Nushell Cheat Sheet

by lomm28 via cheatography.com/191865/cs/39872/

Data Types

`> " -5" | into int`	converts string to integer
`> date now | date to-tim ezone " Eur ope /Lo ndo n"`	converts present date to provided time zone

`> {'name': 'nu', 'stars': 5, 'langu age': 'Python'}`
`  | upsert language 'Rust'`
	updates a record's language and if none is specified inserts provided value

`> [one two three] | to yaml`	converts list of strings to yaml

`> [{name: 'Robert' age: 34 position: 'Desig ner'}`
`{name: 'Margaret' age: 30 position: 'Software Develo per'}`
 	
`{name: 'Natalie' age: 50 position: 'Accou nta nt'}]`
`| select name position`
	selects two columns from the table and prints their values

Lists

`> [foo bar baz] | insert 1 'beeze'`
`> $strin g_list`
	inserts **`beeze`** value at st index in the list

`> [1, 2, 3, 4] | update 1 10`	updates 2nd value to 10

`> let numbers = [1, 2, 3, 4, 5]`
`> $numbers | prepend 0`
	adds value at the beginning of the list

`> let numbers = [1, 2, 3, 4, 5]`
`> $numbers | append 6`
	adds value at the end of the list

`> let flowers = [cammomile marigold rose forget -me -not]`
`> let flowers = ($flowers | first 2)`
`> $flowers`
	creates slice of the first two values from **`flowers`** list

`> let planets = [Mars Jupiter Saturn Uranus Neptune]`
`> $planets | each { |it| $"($it) is a planet of solar system " }`
	iterates over a list; **`it`** is current list value

`> $planets | enumerate | each {`
`  |it| $"($ it.i ndex + 1) - ($it.i tem )"`
`}`
	iterates over a list and provides index and value in **`it`**

`> let scores = [3 8 4]`
`> $"total = ($scores | reduce { |it, acc| $acc + $it })"`
	reduces the list to a single value, **`reduce`** gives access to accumulator that is applied to each element in the list

`> $"total = ($scores | reduce --fold 1 { |it, acc| $acc * $it })"`	initial value for accamulator value can be set with **`--fold`**

`> let planets = [Mars Jupiter Saturn Uranus Neptune]`
`> $planets.2`
	gives access to the 3rd item in the list

Lists (cont)

`> let planets = [Mars Jupiter Saturn Uranus Neptune]`
`> $planets | any {|it| $it | str starts -with " E" }`
	checks if any string in the list starts with `E`

`> let cond = {|x| $x < 0 }; [-1 -2 9 1] | take while $cond`	creates slice of items that satisfy provided condition

Tables

`> ls | sort-by size`	sorting table by size of files
`> ls | sort-by size | first 5`	sorting table by size of files and show first 5 entries

`> let $a = [`
`   [a_col b_col c_col]; [foo bar snooze]`
` ]`
`> let $b = [`
`   [a_col b_col c_col]; [hex seeze feeze]`
` ]`
`> $a | append $b`
	concatenates two tables with same columns

`> let teams_ scores = [`
`   [team score plays]; ['team_1' 311 3] ['team_2', 245 2]`
` ]`
`> $teams _scores | drop column`
	removes the last column of a table

Files & Filesystem

`> start file.txt`	opens a text file with the default text editor
`> 'lorem ipsum ' | save file.txt`	saves a string to text file
`> 'dolor sit amet' | save --append file.txt`	appends a string to the end of file.txt
`> { a: 1, b: 2 } | save file.json`	saves a record to file.json
`> glob **/*.{ rs, toml} --depth 2`	searches for `.rs` and `.toml` files recursively up to 2 folders deep
`> watch . --glob =** /*.rs {|| cargo test }`	runs cargo test whenever a Rust file changes

Custom Commands

`def greet [name: string] {`
`   $"hello ($name )"`
`}`
	custom command with parameter type set to `string`

`def greet [name = " nus hel l"] {`
`   $"hello ($name )"`
`}`
	custom command with default parameter set to `nushell`

Custom Commands (cont)

`def greet [`
`    name: string`
`    --age: int`
`] {`
`    [$name $age]`
`}`
`> greet world --age 10`
	passing named parameter by defining flag for custom commands

`def greet [`
`    name: string`
`    --age (-a): int`
`    --twice`
`] {`
`    if $twice {`
`        [$name $age $name $age]`
`    } else {`
`        [$name $age]`
`    }`
`}`
`> greet -a 10 --twice hello`
	using flag as a switch with a shorthand flag (`-a`) for the age

`def greet [...name: string] {`
`    print " hello all:"`
`    for $n in $name {`
`        print $n`
`    }`
`}`
`> greet earth mars jupiter venus`
	custom command which takes any number of positional arguments using rest params

Variables and Subexpressions

`> let val = 42`
`> print $val`
`42`
	an immutable variable cannot change its value after declaration

`> let val = 42`
`> do { let val = 101;  $val }`
`101`
`> $val`
`42`
	shadowing variable (declaring variable with the same name in a different scope)

`> mut val = 42`
`> $val += 27`
`> $val`
`69`
	declaring a mutable variable with `mut` key word

`> mut x = 0`
`> [1 2 3] | each { $x += 1 }`
	closures and nested defs cannot capture mutable variables from their environment. This expression results in error.

Variables and Subexpressions (cont)

`> const plugin = 'path/ to/ plugin'`
`> register $plugin`
	a constant variable is immutable value which is fully evaluated at parse-time

`> let files = (ls)`
`> $files.na me?.0?`
	using question mark operator to return `null` instead of error if provided path is incorrect

`> let big_files = (ls | where size > 10kb)`
`>  $big_files`
	using subexpression by wrapping the expression with parentheses `()`

Modules

```
> module greetings {
   export def hello [name: string] {
       $"hello ($name )!"
   }
   export def hi [where: string] {
       $"hi ($wher e)! "
   }
}
> use greetings hello
> hello " wor ld"
```

using inline module

```
# greeti ngs.nu
export-env {
   $env.M YNAME = " Arthur, King of the Briton s"
}
export def hello [] {
   $"hello ($env.M YN AME )"
}
> use greeti ngs.nu
> $env.M YNAME
Arthur, King of the Britons
> greetings hello
hello Arthur, King of the Britons!
```

importing module from file and using its environment in current scope

Modules (cont)

```
# greeti ngs.nu
export def hello [name: string] {
    $"hello ($name )!"
}
export def hi [where: string] {
    $"hi ($wher e)! "
}
export def main [] {
   " gre etings and salutations!"
}
> use greeti ngs.nu
> greetings
greetings and saluta tions!
> greetings hello world
hello world!
```

using `main` command in module

Strings

`> let name = " Ali ce"`
`> $"gr eet ings, ($name )!"`
	prints `greetings, Alice!`

`> let string _list = " one ,tw o,t hre e" | split row " ,"`
`> $strin g_list`
	splits the string with specified delimiter and saves the list to `string _list` variable

`> " Hello, world! " | str contains "o, w"`
`> $strin g_list`
	checks if a string contains a substring and returns `boo lean`

`> let str_list = [zero one two]`
`> $str_list str join ','`
	joins the list of strings using provided delimiter

`> 'Hello World!' | str substring 4..8`	created a slice from a given string with start (4) and end (8) indices
`> 'Nushell 0.80' | parse '{shell} {version}'`	parses a string into a table
`> " acr ony m,l ong \nAPL,A Progra mming Langua ge" | from csv`	parses comma separated values (csv)
`> $'(ansi purple _bo ld)This text is a bold purple !(ansi rese t)'`	ansi command colors the text (alsways end with `ansi reset` to reset color to default)