```
[[questions]]
type = "MultipleChoice"
prompt.prompt = """
Which line of code is correct to return the error code `2` and the error message `This is
the error message`?
"""
prompt.distractors = [
  """`panic(array![2, "This is the error message"]);`""",
  """`panic!("2", "This is the error message");`""",
  """`core::panic_with_felt252(2, 'This is the error message');`""",
]
answer.answer = """`panic(array![2, 'This is the error message']);`"""
context = """
The macro `panic!` takes one and only one `ByteArray` argument.
The function `core::panic_with_felt252` takes one and only one `felt252` argument.
The function `panic` takes an array of `felt252` elements as an argument and returns it.
"""
id = "5bf03883-59ab-4063-85e8-8c1d2fafa3e3"
[[questions]]
type = "MultipleChoice"
prompt.prompt = """
Which of the following is **NOT** a good reason to use a panic?
"""
prompt.distractors = [
  "The program is about to perform a dangerous operation",
  "The program should stop executing as soon as possible",
  "The program has reached an unrecoverable error state",
]
answer.answer = "The program has reached an error state which should be
communicated to a caller function"
context = """
A panic should not be used to communicate failure *within* the program, as it will
terminate the program immediately.
"""
id = "0e7bb9f9-af5a-441d-83b4-bcd0a97daf70"
[[questions]]
type = "MultipleChoice"
prompt.prompt = """
Which of the following programs will **NOT** cause a runtime panic?
```
fn main() {
    if true {
        panic!("2");
    }
    println!("This line shouldn't be reached");
}
```
```

```
```
fn main() {
    let arr = array![1, 2, 3];
    let index = 5;
    println!("Value at index {}: {}", index, arr[index]);
}
```
```
```
fn main() {
    let mut arr: Array<u128> = array![5];
    let index_to_access = 3;
    let value = match arr.get(index_to_access) {
        Option::Some(x) => { *x.unbox() },
        Option::None => { 0 }
    };
    println!("Value at index {} is {}", index_to_access, value);
}
```
"""
prompt.distractors = [
  "Program 1",
  "Program 2",
  "Program 2 and Program 3",
]
answer.answer = "Program 3"
context = """
Program 3 tries to access an element at an index that is out of bounds, but it matches
the
resulting `Option` and returns a default value `0` instead of panicking.
"""
id = "b6ee33f0-5b70-4286-a6d9-a08aad66f9c8"
[[questions]]
type = "MultipleChoice"
prompt.prompt = """
What is the purpose of the `panic_with` attribute in Cairo?
"""
prompt.distractors = [
  "To handle runtime errors gracefully",
  "To indicate that a function may panic",
  "To catch errors at compile time",
]
answer.answer = "To create a wrapper function that panics if the annotated function
returns `Option::None` or `Result::Err`"
context = """
If a function is annotated with `#[panic_with]`, a wrapper function is created that panics

if the
annotated function returns `Option::None` or `Result::Err`.
"""
id = "f618cfc9-3419-4cde-962d-72210879b3b3"