datacamp

julia

Julia Basics Cheat Sheet

Learn Julia online at www.DataCamp.com

> Accessing help

```
# Access help mode with an empty ?
?

# Get help on a function with ?functionname
?first

# Search for help on a topic with ?topic
?function
```

> Comments

```
# This is a single-line comment
```

```
#= This is a
multi-line comment =#
```

> Information about objects

```
# Get the type of an object with typeof() – Example returns Int64
typeof(20)
```

> Using packages

Packages are libraries of pre-written code (by other programmers) that we can add to our Julia installation, which help us solve specific problems. Here's how to install and work with packages in Julia.

```
# Enter package mode with ] to install and work with packages
]

# Install a new package with add
add CSV

# Exit package mode with DELETE
<DEL>

# Load a package with using
using CSV

# Load a package with import without an alias
import CSV

# Load a package with import with an alias
import DataFrames as df
```

> The working directory

The working directory is a file path that Julia will use as the starting point for relative file paths. That is, it's the default location for importing and exporting files. An example of a working directory looks like "/Users/myname/workspace/myproject"

```
# Get current working director with pwd()
pwd()
"/home/programming_languages/julia"

# Set the current directory with cd()
cd("/home/programming_languages/julia/cheatsheets")
```

> Operators

Arithmetic operators

```
# Add two numbers with +
37 + 102

# Subtract two numbers with -
102 - 37

# Multiply two numbers with *
4 * 6

# Divide a number by another with /
21/7

# Integer divide a number with ÷
22 ÷ 7 # This returns 3

# Inverse divide a number with \
5 \ 0 # This is equivalent to 0/5

# Raise to the power using ^
3 ^ 3

# Get the remainder after division with %
22 % 7
```

Assignment operators

```
# Assign a value to an object with =
a = 5

# Add two objects; store in left-hand object with +=
a += 3 # This is the same as a = a + 3

# Subtract an object from another; store in left-hand object with -=
a -= 3 # This is the same as a = a - 3
```

Numeric comparison operators

```
# Test for equality with ==
3 == 3 # This returns true

# Test for not-equality with ≠
3 ≠ 3 # This returns false

# Test greater than with >
3 > 1

# Test greater or equal than with ≥
3 ≥ 3

# Test less than with <
3 < 4

# Test less or equal than with ≤
3 ≤ 4
```

Other operators

```
# Determine if a value is in an array with x in arr
x = [11, 13, 19]
13 in x # This returns true

# Pipe values to a function with value ▷ fn
x ▷ (y → length(y) + sum(y)) # This returns 43
```

Logical operators

```
# Logical not with ~
~(2 == 2) # Returns false

# Elementwise and with &
(1 ≠ 1) & (1 < 1) # Returns false

# Elementwise or with |
(1 ≥ 1) | (1 < 1) # Returns true

# Elementwise xor (exclusive or) with ⊻
(1 ≠ 1)⊻(1 < 1) # Returns false
```

> Vectors

Vectors are one-dimensional arrays in Julia. They allow a collection of items such as floats, integers, strings or a mix that allows duplicate values.

Creating vectors

```
# Create vectors with square brackets, [x1, x2, x3]
x = [1, 2, 3]

# Create vectors, specifying element types using Vector{type}()
Vector{Float64}([1, 2, 3])

# Create sequence of numbers from a to b with a:b
37:100

# Create sequence of numbers from a to b in steps with a:step:b
1:2:101

# Create vector that repeats m times and each element repeats n times
repeat(vector, inner=n, outer=m)
```

Vector functions

```
# Sorting vectors with sort(x)
x = [9, 1, 4]
sort(x)

# Reversing vectors with reverse(x)
reverse(x)

# Reversing in-place with reverse!(x)
reverse!(x)

# Get vector's unique elements with unique()
unique(x)
```

Selecting vector elements

```
# Selecting the 6th element of a vector with x[6]
x = [9, 1, 4, 6, 7, 11, 5]
x[6]

# Selecting the first element of a vector with x[begin]
x[begin] # This is the same as x[1]

# Selecting the last element of a vector with x[end]
x[end] # This is the same as x[7]

# Slicing elements two to six from a vector with x[2:6]
x[2:6]

# Selecting the 2nd and 6th element of a vector with x[[2, 6]]
x[[2,6]]

# Selecting elements equal to 5 with x[x .== 5]
x[x .== 5]

# Selecting elements less than 5 with x[x .< 5]
x[x .< 5]

# Selecting elements in the vector 2, 5, 8 with x[in([2, 5, 8]).(x)]
x[in([2, 5, 8]).(x)]
```

datacamp

julia

Julia Basics Cheat Sheet

Learn Julia online at www.DataCamp.com

> Math functions

```
# Example vector
x = [9, 1, 4, 6, 7, 11, 5]

# Get the logarithm of a number with log()
log(2)

# Get the element-wise logarithm of a vector with log.()
log.(x)

# Get the exponential of a number with exp()
exp(2)

# Get the element-wise exponential of a vector with exp.()
exp.(x)

# Get the maximum of a vector with maximum()
maximum(x)

# Get the minimum of a vector with minimum()
minimum(x)

# Get the sum of a vector with sum()
sum(x)
```

The following code requires installing and loading the `Statistics` and `StatsBase` packages. This can be done with the command below

```
] # Enter package mode
add Statistics # Add the Statistics package
add StatsBase # Add the StatsBase package
using Statistics # Load the package with using
using StatsBase # Load the package with using

# Get the mean of a vector with mean()
mean(x)

# Get the median of a vector with median()
median(x)

# Get quantiles of a vector with quantile(x, p)
quantile(x, [0.25, 0.75])

# Round values of a vector with round.(x, digits = n)
round.(x, 2)

# Get the ranking of vector elements with StatsBase.ordinalrank()
ordinalrank(x)

# Get the variance of a vector with var()
var(x)

# Get the standard deviation of a vector with std()
std(x)

# Get the correlation between two vectors with cor(x, y)
y = [1, 4, 2, 10, 23, 16, 5]
cor(x, y)
```

> Getting started with characters and strings

Characters and strings are text data types in Julia. Characters refer to text data with exactly one character, and are created with single quotes, `''`. Strings are sequences of characters, and are created with double or triple-double quotes, `" "` or `""" """`.

```
# Create a character variable with single quotes
char = 'a'

# Create a string variable with double quotes
string = "Hello World!"

# Create a string variable with triple double quotes
string = """Hello
            World!"""

# Extract a single character from a string
string = "Hello World!"

string[1] # This extracts the first character
string[begin] # This extracts the first character
string[end] # This extracts the last character

# Extract a string from a string
string[1:3] # Extract first three characters as a string
string[begin:4] # Extract first four characters as a string
string[end-2: end] # Extract last three characters as a string
```

Combining and splitting strings

```
# Combine strings with *
"Listen" * " to " * "DataFramed!" # This returns "Listen to DataFramed!"

# Repeat strings with ^
"Echo! " ^ 3 # Returns "Echo! Echo! Echo! "

# Interpolate strings with "$value"
language = "Julia"
"I'm learning $language" # Returns "I'm learning Julia"

# Split strings on a delimiter with split()
split("lions and tigers and bears", " and ") # Returns 3-element vector
```

Finding and mutating strings

```
# Detect the presence of a pattern in a string with occursin()
occursin("Julia", "Julia for data science is cool") # This returns true

# Find the position of the first match in a string with findfirst()
findfirst("Julia", "Julia for data science is cool") # This returns 1:5

# Convert a string to upper case with uppercase()
uppercase("Julia") # Returns "JULIA"

# Convert a string to lower case with lowercase()
lowercase("Julia") # Returns "julia"

# Convert a string to title case case with titlecase()
titlecase("Julia programming") # Returns "Julia Programming"

# Replace matches of a pattern with a new string with replace()
replace("Learn Python on DataCamp.", "Python" ⇒ "Julia")
```

> Getting started with DataFrames

```
# Install the DataFrames and CSV packages
]
add DataFrames
add CSV
using DataFrames
using CSV

# Create a DataFrame with DataFrame()
df = DataFrame(
    numeric_column = 1:4, # Vector of integers
    string_column= ['M', 'F', 'F', 'M'], # Vector of characters
    a_number = 0, # Fill whole column with one integer
    a_string = "data frames" # Fill whole column with one string
)

# Select a row from a data frame with [ and column number
df[3, :] # Return the third row and all columns

# Select a column from a DataFrame using . and column name
df.string_column

# Select a column from a DataFrame using [ and column number
df[:, 2] # Return the second column and all rows

# Select an element from a DataFrame using [ and row and column numbers
df[1, 2] # Return the first row of the second column
```

> Manipulating data frames

```
# Concatenate two data frames horizontally with hcat()
df1 = DataFrame(column_A = 1:3, column_B = 1:3)
df2 = DataFrame(column_C = 4:6, column_D = 4:6)

df3 = hcat(df1, df2) # Returns 4-column DataFrame with columns A, B, C, D

# Filter for rows of a df3 with filter() where column_A > 2
df_filter = filter(row → row.column_A > 2, df3)

# Select columns of a data frame with select()
select(df3, 2) # Return the second column

# Drop columns of a data frame with select(Not())
select(df3, Not(2)) # Return everything except second column

# Rename columns of a data frame with rename(old → new)
rename(df3, ["column_A" → "first_column"])

# Get rows of a df3 with distinct values in column_A with unique(df, :col)
unique(df3, :column_A)

# Order the rows of a data frame with sort()
sort(df3, :numeric_column)

# Get data frame summary statistics with describe()
describe(df3)
```