The Ed Jupyter marking system allows you to mark student code on a cell by cell basis. It uses special marker tags in the code cell itself to let the Ed system know that a particular cell requires a particular marking response.

The system **only** supports Python 3.

Example

Suppose we expect a student to create a `factorial` function in a particular cell.

Jupyter cell

```
### TEST FUNCTION: test_factorial
# DO NOT REMOVE THE LINE ABOVE

def factorial(n):
   # TODO your solution here
   pass
```

Testing script

```
def test_factorial(ctx):
    factorial = ctx.factorial

    assert factorial(0) == 1, 'Expected factorial(0) == 1'
    assert factorial(1) == 1, 'Expected factorial(1) == 1'
    assert factorial(2) == 2, 'Expected factorial(2) == 2'
```

Operation

The Ed marking system executes every single cell of the student's notebook, except for those containing the `### SKIP` marker.

When it encounters a cell with a `### TEST FUNCTION: test_name` marker, it will invoke the respective function defined in the marking script. The function must have a name that begins with `test_` .

An `EdContext` object is passed into the marking function and you can use it to access objects defined in the student's cell.

The `EdContext` object contains the following attributes:

The `EdContext` object contains the following attributes:

- `source` - the source code of the student's cell
- `stdout` - the standard output produced by the cell, e.g. through the `print()` function
- `stderr` - the standard error output produced by the cell

The `EdContext` object contains the following attributes:

- `ctx.evaluate(str)` - evaluate the specified expression in the context of the student's cell
- `ctx.execute(str)` - execute the specified statement in the context of the student's cell
- `ctx.__getattr(str)__` - same as `evaluate`, this allows you to directly access a function or variable defined by the student using `ctx.`*`name`*
- `ctx.__getitem(str)__` - same as `evaluate`, this allows you to directly access a function or variable defined by the student using `ctx['`*`name`*`']`

Return values

If your `test_` function returns `None` then the Ed system will assume the test has passed.

You may also return a Python dict with the following fields:

```
{
    ok: bool,       // required
    passed: bool,   // required
    name: str,      // optional
    feedback: str,  // optional
    score: int,     // optional
    max_score: int  // optional
}
```

Your test function may also return a list of test results to display multiple tests:

```
return [
    {
        'ok': True,
        'passed': True,
        'name': 'Test 1',
    },
    {
        'ok': True,
        'passed': True,
        'name': 'Test 2',
    },
]
```

Any unhandled exceptions that occur during the execution of cells or marking will result in the test failing.

Scoring

The score assigned to the student is `score` if `passed` and `zero` otherwise. If `max_score` is non-zero, then `score` will always be assigned to the student. This allows you to assign partial marks if the student does not pass the test.