Programming

optimisation and operations research algorithms with Julia

for Business Tasks

Prof. Dr. Xavier Gandibleux

Université de Nantes
Département Informatique – Faculté des Sciences et Techniques
France

Lesson 2 – May-June 2022

Some basics

The journey of a thousand miles begins with a single step.
Lao Tzu

How to assign a value to a variable (1/2)...

Integer (Int8, Int16, Int32, Int64):

```julia
julia> zipcode = 44000
```

44000 is an integer literal

Float (Float16, Float32, Float64):

```julia
julia> price = 29.95
```

29.95 is a floating point literal

Boolean:

```julia
julia> positive = true
```

How to assign a value to a variable (1/2)...

Integer (Int8, Int16, Int32, Int64):

```
julia> zipcode = 44000
```

44000 is an integer literal

Float (Float16, Float32, Float64):

```
julia> price = 29.95
```

29.95 is a floating point literal

Boolean:

```
julia> positive = true
```

How to assign a value to a variable (1/2)...

Integer (Int8, Int16, Int32, Int64):

```
julia> zipcode = 44000
```

44000 is an integer literal

Float (Float16, Float32, Float64):

```
julia> price = 29.95
```

29.95 is a floating point literal

Boolean:

```
julia> positive = true
```

How to assign a value to a variable (1/2)...

Integer (Int8, Int16, Int32, Int64):

```
julia> zipcode = 44000
```

44000 is an integer literal

Float (Float16, Float32, Float64):

```
julia> price = 29.95
```

29.95 is a floating point literal

Boolean:

```
julia> positive = true
```

How to assign a value to a variable (2/2)...

Character:

```julia
julia> dot = '.'
```

String:

```julia
julia> sentence = "."
```

How to assign a value to a variable (2/2)...

Character:

```
julia> dot = '.'
```

String:

```
julia> sentence = "."
```

Name of variables

Julia manages the unicode characters.

Exemples of valid names for a variable :

- greek letters:

```
\Delta + [TAB]
```

- an emoji:

```
\:smile:  + [TAB]
```

- etc.

(see https://docs.julialang.org/en/v1/manual/unicode-input/)

How to get the type of an expression...

Definition:

```
typeof(...)
```

Example:

```
julia> typeof(zipcode)
```
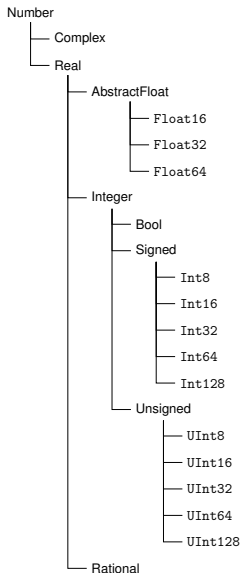
How to get the type of an expression...

Definition:

```
typeof(...)
```

Example:

```
julia> typeof(zipcode)
```

JYU
JOHANNES KEPLER
UNIVERSITY LINZ

How to get the type of an expression...

Definition:

```
typeof(...)
```

Example:

```
julia> typeof(zipcode)
```
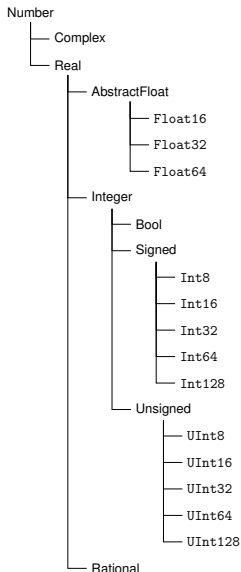
Partial hierarchy of types for numbers

```
Number
  └── Complex
  └── Real
        ├── AbstractFloat
        │     ├── Float16
        │     ├── Float32
        │     └── Float64
        ├── Integer
        │     ├── Bool
        │     ├── Signed
        │     │     ├── Int8
        │     │     ├── Int16
        │     │     ├── Int32
        │     │     ├── Int64
        │     │     └── Int128
        │     └── Unsigned
        │           ├── UInt8
        │           ├── UInt16
        │           ├── UInt32
        │           ├── UInt64
        │           └── UInt128
        └── Rational
```

For a type T,

Function	Definition
supertypef(T)	parent type
subtypes(T)	child type(s)

Partial hierarchy of types for numbers

```
Number
  ├── Complex
  └── Real
        ├── AbstractFloat
        │     ├── Float16
        │     ├── Float32
        │     └── Float64
        ├── Integer
        │     ├── Bool
        │     ├── Signed
        │     │     ├── Int8
        │     │     ├── Int16
        │     │     ├── Int32
        │     │     ├── Int64
        │     │     └── Int128
        │     └── Unsigned
        │           ├── UInt8
        │           ├── UInt16
        │           ├── UInt32
        │           ├── UInt64
        │           └── UInt128
        └── Rational
```

For a type T,

Function	*Definition*
supertypef(T)	parent type
subtypes(T)	child type(s)

Integer and floating point numbers

For a type T,

Function	Definition
sizeof(T)	number of bytes used to represent a value of this type
typemax(T)	largest value for this type
typemin(T)	smallest value for this type

Example:

T	*Signed?*	*sizeof(T)*	*typemin(T)*	*typemax(T)*
Int16	yes	2	$-2^{15} = -32768$	$2^{15} - 1 = 32767$

JYU
JOHANNES KEPLER
UNIVERSITY LINZ

Constants

Definition:

```
const variable = value
```

Example:

```
julia> const g = 9.81
```

Some predefined symbolic constants:

pi or π	value of pi
Inf	positive infinity of type Float64
NaN	not a number

How to print (1/3)...

Display without carriage return:

```
print(...)
```

Display with carriage return:

```
println(...)
```

Examples:

```
julia> print(price)
julia> println()
julia> println(zipcode)
```

```
julia> println("Zip Code:  ", zipcode)
julia> println(zipcode, " | ", price, "€ ", dot)
julia> println("Zip Code:  $zipcode")
```

How to print (1/3)...

Display without carriage return:

```
print(...)
```

Display with carriage return:

```
println(...)
```

Examples:

```
julia> print(price)
julia> println()
julia> println(zipcode)
```

```
julia> println("Zip Code:  ", zipcode)
julia> println(zipcode, " | ", price, "€ ", dot)
julia> println("Zip Code: $zipcode")
```

How to print (1/3)...

Display without carriage return:

```
print(...)
```

Display with carriage return:

```
println(...)
```

Examples:

```julia
julia> print(price)
julia> println()
julia> println(zipcode)
```

```julia
julia> println("Zip Code:  ", zipcode)
julia> println(zipcode, " | ", price, "€ ", dot)
julia> println("Zip Code:  $zipcode")
```

How to print (1/3)...

Display without carriage return:

```
print(...)
```

Display with carriage return:

```
println(...)
```

Examples:

```
julia> print(price)
julia> println()
julia> println(zipcode)
```

```
julia> println("Zip Code:  ", zipcode)
julia> println(zipcode, " | ", price, "€ ", dot)
julia> println("Zip Code:  $zipcode")
```

How to print (2/3)...

Display in c-like style:

```
@printf(...)
```

Example:

```
julia> using Printf

julia> @printf("Zip Code:  %d \n",zipcode)
julia> @printf("π = %.2f \n", pi)
```

How to print (2/3)...

Display in c-like style:

```
@printf(...)
```

Example:

```
julia> using Printf
```

```
julia> @printf("Zip Code:  %d \n",zipcode)
julia> @printf("π = %.2f \n", pi)
```

How to print (2/3)...

Display in c-like style:

```
@printf(...)
```

Example:

```
julia> using Printf
```

```
julia> @printf("Zip Code:  %d \n",zipcode)
julia> @printf("π = %.2f \n", pi)
```

Display an expression and result:

```
@show ...
```

Example:

```
julia> @show zipcode
```

How to print (3/3)...

Display an expression and result:

```
@show ...
```

Example:

```
julia> @show zipcode
```

Read a line of text from from the console (STDIN) until a [NEWLINE]:

```
readline()
```

Example:

```
julia> name = readline()
```

The type of name, obtained with typeof(name), is string.

Get input from users (1/3)...

Read a line of text from from the console (STDIN) until a [NEWLINE]:

```
readline()
```

Example:

```
julia> name = readline()
```

The type of `name`, obtained with `typeof(name)`, is `string`.

Get input from users (2/3)...

Read numerical data types from console:

```
parse()
```

...convert a numeric string (Float or Int) into a numerical value.

Example:

```
julia> num = parse(Int64, readline())
```

The type of `num`, obtained with `typeof(num)`, is `Int64`.

Get input from users (2/3)...

Read numerical data types from console:

```
parse()
```

...convert a numeric string (Float or Int) into a numerical value.

Example:

```
julia> num = parse(Int64, readline())
```

The type of `num`, obtained with `typeof(num)`, is `Int64`.

Get input from users (3/3)...

Read N lines of text input from the console:

```
readlines()
```

N lines are stored as the entries of a one-dimensional String array.
Lines must be delimited with [NEWLINE] or by pressing [ENTER].
[Ctrl-D] to stop taking input.

Example:

```
julia> lines = readlines()
```

See also `TerminalMenus` for simple interactive menus in the terminal
(an example is given in the project).

Get input from users (3/3)...

Read N lines of text input from the console:

```
readlines()
```

> N lines are stored as the entries of a one-dimensional String array.
> Lines must be delimited with [NEWLINE] or by pressing [ENTER].
> [Ctrl-D] to stop taking input.

Example:

```
julia> lines = readlines()
```

See also `TerminalMenus` for simple interactive menus in the terminal
(an example is given in the project).

Get input from users (3/3)...

Read N lines of text input from the console:

```
readlines()
```

> N lines are stored as the entries of a one-dimensional String array.
> Lines must be delimited with [NEWLINE] or by pressing [ENTER].
> [Ctrl-D] to stop taking input.

Example:

```
julia> lines = readlines()
```

See also `TerminalMenus` for simple interactive menus in the terminal
(an example is given in the project).

How to write a comment...

Definition:

```
# ...
```

```
#= ...  =#
```

Example:

```
julia> # this is a line comment
```

```
julia> #= And this
        is a block of comment
        =#
```

How to write a comment...

Definition:

```
# ...
```

```
#= ...   =#
```

Example:

```
julia> # this is a line comment
```

```
julia> #= And this
        is a block of comment
        =#
```

How to write a comment...

Definition:

```
# ...
```

```
#= ...    =#
```

Example:

```
julia> # this is a line comment
```

```
julia> #= And this
        is a block of comment
        =#
```

How to do basic math...

Arithmetic operators

With a and b, two variables:

Operator	Expression	Name
-	-a	unary minus
+	a + b	sum
-	a - b	difference
*	a * b	product
/	a / b	quotient
^	a ^ b	power
÷	a ÷ b	integer divide
%	a % b	modulus

How to do basic math...

Arithmetic operators

With a and b, two variables:

Operator	Expression	Name
–	-a	unary minus
+	a + b	sum
–	a - b	difference
*	a * b	product
/	a / b	quotient
^	a ^ b	power
÷	a ÷ b	integer divide
%	a % b	modulus

How to do basic math...

Updating operators

With a and b, two variables:

Operator	example	equivalent to
+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
^=	a^=b	a=a^b
÷=	a÷=b	a=a÷b
%=	a%=b	a=a%b

JΣU
JOHANNES KEPLER
UNIVERSITY LINZ

Updating operators

With a and b, two variables:

Operator	example	equivalent to
+=	a+=b	a=a+b
-=	a-=b	a=a-b
*=	a*=b	a=a*b
/=	a/=b	a=a/b
^=	a^=b	a=a^b
÷=	a÷=b	a=a÷b
%=	a%=b	a=a%b

How to do basic math...

Numeric comparisons operators

With a and b, two variables:

Operator	Expression	Name
==	a == b	equality
!= or $\neq$	a != b	inequality
<	a < b	less than
<= or $\leqslant$	a <= b	less than or equal to
>	a > b	greater than
>= or $\geqslant$	a >= b	greater than or equal to

A selection of usual functions available (1/4)

Arithmetic functions

With x and y, two variables; n and p, two integer variables:

Function	Description
div(x,y)	truncated division; quotient rounded towards zero
mod(x,y)	modulus
abs(x)	a positive value with the magnitude of *x*
sign(x)	indicates the sign of *x*, returning -1, 0, or +1
sqrt(x)	square root of *x*
exp(x)	e^x
exp2(x)	2^x
log(x)	natural logarithm of *x*
log(b,x)	base *b* logarithm of *x*
factorial(n)	*n*!
binomial(n,p)	the binomial coefficient

A selection of usual functions available (2/4)

Rounding functions

With x, a variable and T, a numeric type:

Function	*Description*	*Return type*
round(x)	round x to the nearest integer	typeof(x)
round(T,x)	round x to the nearest integer	T
round(x,digits=n)	round to n digits after the decimal place	typeof(x)
floor(x)	round x towards -Inf	typeof(x)
floor(T,x)	round x towards -Inf	T
ceil(x)	round x towards +Inf	typeof(x)
ceil(T,x)	round x towards +Inf	T
trunc(x)	round x towards zero	typeof(x)
trunc(T,x)	round x towards zero	T

A selection of usual functions available (3/4)

Trigonometric functions

With `x`, a variable:

Function	Description
`sin(x)`	Compute sine of *x*, where *x* is in radians
`cos(x)`	Compute cosine of *x*, where *x* is in radians
`tan(x)`	Compute tangent of *x*, where *x* is in radians
`sind(x)`	Compute sine of *x*, where *x* is in degrees
`cosd(x)`	Compute cosine of *x*, where *x* is in degrees
`tand(x)`	Compute tangent of *x*, where *x* is in degrees
`deg2rad(x)`	Convert *x* from degrees to radians
`rad2deg(x)`	Convert *x* from radians to degrees

A selection of usual functions available (4/4)

functions on strings

With c, a character, s, s2, strings, :

Function	*Description*
reverse(s)	Reverse a string
occursin(c,s)	Determine if c is a substring of s
split(s,c)	Split s into an array of substrings on occurrences of c
endswith(s,s2)	Return true if s ends with s2
uppercase(s)	Return s with all characters converted to uppercase
textwidth(s)	Give the number of columns needed to print s
isdigit(c)	Test if c is a decimal digit (0-9)
isletter(c)	Test if c is a letter

At this stage, julia is a super calculator

Review and exercises

(notebook)