Package: gradecode (via r-universe)

May 7, 2026

Title Grade Students' Code

Version 0.1.3

Description Provides functions for inspeacting and providing feedback for code input by students with 'gradethis'.

License MIT + file LICENSE

URL https://github.com/rstudio/gradecode

BugReports https://github.com/rstudio/gradecode/issues

Imports and, cli (>= 3.4.1), glue, gradethis (>= 0.2.8.9000), magrittr, purrr, R6, rlang, tibble, xml2, xmlparsedata

Suggests dplyr, fs, roxygen2, rstudioapi, testthat (>= 3.0.0), usethis, utils, withr

Remotes rstudio/gradethis

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Config/pak/sysreqs cmake make libuv1-dev libxml2-dev zlib1g-dev

Repository https://rstudio.r-universe.dev

Date/Publication 2025-11-10 15:29:03 UTC

RemoteUrl https://github.com/rstudio/gradecode

RemoteRef HEAD

RemoteSha 157a0985b6c1301c1a01b45ef31ce09e365c4a6a

Contents

`fail_if_not_found` *Return a failing grade based on the result of a find function*

Description

Return a failing grade based on the result of a find function

Usage

```
fail_if_not_found(gradecode_found, message = NULL, ..., env = parent.frame())

fail_if_found(gradecode_found, message = NULL, ..., env = parent.frame())
```

Arguments

`gradecode_found`	The result of a `find_*()` function
`message`	A character string of the message to be displayed. In all grading helper functions other than `graded()`, `message` is a template string that will be processed with `glue::glue()`.
`...`	Arguments passed on to `gradethis::fail` `hint` Include a code feedback hint with the failing message? This argument only applies to `fail()` and `fail_if_equal()` and the message is added using the default options of `give_code_feedback()` and `maybe_code_feedback()`. The default value of `hint` can be set using `gradethis_setup()` or the `gradethis.fail.hint` option. `encourage` Include a random encouraging phrase with `random_encouragement()`? The default value of `encourage` can be set using `gradethis_setup()` or the `gradethis.fail.encourage` option.
`env`	environment to evaluate the glue `message`. Most users of **gradethis** will not need to use this argument.

Value

- `fail_if_not_found()` returns a failing `gradethis` grade if the last `find_*()` function did not find a result. Otherwise, it returns the `gradecode_found` object unaltered.
- `fail_if_found()` returns a failing `gradethis` grade if the last `find_*()` function did not find a result. Otherwise, it returns the `gradecode_found` object unaltered.

Examples

```
library(dplyr)
.user_code <- "mtcars %>% mutate(mpg = mean(mpg))"
.solution_code <- "mtcars %>% group_by(cyl) %>% summarize(mpg = mean(mpg))"

find_functions(.user_code) %>%
  fail_if_not_found()

group_by_message <- "Remember to use `group_by()` to summarize each group individually."
find_functions(.user_code, match = group_by) %>%
  fail_if_not_found(message = group_by_message)
find_functions(.solution_code, match = group_by) %>%
  fail_if_not_found(message = group_by_message)

mutate_message <- "`mutate()` returns a value for each observation, not each group."
find_functions(.user_code, match = mutate) %>%
  fail_if_found(message = mutate_message)
find_functions(.solution_code, match = mutate) %>%
  fail_if_found(message = mutate_message)
```

---

find_argument_values *Find the values of arguments in R code*

Description

Find the values of arguments in R code

Usage

```
find_argument_values(
  code = .user_code,
  match = NULL,
  recurse = NULL,
  env = parent.frame()
)

uses_argument_value(
  code = .user_code,
  match = NULL,
  recurse = NULL,
```

```
  env = parent.frame()
)
```

Arguments

code	A character string containing the code to search within. Defaults to `.user_code`.
match	What argument value to find in code. Defaults to `NULL`, which searches for any argument.
recurse	If `TRUE`, find arguments to all functions. If `FALSE`, find only arguments of the current top-level node. If `NULL` (the default), intelligently select whether to use recursive searching or not: • Use recursive searching unless called immediately after `find_functions()` or `find_arguments()`. • If called after `find_functions()` or `find_arguments()`, only search for arguments of the current top-level node.
env	Environment in which to find `.user_code`. Most users will not need to use this argument.

Value

`find_argument_values()` returns a `gradecode_found` object, which can be passed to another `find_*()` function or to `fail_if_not_found()`.

`uses_argument_value()` returns a `TRUE`/`FALSE` value.

Examples

```
.user_code <- "mean(rnorm(10, mean = 1, sd = 0.1), na.rm = TRUE)"

find_argument_values(.user_code)

.user_code %>% find_functions(mean) %>% find_argument_values()
.user_code %>% find_functions(rnorm) %>% find_argument_values()

# If `match` is specified, `find_argument_values()` only finds that argument
find_argument_values(.user_code, match = TRUE)
# If `match` does not exist in the code, `find_argument_values()` returns nothing
find_argument_values(.user_code, match = FALSE)

# `uses_argument_value()` returns `TRUE` if `match` appears in the code
uses_argument_value(.user_code, TRUE)
uses_argument_value(.user_code, FALSE)
.user_code %>% find_functions(mean) %>% uses_argument_value(10)
.user_code %>% find_functions(rnorm) %>% uses_argument_value(10)
```

find_arguments *Find arguments in R code*

Description

Find arguments in R code

Usage

```
find_arguments(
  code = .user_code,
  match = NULL,
  recurse = NULL,
  env = parent.frame()
)

uses_argument(
  code = .user_code,
  match = arg(),
  recurse = NULL,
  env = parent.frame()
)

arg(...)
```

Arguments

code	A character string containing the code to search within. Defaults to `.user_code`.
match	What argument to find in `code`. Either `NULL`, a character vector, or a call to `arg()`. If `match` is a character vector, find all arguments with names matching the character vector. An empty string `""` will match unnamed arguments. If `match` is a call to `arg()`, find arguments with both names and values matching the arguments to `arg()` (see examples). Defaults to `NULL`, which searches for any argument.
recurse	If `TRUE`, find arguments to all functions. If `FALSE`, find only arguments of the current top-level node. If `NULL` (the default), intelligently select whether to use recursive searching or not: • Use recursive searching unless called immediately after `find_functions()` or `find_arguments()`. • If called after `find_functions()` or `find_arguments()`, only search for arguments of the current top-level node.
env	Environment in which to find `.user_code`. Most users will not need to use this argument.
...	Any argument passed to `...` in `arg()` will be used to match arguments in `code`.

Value

`find_arguments()` returns a `gradecode_found` object, which can be passed to another `find_*()` function or to `fail_if_not_found()`.

`uses_argument()` returns a `TRUE`/`FALSE` value.

Examples

```
.user_code <- "mean(rnorm(10, mean = 1, sd = 0.1), na.rm = TRUE)"

find_arguments(.user_code)

.user_code %>% find_functions(mean) %>% find_arguments()
.user_code %>% find_functions(rnorm) %>% find_arguments()

# If `match` is specified, `find_arguments()` only finds that argument
# Use a character string to find an argument by name (regardless of value)
find_arguments(.user_code, match = "na.rm")

# Use `arg()` to find an argument by name and value
find_arguments(.user_code, match = arg(na.rm = TRUE))
find_arguments(.user_code, match = arg(na.rm = FALSE))

# If `match` is a character vector of length > 1, find any of those arguments
find_arguments(.user_code, match = c("mean", "sd"))

# If `match` is an empty string, find unnamed arguments
find_arguments(.user_code, match = "")

# Chained calls to `find_arguments()` can be useful for targeted feedback
.user_code %>%
  find_arguments("na.rm") %>%
  fail_if_not_found("Make sure to specify an `na.rm` argument.") %>%
  find_arguments(arg(na.rm = TRUE)) %>%
  fail_if_not_found("Make sure to set the `na.rm` argument to `TRUE`.")

# `uses_argument()` returns `TRUE` if `match` appears in the code
.user_code %>% find_functions(mean) %>% uses_argument("na.rm")
.user_code %>% find_functions(mean) %>% uses_argument(arg(na.rm = TRUE))
.user_code %>% find_functions(mean) %>% uses_argument("sd")
.user_code %>% find_functions(mean) %>% uses_argument(arg(sd = 0.1))
```

---

find_arithmetic *Find arithmetic operators in R code*

Description

Find arithmetic operators in R code

Usage

```
find_arithmetic(code = .user_code, match = NULL, env = parent.frame())

uses_arithmetic(code = .user_code, match = NULL, env = parent.frame())
```

Arguments

code	A character string containing the code to search within. Defaults to .user_code.
match	Which arithmetic operator to find in code. Either NULL or a character vector containing arithmetic operators ("+", "-", "*", "/", "^", "%%", "%/%", "%*%", "%o%" or "%x%"). If match is a character vector of length greater than 1, find_arithmetic() will find *any* of the specified operators. Defaults to NULL, which searches for any arithmetic operator.
env	Environment in which to find .user_code. Most users will not need to use this argument.

Value

find_arithmetic() returns a gradecode_found object, which can be passed to another find_*() function or to fail_if_not_found().

uses_arithmetic() returns a TRUE/FALSE value.

See Also

Other operator functions: find_assigns(), find_comparisons(), find_extractions(), find_infixes(), find_logical_operators(), find_operators(), find_pipes()

Examples

```
find_arithmetic("-1 + 2 * 3 %/% 4")

# `match` lets you look for a specific operator
find_arithmetic("-1 + 2 * 3 %/% 4", match = "+")
find_arithmetic("-1 + 2 * 3 %/% 4", match = "-")
find_arithmetic("-1 + 2 * 3 %/% 4", match = "%/%")

# If `match` is a vector, find multiple operators
find_arithmetic("-1 + 2 * 3 %/% 4", match = c("+", "-"))
# It's okay if not all elements of a vector `match` are present
find_arithmetic("-1 + 2 * 3 %/% 4", match = c("*", "/"))

# `uses_arithmetic()` returns a TRUE or FALSE value
uses_arithmetic("-1 + 2 * 3 %/% 4", match = "+")
uses_arithmetic("-1 + 2 * 3 %/% 4", match = "-")
uses_arithmetic("-1 + 2 * 3 %/% 4", match = "^")
```

find_assigns *Find assignments in R code*

Description

Find assignments in R code

Usage

```
find_assigns(code = .user_code, match = NULL, env = parent.frame())

uses_assign(code = .user_code, match = NULL, env = parent.frame())
```

Arguments

code	A character string containing the code to search within. Defaults to `.user_code`.
match	Which assignment operator to find in code. Either `NULL` or a character vector containing assignment operators (`"<-"`, `"->"`, `"="`, `"<<-"`, `"->>"` or `":="`). If `match` is a character vector of length greater than 1, `find_assigns()` will find *any* of the specified operators. Defaults to `NULL`, which searches for any assignment operator.
env	Environment in which to find `.user_code`. Most users will not need to use this argument.

Value

`find_assigns()` returns a `gradecode_found` object, which can be passed to another `find_*()` function or to `fail_if_not_found()`.

`uses_assign()` returns a `TRUE`/`FALSE` value.

See Also

Other operator functions: `find_arithmetic()`, `find_comparisons()`, `find_extractions()`, `find_infixes()`, `find_logical_operators()`, `find_operators()`, `find_pipes()`

Examples

```
find_assigns("x <- 1; y = 2; 3 -> z")

# `match` lets you look for a specific operator
find_assigns("x <- 1; y = 2; 3 -> z", match = "<-")
find_assigns("x <- 1; y = 2; 3 -> z", match = "=")
find_assigns("x <- 1; y = 2; 3 -> z", match = "->")

# If `match` is a vector, find multiple operators
find_assigns("x <- 1; y = 2; 3 -> z", match = c("<-", "->"))
```

```
# It's okay if not all elements of a vector `match` are present
find_assigns("x <- 1; y = 2; 3 -> z", match = c("=", ":="))

# `uses_assign()` returns a TRUE or FALSE value
uses_assign("3 -> z", match = "<-")
uses_assign("3 -> z", match = "=")
uses_assign("3 -> z", match = "->")
```

`find_comparisons` *Find comparison operators in R code*

Description

Find comparison operators in R code

Usage

```
find_comparisons(code = .user_code, match = NULL, env = parent.frame())

uses_comparison(code = .user_code, match = NULL, env = parent.frame())
```

Arguments

`code`	A character string containing the code to search within. Defaults to `.user_code`.
`match`	Which comparison operator to find in `code`. Either `NULL` or a character vector containing comparison operators (`"<"`, `">"`, `"<="`, `">="`, `"=="` or `"!="`). If `match` is a character vector of length greater than 1, `find_comparisons()` will find *any* of the specified operators. Defaults to `NULL`, which searches for any comparison operator.
`env`	Environment in which to find `.user_code`. Most users will not need to use this argument.

Value

`find_comparisons()` returns a `gradecode_found` object, which can be passed to another `find_*()` function or to `fail_if_not_found()`.

`uses_comparison()` returns a `TRUE`/`FALSE` value.

See Also

Other operator functions: `find_arithmetic()`, `find_assigns()`, `find_extractions()`, `find_infixes()`, `find_logical_operators()`, `find_operators()`, `find_pipes()`

Examples

```
find_comparisons("x < 1; x >= -1; x != 0")

# `match` lets you look for a specific operator
find_comparisons("x < 1; x >= -1; x != 0", match = "<")
find_comparisons("x < 1; x >= -1; x != 0", match = ">=")
find_comparisons("x < 1; x >= -1; x != 0", match = "!=")

# If `match` is a vector, find multiple operators
find_comparisons("x < 1; x >= -1; x != 0", match = c("<", ">="))
# It's okay if not all elements of a vector `match` are present
find_comparisons("x < 1; x >= -1; x != 0", match = c("<", "<=", ">", ">="))

# `uses_comparison()` returns a TRUE or FALSE value
uses_comparison("x < 1; x >= -1; x != 0", match = "<")
uses_comparison("x < 1; x >= -1; x != 0", match = "!=")
uses_comparison("x < 1; x >= -1; x != 0", match = ">")
```

`find_extractions` *Find extraction operators in R code*

Description

Find extraction operators in R code

Usage

```
find_extractions(code = .user_code, match = NULL, env = parent.frame())

uses_extraction(code = .user_code, match = NULL, env = parent.frame())
```

Arguments

code	A character string containing the code to search within. Defaults to `.user_code`.
match	Which extraction operator to find in code. Either `NULL` or a character vector containing extraction operators (`"["`, `"[["`, `"$"` or `"@"`). If `match` is a character vector of length greater than 1, `find_extractions()` will find *any* of the specified operators. Defaults to `NULL`, which searches for any extraction operator.
env	Environment in which to find `.user_code`. Most users will not need to use this argument.

Value

`find_extractions()` returns a `gradecode_found` object, which can be passed to another `find_*()` function or to `fail_if_not_found()`.

`uses_extraction()` returns a `TRUE`/`FALSE` value.

See Also

Other operator functions: find_arithmetic(), find_assigns(), find_comparisons(), find_infixes(), find_logical_operators(), find_operators(), find_pipes()

Examples

```
find_extractions("mtcars[mtcars$cyl == 4, 'mpg'][[1]]")

# `match` lets you look for a specific operator
find_extractions("mtcars[mtcars$cyl == 4, 'mpg'][[1]]", match = "[")
find_extractions("mtcars[mtcars$cyl == 4, 'mpg'][[1]]", match = "$")
find_extractions("mtcars[mtcars$cyl == 4, 'mpg'][[1]]", match = "[[")

# If `match` is a vector, find multiple operators
find_extractions("mtcars[mtcars$cyl == 4, 'mpg'][[1]]", match = c("[", "[["))
# It's okay if not all elements of a vector `match` are present
find_extractions("mtcars[mtcars$cyl == 4, 'mpg'][[1]]", match = c("$", "@"))

# `uses_extraction()` returns a TRUE or FALSE value
uses_extraction("mtcars[mtcars$cyl == 4, 'mpg'][[1]]", match = "[")
uses_extraction("mtcars[mtcars$cyl == 4, 'mpg'][[1]]", match = "[[")
uses_extraction("mtcars[mtcars$cyl == 4, 'mpg'][[1]]", match = "@")
```

---

find_functions *Find functions in R code*

Description

Find functions in R code

Usage

```
find_functions(
  code = .user_code,
  match = NULL,
  recurse = TRUE,
  env = parent.frame()
)

uses_function(code = .user_code, match = NULL, env = parent.frame())
```

Arguments

code	A character string containing the code to search within. Defaults to .user_code.

match	Which function to find in code. Either NULL, a list of unquoted function names (e.g. mean or c(mean, median)), or a character vector of function names (e.g. "mean" or c("mean", "median")). If match has length greater than 1, find_functions() will find *any* of the specified functions. Defaults to NULL, which searches for any function. Unquoted function names search by value, e.g. summarize and summarise will match because they are two names for the same function. Character strings search by name, e.g. "summarize" and "summarise" will not match because they have different names.
recurse	If TRUE, find functions at any level, including functions nested inside the arguments to another function. If FALSE, find only top-level functions (i.e. functions that are not nested inside the arguments to another function). find_functions(recurse = FALSE) will only find nested functions if it is used after another find_*() function that found a function's arguments (e.g. find_arguments(). Defaults to TRUE.
env	Environment in which to find .user_code. Most users will not need to use this argument.

Value

find_functions() returns a gradecode_found object, which can be passed to another find_*() function or to fail_if_not_found().

uses_function() returns a TRUE/FALSE value.

Examples

```
library(dplyr)
.user_code <- "mtcars %>% group_by(cyl) %>% summarize(mpg = mean(mpg))"

find_functions(.user_code)

# If `match` is specified, `find_functions()` only finds that function
find_functions(.user_code, match = group_by)
# If `match` does not exist in the code, `find_functions()` returns nothing
find_functions(.user_code, match = mutate)

# `match = summarise` can find the function `summarize()`,
# because they are two names for the same function
find_functions(.user_code, match = summarise)
# but `match = "summarise"` looks for an exact name,
# so it can't find the function `summarize()`
find_functions(.user_code, match = "summarise")

# If `match` has length > 1, `find_functions()` finds any of those operators
find_functions(.user_code, match = c(group_by, summarize))
# There is no issue if only some of the `match` functions exist in the code
find_functions(.user_code, match = c(mean, median))
```

```
# `uses_function()` returns `TRUE` if `match` appears in the code
uses_function(.user_code, match = group_by)
uses_function(.user_code, match = mean)
uses_function(.user_code, match = median)

uses_function(.user_code, match = summarise)
uses_function(.user_code, match = "summarise")
```

`find_infixes` *Find infixes in R code*

Description

Infixes are R functions that are surrounded by %s and are placed between two arguments, like `%in%`, `%%`, or `%>%`.

Usage

```
find_infixes(code, match = NULL, env = parent.frame())

uses_infix(code = .user_code, match = NULL, env = parent.frame())
```

Arguments

`code`	A character string containing the code to search within. Defaults to `.user_code`.
`match`	What infix to find in code. Either `NULL`, an unquoted infix operator (e.g. `` `%in%` ``), or a character string containing an infix (e.g. `"%in%"`). Defaults to `NULL`, which searches for any infix. Unquoted infixes search by value, while character strings search by name.
`env`	Environment in which to find `.user_code`. Most users will not need to use this argument.

Value

`find_infixes()` returns a `gradecode_found` object, which can be passed to another `find_*()` function or to `fail_if_not_found()`.

`uses_infix()` returns a `TRUE`/`FALSE` value.

See Also

Other operator functions: `find_arithmetic()`, `find_assigns()`, `find_comparisons()`, `find_extractions()`, `find_logical_operators()`, `find_operators()`, `find_pipes()`

Examples

```
find_infixes("'mpg' %in% names(mtcars)")

# `match` lets you look for a specific operator
find_infixes("'mpg' %in% names(mtcars)", match = `%in%`)
find_infixes("'mpg' %in% names(mtcars)", match = `%*%`)

# `uses_pipe()` returns a TRUE or FALSE value
uses_infix("'mpg' %in% names(mtcars)", match = `%in%`)
uses_infix("'mpg' %in% names(mtcars)", match = `%*%`)
```

---

find_lhs *Find the left- or right-hand side of an operator in R code*

Description

- `find_lhs()` and `uses_lhs()` look for the left-hand side of an operator.
- `find_rhs()` and `uses_rhs()` look for the right-hand side of an operator.

Usage

```
find_lhs(code, env = parent.frame())

uses_lhs(code = .user_code, env = parent.frame())

find_rhs(code, env = parent.frame())

uses_rhs(code = .user_code, env = parent.frame())
```

Arguments

code	A character string containing the code to search within. Defaults to `.user_code`.
env	Environment in which to find `.user_code`. Most users will not need to use this argument.

Value

`find_lhs()` and `find_rhs()` return a `gradecode_found` object, which can be passed to another `find_*()` function or to `fail_if_not_found()`.

`uses_lhs()` and `uses_rhs()` return a `TRUE`/`FALSE` value.

Examples

```
"rnorm(10) %>% mean(na.rm = TRUE)" %>% find_pipes() %>% find_lhs()
"rnorm(10) %>% mean(na.rm = TRUE)" %>% find_pipes() %>% find_rhs()

"2 ^ 5" %>% find_operators() %>% find_lhs()
"2 ^ 5" %>% find_operators() %>% find_rhs()

"!1" %>% find_operators() %>% uses_lhs()
"!1" %>% find_operators() %>% uses_rhs()
```

---

find_logical_operators

Find logical operators in R code

Description

Find logical operators in R code

Usage

```
find_logical_operators(code = .user_code, match = NULL, env = parent.frame())

uses_logical_operator(code = .user_code, match = NULL, env = parent.frame())
```

Arguments

code	A character string containing the code to search within. Defaults to .user_code.
match	Which logical operator to find in code. Either NULL or a character vector containing logical operators ("&", "&&", "|", "||" or "!"). If match is a character vector of length greater than 1, find_logical_operators() will find *any* of the specified operators. Defaults to NULL, which searches for any logical operator.
env	Environment in which to find .user_code. Most users will not need to use this argument.

Value

find_logical_operators() returns a gradecode_found object, which can be passed to another find_*() function or to fail_if_not_found().

uses_logical_operator() returns a TRUE/FALSE value.

See Also

Other operator functions: find_arithmetic(), find_assigns(), find_comparisons(), find_extractions(), find_infixes(), find_operators(), find_pipes()

Examples

```
find_logical_operators("FALSE | !FALSE && TRUE")

# `match` lets you look for a specific operator
find_logical_operators("FALSE | !FALSE && TRUE", match = "|")
find_logical_operators("FALSE | !FALSE && TRUE", match = "!")
find_logical_operators("FALSE | !FALSE && TRUE", match = "&&")

# If `match` is a vector, find multiple operators
find_logical_operators("FALSE | !FALSE && TRUE", match = c("|", "!"))
# It's okay if not all elements of a vector `match` are present
find_logical_operators("FALSE | !FALSE && TRUE", match = c("&", "&&"))

# `uses_logical_operator()` returns a TRUE or FALSE value
uses_logical_operator("FALSE | !FALSE && TRUE", match = "|")
uses_logical_operator("FALSE | !FALSE && TRUE", match = "!")
uses_logical_operator("FALSE | !FALSE && TRUE", match = "&")
```

---

find_objects *Find objects in R code*

Description

Find objects in R code

Usage

```
find_objects(code = .user_code, match = NULL, env = parent.frame())

uses_object(code = .user_code, match = NULL, env = parent.frame())
```

Arguments

code	A character string containing the code to search within. Defaults to `.user_code`.
match	Which object to find in code. Either `NULL`, an unquoted object name (e.g. `mtcars` or `dplyr::storms`), or a quoted object name (e.g. `"mtcars"` or `"dplyr::storms"`). Defaults to `NULL`, which searches for any object. Unquoted object names search by value, e.g. `storms` and dplyr::storms will match because they are two ways to refer to the same object. Character strings search by name, e.g. `"storms"` and `"dplyr::storms"` will not match because they are not the same character string.
env	Environment in which to find `.user_code`. Most users will not need to use this argument.

Value

`find_objects()` returns a `gradecode_found` object, which can be passed to another `find_*()` function or to `fail_if_not_found()`.

`uses_object()` returns a `TRUE`/`FALSE` value.

Examples

```
library(dplyr)
.user_code <- "band_members %>% full_join(band_instruments)"

find_objects(.user_code)

# If `match` is specified, `find_objects()` only finds that object
find_objects(.user_code, match = band_members)
# If `match` does not exist in the code, `find_objects()` returns nothing
find_objects(.user_code, match = storms)

# `match = dplyr::band_members` can find the object `band_members`,
# because they are two ways of referring to the same object
find_objects(.user_code, match = dplyr::band_members)
# but `match = "dplyr::band_members"` looks for an exact name,
# so it can't find the object `band_members`
find_objects(.user_code, match = "dplyr::band_members")

# `uses_function()` returns `TRUE` if `match` appears in the code
uses_object(.user_code, match = band_members)
uses_object(.user_code, match = band_instruments)
uses_object(.user_code, match = storms)

uses_object(.user_code, match = dplyr::band_members)
uses_object(.user_code, match = "dplyr::band_members")
```

---

find_operators *Find operators in R code*

Description

Operators are R functions that are placed between two arguments, like `+`, `&&`, `>=` or `%in%`, or that are placed directly in front of an argument, like `-`, `!` or `?`.

Usage

```
find_operators(code = .user_code, match = NULL, env = parent.frame())

uses_operator(code = .user_code, match = NULL, env = parent.frame())
```

Arguments

code	A character string containing the code to search within. Defaults to `.user_code`.
match	Which operator to find in code. Either `NULL` or a character vector containing operators (e.g. `"+"`, `"-"`, `"!"`, `"&&"`, `">="`, `"?"` or `"%in%"`). If `match` is a character vector of length greater than 1, `find_operators()` will find *any* of the specified operators. Defaults to `NULL`, which searches for any operator.
env	Environment in which to find `.user_code`. Most users will not need to use this argument.

Value

`find_operators()` returns a `gradecode_found` object, which can be passed to another `find_*()` function or to `fail_if_not_found()`.

`uses_operator()` returns a `TRUE`/`FALSE` value.

See Also

Other operator functions: `find_arithmetic()`, `find_assigns()`, `find_comparisons()`, `find_extractions()`, `find_infixes()`, `find_logical_operators()`, `find_pipes()`

Examples

```
# By default, `find_operators()` finds all operators in the code
find_operators("-1 + 2 * 3 %/% 4")
find_operators("x <- 1; y = 2; 3 -> z")
find_operators("x < 1; x >= -1; x != 0")
find_operators("mtcars[mtcars$cyl == 4, 'mpg'][[1]]")
find_operators("FALSE | !FALSE && TRUE")

.user_code <- "3 ^ 4 > 4 ^ 3 && -1 + -1 < -1 * -1"
find_operators(.user_code)

# If `match` is specified, `find_operators()` only finds that operator
find_operators(.user_code, match = "-")
# If `match` does not exist in the code, `find_operators()` returns nothing
find_operators(.user_code, match = "/")

# If `match` has length > 1, `find_operators()` finds any of those operators
find_operators(.user_code, match = c("<", ">"))
# There is no issue if only some of the `match` operators exist in the code
find_operators(.user_code, match = c("*", "/"))

# `uses_operator()` returns `TRUE` if `match` appears in the code
uses_operator(.user_code, match = "&&")
# and `FALSE` if it does not
uses_operator(.user_code, match = "&")
```

`find_parent` *Find the parent of an expression in R code*

Description

Find the parent of an expression in R code

Usage

```
find_parent(code, env = parent.frame())
```

Arguments

code	A character string containing the code to search within. Defaults to `.user_code`.
env	Environment in which to find `.user_code`. Most users will not need to use this argument.

Value

`find_parent()` returns a `gradecode_found` object, which can be passed to another `find_*()` function or to `fail_if_not_found()`.

Examples

```
.user_code <- "mtcars %>%
  mutate(kg = wt * 1000 / 2.2) %>%
  summarize(avg_kg = mean(kg, na.rm = TRUE))"

# The parent of an argument is the call containing the argument
.user_code %>%
  find_arguments(arg(na.rm = TRUE)) %>%
  find_parent()

# The parent of a function is the call containing the function and its arguments
.user_code %>%
  find_functions(mean) %>%
  find_parent()

.user_code %>%
  find_functions("summarize") %>%
  find_parent()

# The parent of an operator is the left- and right-hand side of the operator
.user_code %>%
  find_operators("*") %>%
  find_parent()

# But remember that, based on order of operations,
# an operator's parent may contain more than you expect
```

```
.user_code %>%
  find_operators("/") %>%
  find_parent()

# The same applies to pipes
.user_code %>%
  find_pipes() %>%
  find_parent()
```

`find_pipes` *Find pipes in R code*

Description

Find pipes in R code

Usage

```
find_pipes(code = .user_code, match = NULL, env = parent.frame())

uses_pipe(code = .user_code, match = NULL, env = parent.frame())
```

Arguments

code	A character string containing the code to search within. Defaults to `.user_code`.
match	Which pipe operator to find in code. Either `NULL` or a character vector containing pipe operators (`"%>%"`, the magrittr pipe, or `"|>"`, the base pipe). You can also include less common `magrittr` pipe operators: `"%<>%"`, `"%T>%"`, `"%!>%"` or `"%$%"`. If `match` is a character vector of length greater than 1, `find_pipes()` will find *any* of the specified operators. Defaults to `NULL`, which searches for `"|>"` or `"%>%"` (but not the less common magrittr pipe operators).
env	Environment in which to find `.user_code`. Most users will not need to use this argument.

Value

`find_pipes()` returns a `gradecode_found` object, which can be passed to another `find_*()` function or to `fail_if_not_found()`.

`uses_pipe()` returns a `TRUE`/`FALSE` value.

See Also

Other operator functions: `find_arithmetic()`, `find_assigns()`, `find_comparisons()`, `find_extractions()`, `find_infixes()`, `find_logical_operators()`, `find_operators()`

Examples

```
find_pipes("10 %>% rnorm() |> mean()")

# `match` lets you look for a specific operator
find_pipes("10 %>% rnorm() |> mean()", match = "%>%")
find_pipes("10 %>% rnorm() |> mean()", match = "|>")

# `uses_pipe()` returns a TRUE or FALSE value
uses_pipe("10 %>% rnorm() |> mean()")
uses_pipe("10 %>% rnorm() |> mean()", match = "%>%")
uses_pipe("10 %>% rnorm() |> mean()", match = "|>")
```

`standardize_arguments` *Standardize arguments to function calls*

Description

Standardize arguments to function calls

Usage

```
standardize_arguments(
  code,
  recurse = TRUE,
  add_defaults = TRUE,
  env = parent.frame()
)
```

Arguments

`code`	The code to standardize. Either a character string, a call, or a list of character strings or calls.
`recurse`	If `TRUE`, standardize arguments to all functions. If `FALSE`, standardize only arguments of the top-level function. Defaults to `TRUE`.
`add_defaults`	If `TRUE`, add missing default arguments. If `FALSE`, only standardize present arguments. Defaults to `TRUE`.
`env`	Environment in which to find function definitions and gradethis placeholder objects. Most users will not need to use this argument.

Value

An object of the same type as `code`.

Examples

```
.user_code <- "mean(rnorm(10))"
standardize_arguments(.user_code)
standardize_arguments(.user_code, recurse = FALSE)
standardize_arguments(.user_code, add_defaults = FALSE)
```

Index