

Julia (v1.0.3) Cheatsheet

Basic Operations

```
typeof(x)  
typeassert(x, type)  
varinfo()  
?  
;
```

Package Installation

```
using Pkg  
Pkg.init()  
Pkg.update()  
Pkg.add("SomePackage")  
Pkg.rm("SomePackage")  
Pkg.status()  
]
```

Data Structures

```
d = Dict("alpha"=>1, "beta"=>2)  
d["beta"]  
get(d, "gamma", 10)  
haskey(d, "gamma")  
  
t = (1, 2, 3)  
length(t)  
  
s = Set([1; 2; 3; 2; 1])  
push!(s, 69)  
in(2, s)
```

```
keys(d)  
values(d)  
delete!(d, "alpha")  
  
t[1:2]  
e, d = d, e  
  
intersect(s, s1)  
union(s, s1)  
setdiff(s, s1)
```

String

```
s1 = "hello"; s2 = " world!"  
s1 * s2 # "hello world!"  
s1 ^ 3 # "hellohellohello"  
s3 = "$(s1)$s2"  
reverse(s3)  
rpad(s3, 100)  
lpad(s2, 100, "fuck ")  
first(s3, 5)  
last(s3, 1)  
  
using Random  
randstring("ATCG", 200)
```

```
cmp(s1, s2)  
findfirst(r, s3)  
occursin(s1, s3)  
replace(s3, "hello" => "nihao")  
split(s3, "o_")  
strip(s3)  
startswith(s3, 'h')  
endswith(s3, "J!")
```

Control Flow

```
# ternary expression  
mod(5, 2) == 0 ? "AT" : "CG"  
  
# if-else  
if rand(1:10) < 5  
    println("Good.")  
elseif rand(1:100) > 45  
    println("Well.")  
else  
    println("What?")  
end  
  
# while loop  
b = 100; i = 1;  
while b > 10  
    b -= i  
    i += 1  
end  
  
# for loop  
s = 0  
a = [1:2:20...]  
for (i, v) in enumerate(a)  
    s += v  
    if s > 80  
        break  
    end  
    println("$i, $s")  
end  
  
# try-catch  
try  
    error("whatever")  
catch e  
    println("ERROR! $e $e")  
end
```

Fundamental Mathematics

```
pi # 3.1415926...  
MathConstants.e # 2.7182818...  
MathConstants.eulergamma # 0.5772156...  
MathConstants.golden # 1.6180339...  
Inf # ∞  
-Inf # -∞  
NaN # not a number  
  
2//3 # rational number  
2//3 + 4//5 # = 22/15  
  
2.0 + 3.0im # complex number
```

Arithmetic operators

a + b # $a + b$	-a # $-a$
a - b # $a - b$	a ^ b # a^b
a * b # $a \times b$	a \ b # $\frac{1}{a} \times b$
a / b # $a \div b$	a % b # $a - \lfloor a/b \rfloor \times b$

Comparison operators

a == b # $a = b$	isequal(a, b) # $a = b$
a != b # $a \neq b$	isfinite(a) # $ a < \infty$
a > b # $a > b$	isinf(a) # $a = \pm\infty$
a ≥ b # $a \geq b$	isnan(a) # a is not a number
a < b # $a < b$	iszero(a) # $a = 0$
a ≤ b # $a \leq b$	isone(a) # $a = 1$
a < b > c # $a < b$ and $b > c$	isinteger(a) # $a = [a]$

Complex numbers

complex(a, b) # $a + ib$	abs(z) # $\ z\ $
real(z)	abs2(z) # $\ z\ ^2$
imag(z)	conj(z) # \bar{z}
angle(z)	cis(z) # e^{iz}
rad2deg(angle(z))	sign(z) # $z/\ z\ $

Number theory

div(x, y) # x/y rounded → 0	divrem(x, y) # $(\text{div}(x, y), \text{rem}(x, y))$
fld(x, y) # $[x/y]$	fldmod(x, y) # $(\text{fld}(x, y), \text{mod}(x, y))$
cld(x, y) # $\lceil x/y \rceil$	gcd(x, y) # GCD
rem(x, y) # $x \% y$ sign same as x	lcm(x, y) # LCM
mod(x, y) # $x \% y$ sign same as y	mod2pi(x) # $x \% 2\pi$

Power and logarithm

sqrt(x) # \sqrt{x}	log(x) # $\ln(x)$
cbrt(x) # $\sqrt[3]{x}$	log1p(x) # $\ln(1+x)$
hypot(x, y) # $\sqrt{x^2 + y^2}$	log(b, x) # $\log_b(x)$
exp(x)	log2(x) # $\log_2(x)$
expm1(x) # $e^x - 1$	log10(x) # $\log_{10}(x)$

Trigonometric and hyperbolic functions

sin(x)	asin(x)	sind(x)	asind(x)	sinh(x)	asinh(x)
csc(x)	acsc(x)	cscd(x)	acscd(x)	csch(x)	acsch(x)
cos(x)	acos(x)	cosd(x)	acosd(x)	cosh(x)	acosh(x)
sec(x)	asec(x)	secd(x)	asecd(x)	sech(x)	asech(x)
tan(x)	atan(x)	tand(x)	atand(x)	tanh(x)	atanh(x)
cot(x)	acot(x)	cotd(x)	acotd(x)	coth(x)	acoth(x)

Functions

```
# define a function  
function fname(a, b = 2; c = 10, d = -3)  
    result = (a + b) * (c + d)  
    return result  
end  
  
# define a function in one line  
fname1(x, y) = a * b  
  
# Anonymous Functions  
(x -> mod(x, 12)).(1:24...)
```

Linear Algebra

Some functions require `using LinearAlgebra`.
Basic information

```
a = [1 2 3 4]  
b = [1; 2; 3; 4]  
A = [1 2 3; 4 5 6; 7 8 9]  
size(a) # (1, 4)  
size(b) # (4, )  
ndims(a) # 2  
ndims(b) # 1  
length(a) # 4
```

```
# indexing  
a[1] # 1  
b[end] # 4  
a[1:3] # [1; 2; 3] !!!  
a[:] # [1; 2; 3; 4] !!!  
b[2:end-1] # [2; 3]  
A[:,2:3] # [2 3; 5 6; 8 9]  
A[:] # [1:9...]
```

Construction

```
collect(1:4) # [1; 2; 3; 4]  
collect(1:2:8) # [1; 3; 5; 7]  
collect(0:1:3.5) # [0.1; 1.2; 2.3; 3.4]  
collect(0.1:0.2:0.8) # [0.1; 0.3; 0.5; 0.7]  
collect(1:-2:-5) # [1; -1; -3; -5]  
[1:4...] # [1; 2; 3; 4]  
[1:2:8...] # [1; 3; 5; 7]  
[x^2 for x in a] # [1; 4; 9; 16]  
[x^y for x=2:3, y=1:3] # [2 4 8; 3 9 27]
```

```
zeros([T=Float64], dims...)  
ones([T=Float64], dims...)  
trues(dims...)  
falses(dims...)  
fill(x, dims...)
```

Modification

```
push!(a, n)  
pushfirst!(a, n)  
append!(a, b)
```

```
pop!(a, n)  
popfirst!(a, n)  
splice!(a, i)
```

Sort

```
sort(a)  
p = sortperm(a); a[p]
```

```
sort(a, by=abs, rev=true)  
sort(A, dims=1)
```

Search

```
n in a  
in(n, a)  
findmax(A)  
findmin(A)  
findmax(A, dims=1)  
findmin(A, dims=2)  
findlast(f::Function, A)  
findnext(f::Function, A, i)  
findprev(f::Function, A, i)
```

```
# return indices !!!  
findall(f::Function, A)  
findfirst(f::Function, A)  
findmin(A, dims=1)  
findmax(A, dims=2)
```

Matrix properties

```
det(A)  
rank(A)  
tr(A)  
inv(A)  
pinv(A)
```

```
diag(M, k::Integer=0)  
diagm(1 => [1, 2, 3], -1 => [4, 5])  
tril(A)  
triu(A)  
norm(A)
```

Matrix, vector operations

```
a' * b # dot product  
dot(x, y) # dot product  
cross(x, y) # cross product  
a .* b  
A * a  
A \ b
```

```
A' # transpose  
dot(A, B)  
A / B  
A \ B  
A^n  
exp(A)
```

Special matrix

```
issymmetric(A)  
isposdef(A)
```

```
ishermitian(A)
```

Decomposition

```
F = qr(A) # QR factorization  
F.Q  
F.R
```

```
E = eigen(A)  
E.values  
E.vectors
```