Лабораторная работа №1.

Julia. Установка и настройка. Основные принципы.

Тазаева А. А.

Российский университет дружбы народов, Москва, Россия

Цели работы

Цели работы

Подготовить рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомиться с основами синтаксиса Julia.

Задание

Задание

1. Установить под свою операционную систему Julia, Jupyter.
2. Используя Jupyter Lab, повторите простейшие примеры с синтаксисом Julia.
3. Выполните задания для самостоятельной работы.

Установка необходимого программного обеспечения

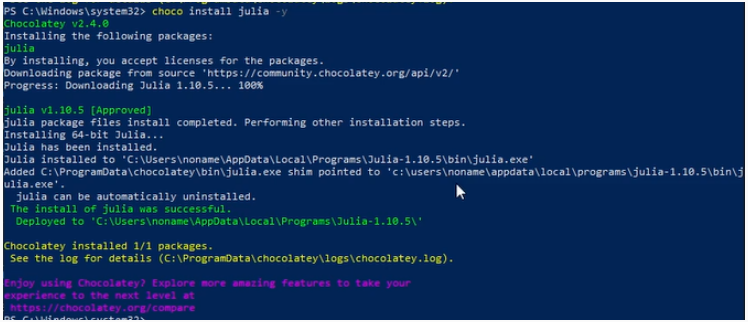

Рис. 1: Установка Julia

Установка необходимого программного обеспечения

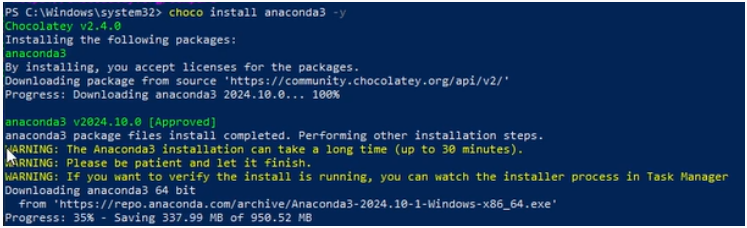

Рис. 2: Установка Anaconda3

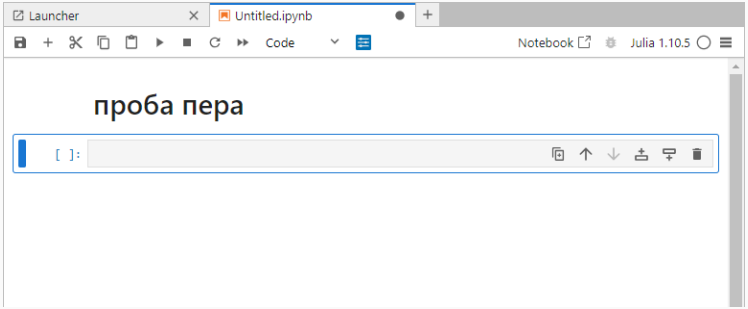

Рис. 3: Создание файла

Рис. 4: Простейшие операции на языке Julia в Jupyter

Лабораторная работа №1. Примеры из раздела 1.3.3.

[20]:
```
typeof(3), typeof(3.5), typeof(3/3.5), typeof(sqrt(3+4im)), typeof(pi)
```

[20]:
```
(Int64, Float64, Float64, ComplexF64, Irrational{:π})
```

[23]:
```
1.0/0.0, 1.0/(-0.0), 0.0/0.0
```

[23]:
```
(Inf, -Inf, NaN)
```

[24]:
```
typeof(1.0/0.0), typeof(1.0/(-0.0)), typeof(0.0/0.0)
```

[24]:
```
(Float64, Float64, Float64)
```

[26]:
```
for T in [Int8,Int16,Int32,Int64,Int128,UInt8,UInt16,UInt32,UInt64,UInt128]
println("$(lpad(T,7)): [$(typemin(T)),$(typemax(T))]")
end
```

```
   Int8: [-128,127]
  Int16: [-32768,32767]
  Int32: [-2147483648,2147483647]
  Int64: [-9223372036854775808,9223372036854775807]
 Int128: [-170141183460469231731687303715884105728,170141183460469231731687303715884105727]
  UInt8: [0,255]
 UInt16: [0,65535]
 UInt32: [0,4294967295]
 UInt64: [0,18446744073709551615]
UInt128: [0,340282366920938463463374607431768211455]
```

Повторение примеров из раздела 1.3.3

```
[27]: Int64(2.0), Char(2), typeof(Char(2))
[27]: (2, '\x02', Char)
[30]: convert(Int64, 2.0), convert(Char,2)
[30]: (2, '\x02')
[31]: 2.0 |> Int64
[31]: 2
[32]: Bool(1), Bool(0)
[32]: (true, false)
[35]: typeof(promote(Int8(1), Float16(4.5), Float32(4.1)))
[35]: Tuple{Float32, Float32, Float32}
```

Рис. 6: Примеры приведения аргументов к одному типу

Самостоятельная работа. Задание 1

```
[53]: ?read()
```

```
[53]: read(io::IO, T)
```

Read a single value of type `T` from `io`, in canonical binary representation.

Note that Julia does not convert the endianness for you. Use `ntoh` or `ltoh` for this purpose.

`read(io::IO, String)`
Read the entirety of `io`, as a `String` (see also `readchomp`).

Examples

```
julia> io = IOBuffer("JuliaLang is a GitHub organization");

julia> read(io, Char)
'J': ASCII/Unicode U+004A (category Lu: Letter, uppercase)

julia> io = IOBuffer("JuliaLang is a GitHub organization");

julia> read(io, String)
"JuliaLang is a GitHub organization"
```

`read(filename::AbstractString)`
Read the entire contents of a file as a `Vector{UInt8}`.

`read(filename::AbstractString, String)`
Read the entire contents of a file as a string.

`read(filename::AbstractString, args...)`
Open a file and read its contents. `args` is passed to `read`: this is equivalent to `open(io-`

```
[57]: task1 = IOBuffer("It is the first task today")
      read(task1, String)

[57]: "It is the first task today"
```

Рис. 8: Функция read(). Пример

Самостоятельная работа. Задание 2

[84]:
```
?parse()
```

[84]:
```
parse(type, str; base)
```

Parse a string as a number. For `Integer` types, a base can be specified (the default is 10). For floating-point types, the string is parsed as a decimal floating-point number. `Complex` types are parsed from decimal strings of the form `"R±Iim"` as a `Complex(R,I)` of the requested type; `"i"` or `"j"` can also be used instead of `"im"`, and `"R"` or `"Iim"` are also permitted. If the string does not contain a valid number, an error is raised.

!!! compat "Julia 1.1" `parse(Bool, str)` requires at least Julia 1.1.

Examples

```
julia> parse(Int, "1234")
1234

julia> parse(Int, "1234", base = 5)
194

julia> parse(Int, "afc", base = 16)
2812

julia> parse(Float64, "1.2e-3")
0.0012

julia> parse(Complex{Float64}, "3.2e-1 + 4.5im")
0.32 + 4.5im
```

```
parse(::Type{Platform}, triplet::AbstractString)
```

Parses a string platform triplet back into a `Platform` object.

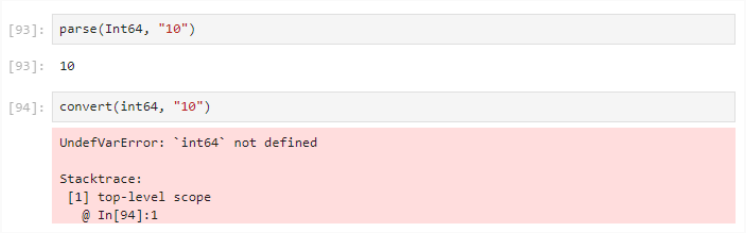

Рис. 10: Функция parse(). Пример

Самостоятельная работа. Задание 3

```
*[97]: plus = 1.0 + 2
[97]: 3.0
[98]: minus = 1 - 2.0
[98]: -1.0
[99]: proizvedenie = 1 * 2.0
[99]: 2.0
[100]: delenie = 1 / 2.0
[100]: 0.5
[102]: delenie_po_modulu = 10 % 3
[102]: 1
[103]: delenie_nacelo = 10 ÷ 3
[103]: 3
[110]: div(10, 3) # delenie nacelo
[110]: 3
[105]: sqrtt = √9
[105]: 3.0
[107]: sqrt(9) # izvlechenie kornya
[107]: 3.0
[108]: pow = 3^2
[108]: 9
[109]: drobi = 10 // 2
[109]: 5//1
```

Самостоятельная работа. Задание 4

```
*[97]: plus = 1.0 + 2
[97]: 3.0
[98]: minus = 1 - 2.0
[98]: -1.0
[99]: proizvedenie = 1 * 2.0
[99]: 2.0
[100]: delenie = 1 / 2.0
[100]: 0.5
[102]: delenie_po_modulu = 10 % 3
[102]: 1
[103]: delenie_nacelo = 10 // 3
[103]: 3
[110]: div(10, 3) # delenie nacelo
[110]: 3
[105]: sqrtt = √9
[105]: 3.0
[107]: sqrt(9) # izvlechenie kornya
[107]: 3.0
[108]: pow = 3^2
[108]: 9
[109]: drobi = 10 // 2
[109]: 5//1
```

Выводы по проделанной работе

Выводы по проделанной работе

В ходе лабораторной работы мной было подготовлено рабочее пространство и инструментарий для работы с языком программирования Julia, также я познакомилась с основами синтаксиса Julia на простейших примерах.