```
# file_to_string

import pytest
from agentparse import file_to_string


# Test reading a .txt file
def test_file_to_string_txt(tmp_path):
    txt_file = tmp_path / "test.txt"
    txt_file.write_text("This is a test text file.")

    content = file_to_string(str(txt_file))
    assert content == "This is a test text file."


# Test reading a .csv file
def test_file_to_string_csv(tmp_path):
    csv_file = tmp_path / "test.csv"
    csv_file.write_text("name,age\nAlice,30\nBob,25")

    content = file_to_string(str(csv_file))
    assert content == "name,age\nAlice,30\nBob,25"


# Test reading a .json file
def test_file_to_string_json(tmp_path):
```

```
    json_file = tmp_path / "test.json"
    json_file.write_text('{"name": "Alice", "age": 30}')

    content = file_to_string(str(json_file))
    assert content == '{"name": "Alice", "age": 30}'


# Test reading a .pdf file (mocking the PdfReader)
def test_file_to_string_pdf(mocker, tmp_path):
    pdf_file = tmp_path / "test.pdf"
    pdf_file.write_bytes(
        b"%PDF-1.4\n1 0 obj\n<< /Type /Page >>\nendobj\n"
    )

    mocker.patch(
        "agentparse.PdfReader",
        return_value=mocker.Mock(
            pages=[
                mocker.Mock(
                    extract_text=lambda: "This is a test PDF text."
                )
            ]
        ),
    )

    content = file_to_string(str(pdf_file))
```

```
    assert "This is a test PDF text." in content


# Test reading a .xlsx file (mocking openpyxl)
def test_file_to_string_xlsx(mocker, tmp_path):
    xlsx_file = tmp_path / "test.xlsx"
    # Create a mock workbook
    mock_workbook = mocker.Mock()
    mock_sheet = mocker.Mock()
    mock_sheet.iter_rows.return_value = [(1, 2), (3, 4)]
    mock_workbook.sheetnames = ["Sheet1"]
    mock_workbook.__getitem__.return_value = mock_sheet

    mocker.patch(
        "agentparse.openpyxl.load_workbook",
        return_value=mock_workbook,
    )

    content = file_to_string(str(xlsx_file))
    assert "Sheet: Sheet1" in content
    assert "1,2" in content
    assert "3,4" in content


# Test unsupported file type
def test_file_to_string_unsupported_file(tmp_path):
```

```
    unsupported_file = tmp_path / "test.xyz"
    unsupported_file.write_text("This is an unsupported file type.")

    with pytest.raises(
        ValueError, match="Unsupported file type: .xyz"
    ):
        file_to_string(str(unsupported_file))


# Test error handling for non-existent file
def test_file_to_string_non_existent_file():
    with pytest.raises(FileNotFoundError):
        file_to_string("non_existent_file.txt")
```