~~~
[[questions]]
type = "ShortAnswer"
prompt.prompt = "What is the annotation you add to a function to indicate that it's a
test?"
answer.answer = "#[test]"
context = """
This informs scarb to treat the function as a test and not source code.
"""
id = "c7365cd4-c0cf-4d92-8e42-c47ab8936775"
[[questions]]
id = "cd77485f-723a-4978-8da6-c4ca3df44c44"
type = "MultipleChoice"
prompt.prompt = """
Let's say you have a function with the type signature:
```
fn f(x: usize) -> Result<usize, ByteArray>;
```
And you want to test that `f(0)` should return `Err(_)`.
Which of the following is **NOT** a valid way to test that?
"""
prompt.distractors = ["""
```
#[test]
fn test() {
   assert!(f(0).is_err());
}
```
""", """
```
#[test]
#[should_panic]
fn test() {
   f(0).unwrap();
}
```
  """, """
```
#[test]
fn test() {
   assert!(match f(0) {
      Ok(_) => false,
      Err(_) => true
   });
}
```
  """]
~~~

~~~
answer.answer = """
```
#[test]
#[should_err]
fn test() -> Result<usize, String> {
    f(0)
}
```
  """
context = """
`should_err` does not exist in Cairo &mdash; tests that return `Result` will pass even if
the result is an `Err`.
"""
[[questions]]
id = "4becac9f-5173-4439-bd1f-e1e9958423ab"
type = "MultipleChoice"
prompt.prompt = """
Does the test pass?
```
fn division_operation(number1: u16, number2: u16) -> u16 {
    if number2 == 0 {
        panic!("ZeroDivisionError not allowed!");
    }
    let result = number1 / number2;
    result
}
#[cfg(test)]
mod tests {
    use super::{division_operation};
    #[test]
    #[should_panic(expected: ("Zerodivisionerror not allowed!",))]
    fn test_division_operation() {
        division_operation(10, 0);
    }
}
```
"""
prompt.distractors = ["Yes"]
answer.answer = "No"
context = """
The expected string `"Zerodivisionerror not allowed!"` should be exactly
the same as the panic string `"ZeroDivisionError not allowed!"`
"""
[[questions]]
id = "0b3385b4-069f-4883-ab3f-6feb8ebf72f8"
type = "MultipleChoice"
~~~

~~~
prompt.prompt = """
What is the output when these tests are run with the command `scarb cairo-test -f test_`
```rust
#[cfg(test)]
mod tests {
    #[test]
    #[ignore]
    fn test_addition() {
        assert_ne!((5 + 4), 5);
    }
    #[test]
    fn division_function() {
        assert_eq!((10_u8 / 5), 2);
    }
    #[test]
    fn test_multiplication() {
        assert_ne!((3 * 2), 8);
        assert_eq!((5 * 5), 25);
    }
    #[test]
    fn test_subtraction() {
        assert!((12 - 11) == 1, "The first argument was false");
    }
}
```
"""
prompt.distractors = [
  "Error: test result: FAILED. 1 passed; 1 failed; 1 ignored;",
  "test result: ok. 1 passed; 0 failed; 1 ignored; 2 filtered out;",
  "test result: ok. 2 passed; 0 failed; 2 ignored; 0 filtered out;",
]
answer.answer = "test result: ok. 2 passed; 0 failed; 1 ignored; 1 filtered out;"
context = """
One ignored: `test_addition`, because it has the `ignore` attribute\n
One filtered out: `division_with_available_gas`, because its name doesn't match the
filter `test_`\n
Zero failed\n
Two passed: `test_multiplication` and `test_subtraction`, because all the conditions in
the assertions are true
"""
~~~