struct
primitive type
quote
return
struct
true
try
using
while
```

Primitiivit

Valmiit primitiivit, määritelty itse kielessä.

```
primitive type Float16 <: AbstractFloat 16 end
primitive type Float32 <: AbstractFloat 32 end
primitive type Float64 <: AbstractFloat 64 end
primitive type Bool <: Integer 8 end
```

```
primitive type Char <: AbstractChar 32 end
primitive type Int8 <: Signed 8 end
primitive type UInt8 <: Unsigned 8 end
primitive type Int16 <: Signed 16 end
primitive type UInt16 <: Unsigned 16 end
primitive type Int32 <: Signed 32 end
primitive type UInt32 <: Unsigned 32 end
primitive type Int64 <: Signed 64 end
primitive type UInt64 <: Unsigned 64 end
primitive type Int128 <: Signed 128 end
primitive type UInt128 <: Unsigned 128 end
```

Primitiivien tilanvaraus ja arvoalueet

Kokonaislukujen arvoalueet

Type	Number of bits	Smallest value	Largest value
Int8	8	-2^7	-2^7 - 1
UInt8	8	0	-2^8 - 1
Int16	16	-2^15	-2^15 - 1
UInt16	16	0	-2^16 - 1
Int32	32	-2^31	-2^31 - 1
UInt32	32	0	-2^32 - 1
Int64	64	-2^63	-2^63 - 1
UInt64	64	0	-2^64 - 1
Int128	128	-2^127	-2^127 - 1
UInt128	128	0	-2^128 - 1
Bool	8	false(0)	true(1)

Liukulukujen arvoalueet

Type	Precision	Number of bits
Float16	half	16
Float32	single	32
Float64	double	64

Valittuja koodiesimerkkejä

Kaikki esimerkit ovat kokonaisia ja on mahdollista ajaa yksittäin sellaisinaan. Testattu Julian versiolla 1.2.0.

Multiple dispatch

Melko yksinkertainen multiple dispatch esimerkki. Määritellään kolme funktiota, loopataan läpi lista eri tyyppisistä arvoista ja kutsutaan foo-funktiota.

```
foo(bar::Int) = "Sehän on Uuno:$(bar)!"

foo(bar::String) = "Sehän on merkkijono:$(bar)!"

foo(bar::Array) = "Sehän on taulukko:$(bar)!"

foreach([1, "Yksi", [1,1,1,1]]) do i::Any
    println(typeof(i))
    println(foo(i))
end
```

Tulostus

```
Int64
Sehän on Uuno:1!
String
Sehän on merkkijono:Yksi!
Array{Int64,1}
Sehän on taulukko:[1, 1, 1, 1]!
```

Sama koodi Javassa, ei käänny.

```
import java.util.Arrays;

public class Koodi {

    public static void main(String args[]) {
        Koodi koodi = new Koodi();
        koodi.aja();
    }

    public void aja() {
        Object[] lista = {1, "Yksi", new int[]{1,1,1,1} };
        for ( Object i : lista )
        {
            System.out.println(foo(i));
        };
    }

    public String foo(int bar) {
        return "Sehän on Uuno: " + bar + "!";
```

```
    }
    public String foo(String bar) {
        return "Sehän on merkkijono: " + bar + "!";
    }
    public String foo(int[] bar) {
        return "Sehän on taulukko: "+ bar + "!";
    }

}
```

Tulostus

```
Koodi.java:15: error: no suitable method found for foo(Object)
            System.out.println(foo(i));
                               ^
    method Koodi.foo(int) is not applicable
      (argument mismatch; Object cannot be converted to int)
    method Koodi.foo(String) is not applicable
      (argument mismatch; Object cannot be converted to String)
    method Koodi.foo(int[]) is not applicable
      (argument mismatch; Object cannot be converted to int[])
1 error
```

Primitiivit ja yksinkertaiset rakenteet

Julian perusdatatyyppejä ja tyyppimuunnoksia.

```
# tyypin kysyminen
what = function(x)
    println("tyyppi:$(typeof(x))")
end
value_and_what = function(x)
    println("arvo:$x")
    what(x)
end

# luodaan taulukko johon sijoitetaan kaikki arvot ja tulostetaan lopuksi niidet tyypit
kaikki = [

    # tyyppi implisiittisesti

    true,           # Bool
    1,              # Int64
    1.0,            # Float64
    1//3,           # Rational{Int64}
    pi,             # Irrational{:π}
    'x',            # Char
    "Hei maailma",  # String
    1im,            # Complex{Int64}
    4:6,            # UnitRange{Int64}
    [1, 2, 3],      # Array{Int64,1}
    ("kissa", "koira"), # Tuple{String,String}
    (kissa="kissa", koira="koira"), # NamedTuple{(:kissa, :koira),Tuple{String,String}}

    # tyyppi explisiittisesti
```

```
    Bool(true),     # Bool
    Int8(1),        # Int8
    UInt8(1),       # Uint8
    Int16(1),       # Int16
    UInt16(1),      # UInt16
    Int32(1),       # Int32
    UInt32(1),      # UInt32
    Int(1),         # Int64
    Integer(1),     # Int64
    Int64(1),       # Int64
    UInt64(1),      # UInt64
    Int128(1),      # Int128
    UInt128(1),     # UInt128
    BigInt(1),      # BigInt
    Float16(1),     # Float16
    Float32(1),     # Float32
    float(1),       # Float64
    Float64(1),     # Float64
    BigFloat(1),    # BigFloat
    Rational(1,3),  # Rational{Int64}
    Irrational(π),  # Irrational{:π}
    Char('x'),      # Char
    String("Hello World"), # String
    complex(1),     # Complex{Int64}
    Complex(1),     # Complex{Int64}
    Array{Int64}(undef, 1, 2, 3), # Array{Int64,3}
    Tuple{Int64, Int64}((1, 2)), # Tuple{Int64,Int64}
    UnitRange(4,6), # UnitRange{Int64}
    Dict([("name", "John"), ("age",36)]), # Dict{String,Any}
    Dict([("name"=>"John"), ("age"=>36)]), # Dict{String,Any}
    WeakKeyDict([("name", "John"), ("age",36)]), # WeakKeyDict{String,Any}
    Set(), # Set{Any}
    Set{String}(["omena","banaani","kirsikka"]), # Set{String}
    BitSet([1,2,3]), # BitSet
    Array{Int64, 3}(undef, 1, 2, 3), # Array{Int64,3}
    Tuple{Int64,Int64}((1,2)), # Tuple{Int64,Int64}
    NamedTuple{(:kissa, :koira)}((1,2)), # NamedTuple{(:kissa, :koira),Tuple{Int64,Int64}}

    # joitain yhdistelmiä
    true + 2,       # Int64
    true + 1.0,     # Float64
    ℯ,              # Irrational{:ℯ}
    ℯ+1,            # Float64
    π,              # Irrational{:π}
    π+1,            # Float64
    π+ℯ,            # Float64
    1im + true,     # Complex{Int64}
    1:7 + 8,        # UnitRange{Int64}
    [2,3,4] * 3,    # Array{Int64,1}
    [2,3,4] .* 3,   # Array{Int64,1}

]
```

```
println("\nlista tyyppejä:")
foreach(value_and_what, kaikki)

println("\njotain tyyppejä")
what(nothing),        # Nothing
what(missing),        # Missing
what(something),      # typeof(something)

println("\nenums:")

# enums
@enum Väri punainen sininen vihreä
instances(Väri)
value_and_what(Väri) # DataType
value_and_what(punainen) # Väri
```

Tulostus

```
lista tyyppejä:
arvo:true
tyyppi:Bool
arvo:1
tyyppi:Int64
arvo:1.0
tyyppi:Float64
arvo:1//3
tyyppi:Rational{Int64}
arvo:π
tyyppi:Irrational{:π}
arvo:x
tyyppi:Char
arvo:Hei maailma
tyyppi:String
arvo:0 + 1im
tyyppi:Complex{Int64}
arvo:4:6
tyyppi:UnitRange{Int64}
arvo:[1, 2, 3]
tyyppi:Array{Int64,1}
arvo:("kissa", "koira")
tyyppi:Tuple{String,String}
arvo:(kissa = "kissa", koira = "koira")
tyyppi:NamedTuple{(:kissa, :koira),Tuple{String,String}}
arvo:true
tyyppi:Bool
arvo:1
tyyppi:Int8
arvo:1
tyyppi:UInt8
arvo:1
tyyppi:Int16
arvo:1
tyyppi:UInt16
arvo:1
tyyppi:Int32
arvo:1
tyyppi:UInt32
arvo:1
tyyppi:Int64
arvo:1
tyyppi:Int64
arvo:1
tyyppi:Int64
arvo:1
tyyppi:UInt64
```

```
arvo:1
tyyppi:Int128
arvo:1
tyyppi:UInt128
arvo:1
tyyppi:BigInt
arvo:1.0
tyyppi:Float16
arvo:1.0
tyyppi:Float32
arvo:1.0
tyyppi:Float64
arvo:1.0
tyyppi:Float64
arvo:1.0
tyyppi:BigFloat
arvo:1//3
tyyppi:Rational{Int64}
arvo:π
tyyppi:Irrational{:π}
arvo:x
tyyppi:Char
arvo:Hello World
tyyppi:String
arvo:1 + 0im
tyyppi:Complex{Int64}
arvo:1 + 0im
tyyppi:Complex{Int64}
arvo:[0 0]

[0 0]

[0 0]
tyyppi:Array{Int64,3}
arvo:(1, 2)
tyyppi:Tuple{Int64,Int64}
arvo:4:6
tyyppi:UnitRange{Int64}
arvo:Dict{String,Any}("name" => "John","age" => 36)
tyyppi:Dict{String,Any}
arvo:Dict{String,Any}("name" => "John","age" => 36)
tyyppi:Dict{String,Any}
arvo:WeakKeyDict{String,Any}()
tyyppi:WeakKeyDict{String,Any}
arvo:Set(Any[])
tyyppi:Set{Any}
arvo:Set(["kirsikka", "omena", "banaani"])
tyyppi:Set{String}
arvo:BitSet([1, 2, 3])
tyyppi:BitSet
arvo:[0 0]

[0 0]

[0 0]
tyyppi:Array{Int64,3}
arvo:(1, 2)
tyyppi:Tuple{Int64,Int64}
arvo:(kissa = 1, koira = 2)
tyyppi:NamedTuple{(:kissa, :koira),Tuple{Int64,Int64}}
arvo:3
tyyppi:Int64
arvo:2.0
tyyppi:Float64
arvo:ℯ
tyyppi:Irrational{:ℯ}
arvo:3.718281828459045
tyyppi:Float64
arvo:π
tyyppi:Irrational{:π}
```

```
arvo:4.141592653589793
tyyppi:Float64
arvo:5.859874482048838
tyyppi:Float64
arvo:1 + 1im
tyyppi:Complex{Int64}
arvo:1:15
tyyppi:UnitRange{Int64}
arvo:[6, 9, 12]
tyyppi:Array{Int64,1}
arvo:[6, 9, 12]
tyyppi:Array{Int64,1}

jotain tyyppejä
tyyppi:Nothing
tyyppi:Missing
tyyppi:typeof(something)

enums:
arvo:Väri
tyyppi:DataType
arvo:punainen
tyyppi:Väri
```

Luokkarakennelma

Luokan määrittely, instanssin luominen ja luokkaan liittyvien funktioiden kutsuminen.

```
mutable struct Henkilö
    nimi::String
    pituus::Int64
    sijainti::Int64
end
function juokse(henkilö::Henkilö)
    println("$(henkilö.nimi) juoksee")
    henkilö.sijainti = henkilö.sijainti + 2
end
function kävele(henkilö::Henkilö)
    println("$(henkilö.nimi) kävelee")
    henkilö.sijainti = henkilö.sijainti + 1
end
function huuda(henkilö::Henkilö, huuto::String)
    println("$(henkilö.nimi) huutaa $huuto")
end
matti = Henkilö("masa", 7, 8)
println("sijainti: $(matti.sijainti)")
kävele(matti)
println("sijainti: $(matti.sijainti)")
juokse(matti)
println("sijainti: $(matti.sijainti)")
huuda(matti, "jyy")
```

Tulostus

```
sijainti: 8
masa kävelee
sijainti: 9
masa juoksee
sijainti: 11
masa huutaa jyy
```

Valinnainen tyypitys

Valinnaisen tyypityksen esimerkkejä.

```
# valinnainen tyypitys

mutable struct Henkilö
    nimi::String
    pituus::Int64
    sijainti::Int64
end

function juokse(henkilö::Henkilö)
    println("$(henkilö.nimi) juoksee")
    henkilö.sijainti = henkilö.sijainti + 2
end

antti = Henkilö("Antti", 160, 30)

juokse(antti)

# ei tyypitystä

mutable struct Ministeri
    nimi
    pituus
    sijainti
end

function kävele(joku)
    println("$(joku.nimi) kävelee")
    joku.sijainti = joku.sijainti + 1;
end

sanna = Ministeri("Susanna", 150, 1)

kävele(sanna)

kävele(antti)

# koira ei ole henkilö eikä sijaitse missään

mutable struct Koira
    nimi
end

lassie = Koira("Lassie")

# aiheutetaan muutama ajonaikainen virhe

try
    kävele(lassie)
catch e
```

```
    println("herjas siinä: $(e)") #  ErrorException("type Koira has no field sijainti")
end

try
    juokse(lassie)
catch e
    println("herjas toistamiseen: $(e)") #  MethodError(juokse, (Koira("Lassie"),),
0x00000000000065e5)
end

try
    juokse(sanna)
catch x
    println("herjas taas: $(x)") # MethodError(juokse, (Ministeri("Susanna", 150, 2),),
0x00000000000065e5)
end
```

Tulostus

```
Antti juoksee
Susanna kävelee
Antti kävelee
Lassie kävelee
herjas siinä: ErrorException("type Koira has no field sijainti")
herjas toistamiseen: MethodError(juokse, (Koira("Lassie"),), 0x00000000000065e5)
herjas taas: MethodError(juokse, (Ministeri("Susanna", 150, 2),), 0x00000000000065e5)
```

Poikkeustenhallintaa

Poikkeustapaus

```
funktio_poikkeuksella = function(x)
    println("olen alussa.")
    if(x > 100)
        throw(DomainError(x, "argumenttisi ei kelpaa, haluan alle sata!"))
    end
    println("olen lopussa.")
end
try
    funktio_poikkeuksella(200)
catch e
    println("jotain on pielessä: ", e)
    if isa(e, DomainError)
        println("jaa se onkin vain Domainerror")
    else
        println("jotain todella outoa sattui.")
        throw(e)
    end
finally
    println("lopullinen toiminta.")
end
```

Tulostus

```
olen alussa.
```

```
jotain on pielessä: DomainError(200, "argumenttisi ei kelpaa, haluan alle sata!")
jaa se onkin vain Domainerror
lopullinen toiminta.
```

Käyttöjärjestelmäkomennon kutsuminen

Kutsutaan muutamaa konsolikomentoa.

```
run(`echo ajetaan ls:`)
run(`ls`)

run(`echo`)
run(`echo piping to sed:`)
run(pipeline(`echo 'antti rinne'`, `sed 's/rinne/lindtman/g'`))

run(`echo`)
run(`echo pingataan tamperetta ja otetaan vastaus talteen:`)
tampere=read(`ping www.uta.fi -n 1`, String)
println("saatiin tampere:$tampere")
```

Tulostus

```
ajetaan ls:
R_koodi.r     aja_python.sh               julia_koodi.jl    os_kutsu.jl
uusi_kirjasto.jl
aja_R.sh      all_aja.sh                  keywords.txt      primitiivit.jl
aja_julia.sh  dynaaminen_ja_staattinen.jl  luokkarakenne.jl  python_koodi.py

piping to sed:
antti lindtman

pingataan tamperetta ja otetaan vastaus talteen:
saatiin tampere:
Pinging www.uta.fi [153.1.6.41] with 32 bytes of data:
Reply from 153.1.6.41: bytes=32 time=129ms TTL=55

Ping statistics for 153.1.6.41:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 129ms, Maximum = 129ms, Average = 129ms
```

C-kirjastojen kutsuminen

Kutsutaan C:n standardikirjaston funktioita.

```
# kutsutaan standardi c:n clock() -funktiota.
t = ccall(:clock, Int32, ())
println("kello on $t")

# kutsutaan getenv(), haetaan pointteri SHELL-ympäristömuuttujan sijaintiin
path = ccall(:getenv, Cstring, (Cstring,), "SHELL")
# tulostetaan raakana
println("path:$path")
# tulostetaan polku
println("path content:$(unsafe_string(path))")
```

Tulostus

```
kello on 322
path:Cstring(0x0000000002778777)
path content:C:/msys64/usr/bin/bash
```

Javan kutsuminen (periaatteessa)

Yritetään asentaa Java yhteensopivuuskirjasto ja kutsua Javaa. Näin se pitäisi periaatteessa toimia mutta tämä ei nähtävästi tällä hetkellä toimi, ellei asenna vanhempaa Julia-versiota.

```
using Pkg

Pkg.add("JavaCall")

using JavaCall

jlm = @jimport java.lang.Math

x=jcall(jlm, "sin", jdouble, (jdouble,), pi/2)

println(x)

jnu = @jimport java.net.URL

gurl = jnu((String,), "http://www.google.com")

y=jcall(gurl, "getHost", JString,())

println(y)
```

Tulostus

```
  Updating registry at `C:\Users\jussi\.julia\registries\General`
  Updating git-repo `https://github.com/JuliaRegistries/General.git`
 Resolving package versions...
 Installed WinReg ──── v0.3.1
 Installed JavaCall ── v0.7.2
  Updating `C:\Users\jussi\.julia\environments\v1.2\Project.toml`
  [494afd89] + JavaCall v0.7.2
  Updating `C:\Users\jussi\.julia\environments\v1.2\Manifest.toml`
  [494afd89] + JavaCall v0.7.2
  [1b915085] + WinReg v0.3.1
ERROR: LoadError: UndefVarError: penv not defined
Stacktrace:
 [1] jcall(::Type{JavaObject{Symbol("java.lang.Math")}}, ::String, ::Type,
::Tuple{DataType}, ::Float64) at
C:\Users\jussi\.julia\packages\JavaCall\Iz2MQ\src\core.jl:139
 [2] top-level scope at
S:\gdrive\university_and_other_studies\2019_university\csm14203_ohjelmointikielten_periaatt
eet\INFOKOODIT\javacall.jl:9
 [3] include at .\boot.jl:328 [inlined]
 [4] include_relative(::Module, ::String) at .\loading.jl:1094
 [5] include(::Module, ::String) at .\Base.jl:31
 [6] exec_options(::Base.JLOptions) at .\client.jl:295
 [7] _start() at .\client.jl:464
in expression starting at
S:\gdrive\university_and_other_studies\2019_university\csm14203_ohjelmointikielten_periaatt
eet\INFOKOODIT\javacall.jl:9
```

Uuden kirjaston asentaminen ja Pythonin kutsuminen

Asennetaan ja kutsutaan Python yhteensopivuuskirjasto PyCall ja kutsutaan Pythonin math.sin-metodia.

```
using Pkg

Pkg.add("PyCall")

using PyCall

math = pyimport("math")

x = math.sin(math.pi / 4 - sin(pi / 4))

println("x:$x")
```

Tulostus

```
  Updating registry at `C:\Users\jussi\.julia\registries\General`
  Updating git-repo `https://github.com/JuliaRegistries/General.git`
 Resolving package versions...
 Installed CodeTools ──────────────── v0.6.5
 Installed Rmath ──────────────────── v0.6.0
 Installed FFMPEG ─────────────────── v0.2.4
 Installed ArrayInterface ─────── v2.0.0
 Installed Graphics ───────────────── v1.0.0
 Installed QuadGK ─────────────────── v2.3.0
 Installed CategoricalArrays ─ v0.7.4
 Installed Optim ──────────────────── v0.19.7
 Installed StatsFuns ──────────────── v0.9.2
 Installed DiffEqDiffTools ────── v1.5.0
 Installed OffsetArrays ─────────── v0.11.3
 Installed BinDeps ────────────────── v1.0.0
 Installed MacroTools ─────────────── v0.5.3
 Installed JuliaInterpreter ─── v0.7.5
 Installed DoubleFloats ─────────── v1.0.1
 Installed Arpack ─────────────────── v0.3.2
 Installed ImageCore ──────────────── v0.8.6
  Updating `C:\Users\jussi\.julia\environments\v1.2\Project.toml`
 [no changes]
  Updating `C:\Users\jussi\.julia\environments\v1.2\Manifest.toml`
  [7d9fca2a] ↑ Arpack v0.3.1 ⇒ v0.3.2
  [4fba245c] ↑ ArrayInterface v1.2.1 ⇒ v2.0.0
  [9e28174c] ↑ BinDeps v0.8.10 ⇒ v1.0.0
  [324d7699] ↑ CategoricalArrays v0.7.3 ⇒ v0.7.4
  [53a63b46] ↑ CodeTools v0.6.4 ⇒ v0.6.5
  [01453d9d] ↑ DiffEqDiffTools v1.4.0 ⇒ v1.5.0
  [497a8b3b] ↑ DoubleFloats v1.0.0 ⇒ v1.0.1
  [c87230d0] ↑ FFMPEG v0.2.3 ⇒ v0.2.4
  [a2bd30eb] ↑ Graphics v0.4.0 ⇒ v1.0.0
  [a09fc81d] ↑ ImageCore v0.8.5 ⇒ v0.8.6
  [aa1ae85d] ↑ JuliaInterpreter v0.7.4 ⇒ v0.7.5
  [1914dd2f] ↑ MacroTools v0.5.2 ⇒ v0.5.3
  [6fe1bfb0] ↑ OffsetArrays v0.11.1 ⇒ v0.11.3
  [429524aa] ↑ Optim v0.19.4 ⇒ v0.19.7
  [1fd47b50] ↑ QuadGK v2.1.1 ⇒ v2.3.0
  [79098fc4] ↑ Rmath v0.5.1 ⇒ v0.6.0
  [4c63d2b9] ↑ StatsFuns v0.9.0 ⇒ v0.9.2
  Building FFMPEG → `C:\Users\jussi\.julia\packages\FFMPEG\guN1x\deps\build.log`
  Building Rmath ──→ `C:\Users\jussi\.julia\packages\Rmath\BoBag\deps\build.log`
```

```
  Building Arpack → `C:\Users\jussi\.julia\packages\Arpack\zCmTA\deps\build.log`
x:0.0782114250198284
```

LÄHTEET

Julia

Kotisivu
https://julialang.org/

Määrittelydokumentti
https://docs.julialang.org/en/v1.2/
https://raw.githubusercontent.com/JuliaLang/docs.julialang.org/assets/julia-1.2.0.pdf

Wiki
https://en.wikipedia.org/wiki/Julia_(programming_language)

Avainsanat
https://docs.julialang.org/en/v1.2/base/base/#Keywords-1

JuliaCon
https://juliacon.org/2020/

Miksi Julia luotiin
https://julialang.org/blog/2012/02/why-we-created-julia

Julia -logo
https://en.wikipedia.org/wiki/File:Julia_Programming_Language_Logo.svg

Multiple Dispatch

https://en.wikipedia.org/wiki/Multiple_dispatch

Jupyter

https://jupyter.org/
https://en.wikipedia.org/wiki/Project_Jupyter

9.12.2019 Jussi Asp