```r
# Define function to calculate average value in a vector

average <- function(x) {
  sum(x) / length(x)
}
```

```r
# Handle non-numeric input

average <- function(x) {
  if (!is.numeric(x)) {
    return(NA)
  }
  sum(x) / length(x)
}
```

```r
# Message about returning NA

average <- function(x) {
  if (!is.numeric(x)) {
    message("`x` must be a numeric vector. Returning NA instead.")
    return(NA)
  }
  sum(x) / length(x)
}
```

```r
# Warn about returning NA

average <- function(x) {
  if (!is.numeric(x)) {
    warning("`x` must be a numeric vector. Returning NA instead.")
    return(NA)
  }
  sum(x) / length(x)
}
```

```r
# Stop instead of warn

average <- function(x) {
  if (!is.numeric(x)) {
    stop("`x` must be a numeric vector.")
  }
  sum(x) / length(x)
}
```

```r
# Handle NA values

average <- function(x) {
  if (!is.numeric(x)) {
    stop("`x` must be a numeric vector.")
  }
  if (any(is.na(x))) {
    warning("`x` contains one or more NA values.")
    return(NA)
  }
  sum(x) / length(x)
}
```

```r
# Write test function

source("average6.R")

test_average <- function() {
  if (average(c(1, 2, 3)) == 2) {
    cat("`average` passed test :)\n")
  } else {
    cat("`average` failed test :(\n")
  }
}

test_average()
```

```r
# Add test cases

source("average6.R")

test_average <- function() {
  if (average(c(1, 2, 3)) == 2) {
    cat("`average` passed test :)\n")
  } else {
    cat("`average` failed test :(\n")
  }

  if (average(c(-1, -2, -3)) == -2) {
    cat("`average` passed test :)\n")
  } else {
    cat("`average` failed test :(\n")
  }

  if (average(c(-1, 0, 1)) == 0) {
    cat("`average` passed test :)\n")
  } else {
    cat("`average` failed test :(\n")
  }
}

test_average()
```

```r
# Introduce test_that and add representative test cases to catch corner cases

source("average6.R")

test_that("`average` calculates mean", {
  expect_equal(average(c(1, 2, 3)), 2)
  expect_equal(average(c(-1, -2, -3)), -2)
  expect_equal(average(c(-1, 0, 1)), 0)
  expect_equal(average(c(-2, -1, 1, 2)), 0)
})
```

```r
# Test warning about NA values

source("average6.R")

test_that("`average` calculates mean", {
  expect_equal(average(c(1, 2, 3)), 2)
  expect_equal(average(c(-1, -2, -3)), -2)
  expect_equal(average(c(-1, 0, 1)), 0)
  expect_equal(average(c(-2, -1, 1, 2)), 0)
})

test_that("`average` warns about NAs in input", {
  expect_warning(average(c(1, NA, 3)))
  expect_warning(average(c(NA, NA, NA)))
})
```

```r
# Fix ordering of error handling

average <- function(x) {
  if (any(is.na(x))) {
    warning("`x` contains one or more NA values.")
    return(NA)
  }
  if (!is.numeric(x)) {
    stop("`x` must be a numeric vector.")
  }
  sum(x) / length(x)
}
```

```r
# Test NA return values

source("average7.R")

test_that("`average` calculates mean", {
  expect_equal(average(c(1, 2, 3)), 2)
  expect_equal(average(c(-1, -2, -3)), -2)
  expect_equal(average(c(-1, 0, 1)), 0)
  expect_equal(average(c(-2, -1, 1, 2)), 0)
})

test_that("`average` returns NA with NAs in input", {
  expect_equal(suppressWarnings(average(c(1, NA, 3))), NA)
  expect_equal(suppressWarnings(average(c(NA, NA, NA))), NA)
})

test_that("`average` warns about NAs in input", {
  expect_warning(average(c(1, NA, 3)))
  expect_warning(average(c(NA, NA, NA)))
})
```

```r
# Test stop if argument is non-numeric

source("average7.R")

test_that("`average` calculates mean", {
  expect_equal(average(c(1, 2, 3)), 2)
  expect_equal(average(c(-1, -2, -3)), -2)
  expect_equal(average(c(-1, 0, 1)), 0)
  expect_equal(average(c(-2, -1, 1, 2)), 0)
})

test_that("`average` returns NA with NAs in input", {
  expect_equal(suppressWarnings(average(c(1, NA, 3))), NA)
  expect_equal(suppressWarnings(average(c(NA, NA, NA))), NA)
})

test_that("`average` warns about NAs in input", {
  expect_warning(average(c(1, NA, 3)))
  expect_warning(average(c(NA, NA, NA)))
})

test_that("`average` stops if `x` is non-numeric", {
  expect_error(average(c("quack!")))
  expect_error(average(c("1", "2", "3")))
})
```

```r
# Demonstrates floating-point imprecision

print(0.3)
print(0.3, digits = 17)
```

```r
# Test doubles

source("average7.R")

test_that("`average` calculates mean", {
  expect_equal(average(c(1, 2, 3)), 2)
  expect_equal(average(c(-1, -2, -3)), -2)
  expect_equal(average(c(-1, 0, 1)), 0)
  expect_equal(average(c(-2, -1, 1, 2)), 0)
  expect_equal(average(c(0.1, 0.5)), 0.3)
})

test_that("`average` returns NA with NAs in input", {
  expect_equal(suppressWarnings(average(c(1, NA, 3))), NA)
  expect_equal(suppressWarnings(average(c(NA, NA, NA))), NA)
})

test_that("`average` warns about NAs in input", {
  expect_warning(average(c(1, NA, 3)))
  expect_warning(average(c(NA, NA, NA)))
})

test_that("`average` stops if `x` is non-numeric", {
  expect_error(average(c("quack!")))
  expect_error(average(c("1", "2", "3")))
})
```

```r
# Test greet

source("greet1.R")

test_that("`greet` says hello to a user", {
  expect_equal(greet("Carter"), "hello, Carter")
})
```

```r
# Greets a user

greet <- function(to) {
  return(paste("hello,", to))
}
```

```r
# Describe greet

source("greet1.R")

describe("greet()", {
  it("can say hello to a user", {
    name <- "Carter"
    expect_equal(greet(name), "hello, Carter")
  })
})
```

```r
# Describe greet

source("greet2.R")

describe("greet()", {
  it("can say hello to a user", {
    name <- "Carter"
    expect_equal(greet(name), "hello, Carter")
  })
  it("can say hello to the world", {
    expect_equal(greet(), "hello, world")
  })
})
```

```r
# Provides default argument value

greet <- function(to = "world") {
  return(paste("hello,", to))
}
```