

```

.FF..F.....E...FF.FF.....FF.....F....F.....F.
=====
ERROR: test_code_span_with_backtick_inside (test_inline.TestTextToTextNodes.test_code_span_with_backtick_inside)
Code span containing backtick uses double backticks
-----
Traceback (most recent call last):
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/test_inline.py", line 496, in test_code_span_with_backtick_inside
    nodes = text_to_textnodes(text)
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/inline.py", line 87, in text_to_textnodes
    nodes = split_nodes_delimiter(nodes, "`", TextType.CODE)
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/inline.py", line 18, in split_nodes_delimiter
    raise ValueError(f"Could Not find closing delimiter for {old_node}")
ValueError: Could Not find closing delimiter for TextNode(Use `code with ` backtick`, PLAIN, None)

=====
FAIL: test_block_to_block_type_heading (test_block.TestBlockFunctions.test_block_to_block_type_heading)
-----
Traceback (most recent call last):
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/test_block.py", line 105, in test_block_to_block_type_heading
    self.assertEqual(block_to_block_type("  # H1"), BlockType.HEADING)
    ~~~~~^~~~~~
AssertionError: <BlockType.PARAGRAPH: 'PARAGRAPH'> != <BlockType.HEADING: 'HEADING'>

=====
FAIL: test_block_to_block_type_ordered_list (test_block.TestBlockFunctions.test_block_to_block_type_ordered_list)
-----
Traceback (most recent call last):
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/test_block.py", line 215, in test_block_to_block_type_ordered_list
    self.assertEqual(
    ~~~~~^
        block_to_block_type("1. Item 1\n3. Item 2"), BlockType.ORDERED_LIST
        ~~~~~^
    ) # Non-sequential numbers but is still valid ordered list
    ^
AssertionError: <BlockType.PARAGRAPH: 'PARAGRAPH'> != <BlockType.ORDERED_LIST: 'ORDERED_LIST'>

```

```
=====
ERROR: test_escape_pipe_using_backtick (test_table.TestTableParsing.test_escape_pipe_using_backtick)
Include a pipe in a cell's content by escaping it, including inside other inline spans
-----
Traceback (most recent call last):
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/test_table.py", line 219, in test_escape_pipe_using_backtick
    table = get_table(md)
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/test_table.py", line 29, in get_table
    root = markdown_to_html_node(md)
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/block.py", line 251, in markdown_to_html_node
    html_node = get_block_html_node(block)
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/block.py", line 225, in get_block_html_node
    return parse_table_block(markdown_block)
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/block.py", line 137, in parse_table_block
    for n in text_to_textnodes(cell):
      ~~~~~~^~~~~~
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/inline.py", line 87, in text_to_textnodes
    nodes = split_nodes_delimiter(nodes, "`", TextType.CODE)
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/inline.py", line 18, in split_nodes_delimiter
    raise ValueError(f"Could Not find closing delimiter for {old_node}")
ValueError: Could Not find closing delimiter for TextNode(b '\, PLAIN, None)
```

```
FAIL: test_block_to_block_type_unordered_list (test_block.TestBlockFunctions.test_block_to_block_type_unordered_list)
```

```
-----  
Traceback (most recent call last):
```

```
File "/home/samriddha/Winter_Opensource/static_site_generator/src/test_block.py", line 199, in test_block_to_block_type_unordered_list  
    self.assertEqual(  
        ~~~~~~^  
        block_to_block_type("-Item"), BlockType.PARAGRAPH  
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  
    ) # Missing space between '-' and Item  
    ^
```

```
AssertionError: <BlockType.UNORDERED_LIST: 'UNORDERED_LIST'> != <BlockType.PARAGRAPH: 'PARAGRAPH'>
```

```
=====
```

```
FAIL: test_escaped_italics (test_inline.TestTextToTextNodes.test_escaped_italics)
```

```
Backslash escapes italics markers
```

```
-----  
Traceback (most recent call last):
```

```
File "/home/samriddha/Winter_Opensource/static_site_generator/src/test_inline.py", line 484, in test_escaped_italics  
    self.assertListEqual([TextNode("This is *not italicized*", TextType.PLAIN)], nodes)  
    ~~~~~~^ ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
AssertionError: Lists differ: [TextNode(This is *not italicized*, PLAIN, None)] != [TextNode(This is \*not italicized\*, PLAIN, None)]
```

```
First differing element 0:
```

```
TextNode(This is *not italicized*, PLAIN, None)
```

```
TextNode(This is \*not italicized\*, PLAIN, None)
```

```
- [TextNode(This is *not italicized*, PLAIN, None)]
```

```
+ [TextNode(This is \*not italicized\*, PLAIN, None)]
```

```
?
```

```
+
```

```
+
```

```
FAIL: test_image_with_empty_alt (test_inline.TestTextToTextNodes.test_image_with_empty_alt)
Images can have empty alt text
```

```
Traceback (most recent call last):
```

```
AssertionError: <TextType.PLAIN: 'PLAIN'> != <TextType.IMAGE: 'IMAGE'>
```

```
FAIL: test_link_with_empty_url (test_inline.TestTextToTextNodes.test_link_with_empty_url)
Links can have empty URLs
```

```
Traceback (most recent call last):
```

```
AssertionError: <TextType.PLAIN: 'PLAIN'> != <TextType.LINK: 'LINK'>
```

```
FAIL: test_link_with_parentheses_in_url (test_inline.TestTextToTextNodes.test_link_with_parentheses_in_url)
URLs can contain balanced parentheses
```

```
Traceback (most recent call last):
```

```
AssertionError: 'url(with' != 'url(with)parenns'
```

- url(with
- + url(with)paren

```
=====
FAIL: test_only_italic_with_asterik (test_inline.TestTextToTextNodes.test_only_italic_with_asterik)
Text with single asterik before and after is also treated as italics
-----
Traceback (most recent call last):
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/test_inline.py", line 468, in test_only_italic_with_asterik
    self.assertEqual([TextNode("italic word", TextType.ITALIC)], nodes)
    ~~~~~^~~~~~
AssertionError: Lists differ: [TextNode(italic word, ITALIC, None)] != [TextNode(*italic word*, PLAIN, None)]

First differing element 0:
TextNode(italic word, ITALIC, None)
TextNode(*italic word*, PLAIN, None)

- [TextNode(italic word, ITALIC, None)]
?              ^^^^^

+ [TextNode(*italic word*, PLAIN, None)]
?           +       +   +++ ^
=====
```

```

=====
FAIL: test_only_italic_with_no_flanking_whitespace (test_inline.TestTextToTextNodes.test_only_italic_with_no_flanking_whitespace)
Checks for flanking whitespace before '_' for rendering italic font.
-----
Traceback (most recent call last):
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/test_inline.py", line 461, in test_only_italic_with_no_flanking_whitespace
    self.assertEqual([TextNode("my_variable_name", TextType.PLAIN)], nodes)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: Lists differ: [TextNode(my_variable_name, PLAIN, None)] != [TextNode(my, PLAIN, None), TextNode(variable, ITALIC[32 chars]one)]

First differing element 0:
TextNode(my_variable_name, PLAIN, None)
TextNode(my, PLAIN, None)

Second list contains 2 additional elements.
First extra element 1:
TextNode(variable, ITALIC, None)

- [TextNode(my_variable_name, PLAIN, None)]
?  -----^

+ [TextNode(my, PLAIN, None),
?  ^

+ TextNode(variable, ITALIC, None),
+ TextNode(name, PLAIN, None)]
=====

```

```

=====
FAIL: test_table_mismatched_cells_with_alignment (test_table.TestTableParsing.test_table_mismatched_cells_with_alignment)
-----
Traceback (most recent call last):
  File "/home/sanriddha/Winter_Opensource/static_site_generator/src/test_table.py", line 198, in test_table_mismatched_cells_with_alignment
    self.assertEqual(
        ^
        [text_of(td) for td in element_children(row2)],
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
        ["L2", "C2", "R2"],
        ^^^^^^^^^^^^^^^^^^^^^
    )
    ^
AssertionError: Lists differ: ['L2', 'C2', 'R2', 'Extra'] != ['L2', 'C2', 'R2']

First list contains 1 additional elements.
First extra element 3:
'Extra'

- ['L2', 'C2', 'R2', 'Extra']
?      -----
+ ['L2', 'C2', 'R2']

=====

```

```

=====
FAIL: test_table_with_more_body_cells (test_table.TestTableParsing.test_table_with_more_body_cells)
-----
Traceback (most recent call last):
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/test_table.py", line 176, in test_table_with_more_body_cells
    self.assertEqual(
        ^
        [[text_of(td) for td in element_children(r)] for r in rows],
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
        [["a1", "b1"],["a2", "b2"]],
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    )
    ^
AssertionError: Lists differ: [['a1', 'b1', 'c1', 'd1'], ['a2', 'b2', 'c2']] != [['a1', 'b1'], ['a2', 'b2']]

First differing element 0:
['a1', 'b1', 'c1', 'd1']
['a1', 'b1']

- [['a1', 'b1', 'c1', 'd1'], ['a2', 'b2', 'c2']]
?  - - - - -
+ [['a1', 'b1'], ['a2', 'b2']]

=====
FAIL: test_link_no_url (test_textnode.TestTextNodeToHTML.test_link_no_url)
-----
Traceback (most recent call last):
  File "/home/samriddha/Winter_Opensource/static_site_generator/src/test_textnode.py", line 83, in test_link_no_url
    with self.assertRaises(ValueError):
           ^^^^^^^^^^^^^^^^^
AssertionError: ValueError not raised
-----

```