

# Julia 入門

# Julia とは

なぜ僕らはJuliaを作ったか

# インストール

## 直接

- <https://julialang.org/> からダウンロード、インストール
- インストール場所はどこでもOK、パスを通そう
- "julia" で REPL (対話環境) 開始 <span style="color: gray">Read-Eval-Print Loop の略らしい</span>

## Jupyter でも使いたい人 (参考: [Jupyter NotebookeでJuliaを使ってみた](#))

- [conda コマンドからjuliaをインストールする方法](#)もあるらしい (インストールしてしまえば上と同じ状況のはず)
- "]" でパッケージモードに移って "add IJulia"
- Backspace で julia モードに戻って "using IJulia; notebook()" で起動

# REPL の使い方

- "exit()" か Ctrl+D で終了
- "]" でパッケージモード
- "?" でヘルプモード
- BackSpace で Julia モードに戻る
- パッケージの追加：
  - (パッケージモード) add パッケージ名
  - (Julia モード) using Pkg <span style="color:gray">(import Pkg でもいい)</span></span>をした上で Pkg.add("パッケージ名")
- インストールしたパッケージの確認 : Pkg.installed()
- パッケージのアップデート : Pkg.update()

# REPL as 電卓

- 算術、論理、比較、ビット演算子は一般的なものとほぼ同じ
- 累乗は " $^$ "、整数除算は " $\div$ " (`\div+Tab`)、XOR は " $\square$ " (`\xor+Tab`)
- $1/7$  は分数になる
- " $\neq$ " " $\leq$ " " $\geq$ " も用意されている
- 定数として " $\pi$ " " $\square$ " が用意されている (`\pi`, `\euler`) (cf. `Base.Mathconstants`)
- "`\alpha`" "`\Alpha`" などと入力して `Tab` を押すと全角のギリシャ文字 ( $\alpha$ ,  $A$  など) に置換される ( $\pi$  以外は変数用) (TeX 記号は結構対応している)
- <span style="color:gray">余談：LaTeX では `\Alpha` コマンドは用意されておらずアルファベットの  $A$  で代用せざるを得ないが Unicode では区別される。  
cf. [Unicode一覧 0000-0FFF](#)</span>
- ヘルプモードで記号を入力すると TeX での打ち方、演算子の場合は用例も分かる

# データ型

- `typeof(1)` などとしてデータ型を確認できる
- `1.0`, `1//7`, `π`, `2.0im`, `true`, `'a'`, `'□'`, `"ABC"` の型を確認してみよう
- 型変換関数はデータ型と同じ名前。 `Float64(pi * 2)`, `BigFloat(□)`

println() でコンソール出力

## 文字列・配列

- String は Char の配列
- アクセス：最初の要素は [1] または [begin], 最後の要素は [end]
- 配列のスライス："hello"[2:4] とすると "ell" が切り出せる
- 文字列の結合：string("Java","script") とするか "インド" \* "ネシア" とする  
string を使う方法なら文字列以外にも文字列として結合できる
- 文字列の置換：replace("Word to vec", " to " => 2)
- 配列の長さ：length("four")

# 関数

```
function f(x,y)
    x * y # 最後の値が戻り値。return で明示するのも可
end
```

インデントはなくても動く。型も指定できる。

```
function cat(x::String, y::String) :: String
    x * y
end

function cat(x::Int64, y::Int64) :: String
    string(x) * string(y)
end
```

引数の型が違えば違う関数。



# 可變長引數

```
function add(x...)
    sum = 0
    for i = 1:length(x)
        sum += x[i]
    end
    sum
end

println(add(1, 2, 3, 4, 5))
```

# 辞書 (連想配列)

```
戦いの年号集 = Dict{"関ヶ原" => 1600, "桶狭間" => 1560, "小牧・長久手" => 1584)
# 戦いの年号集 = Dict{String, Int32}("関ヶ原" => 1600,
#   "桶狭間" => 1560, "小牧・長久手" => 1584) 型を明示

if haskey(戦いの年号集, "関ヶ原")
    println(get(戦いの年号集, "関ヶ原", 0))
end
```

リスト内法表記のような書き方も可能。

```
Dict{i => i ^ 3 for i = 1:10}
```

順番はばらばら。

**+α**

- $[i^3 \text{ for } i=1:10]$

# グラフをプロット

- "]" でパッケージモードに移って "add Plots" で Plots パッケージをインストール  
(数分かかる)

## 参考文献

- Julia言語プログラミング入門
- REPL (julia コマンド) の使い方