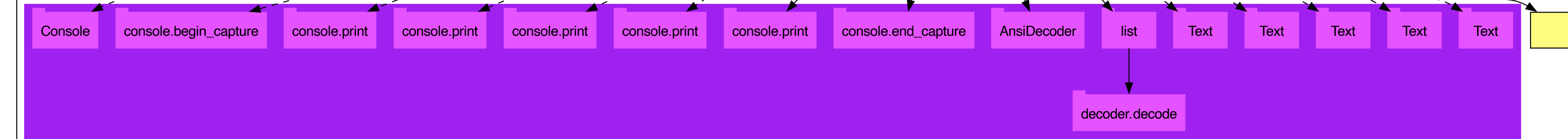
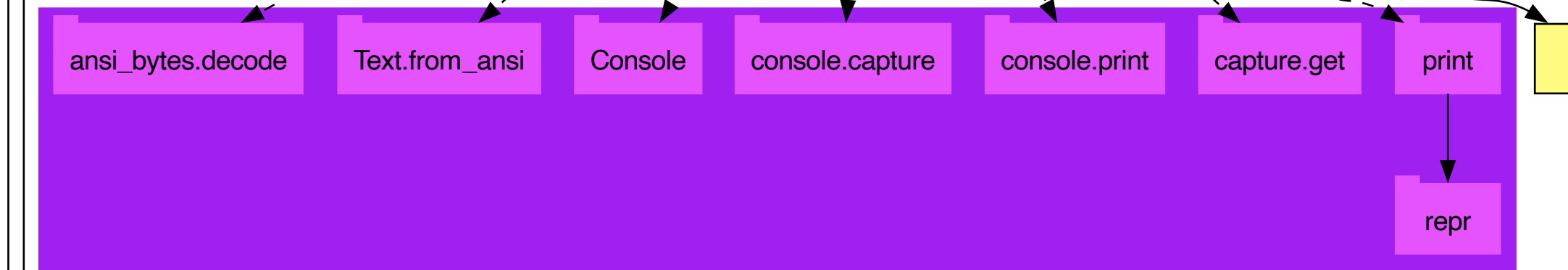


```
import pytest
from rich.ansi import AnsiDecoder
from rich.console import Console
from rich.style import Style
from rich.text import Span, Text
def test_decode():...
def test_decode_example():...
    @pytest.mark.parametrize('ansi_bytes, expected...', [(...
    @pytest.mark.parametrize('code', ['*0123456789;<=>?'])...
```

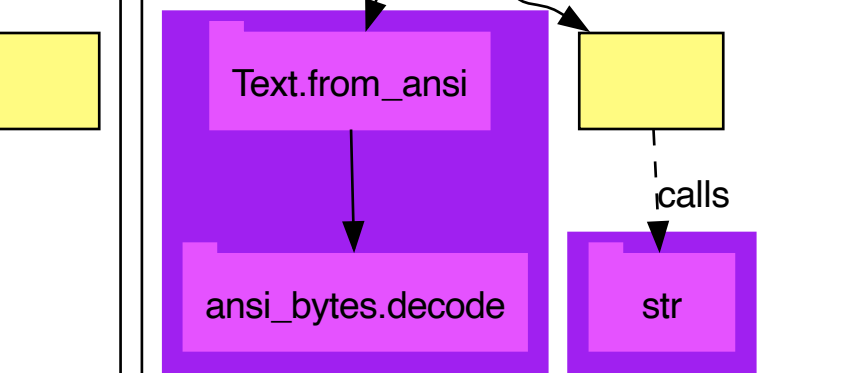
```
console = Console(force_terminal=True, legacy_windows=False, color_system='truecolor')
console.begin_capture()
console.print("Hello")
console.print("[b]foo[/b]")
console.print("[link http://example...")
console.print("[#ff0000 on color(20...")
console.print("[color(200) on #ff00...")
terminal_codes = console.end_capture()
decoder = AnsiDecoder()
lines = list(decoder.decode(terminal_codes))
expected = [Text('Hello'), Text('foo', spans=[Span(0, 3, ...'bold'
))), Text('bar', spans=[Span(0, 3, Style.parse(
'link http://example....'))]), Text('red', spans=[Span(0, 3, Style.
parse('#ff0000 on color(200....'))]), Text('red', spans=[Span(0, 3, Style.
parse('color(200) on #ff00....'))])]
assert lines == expected
```



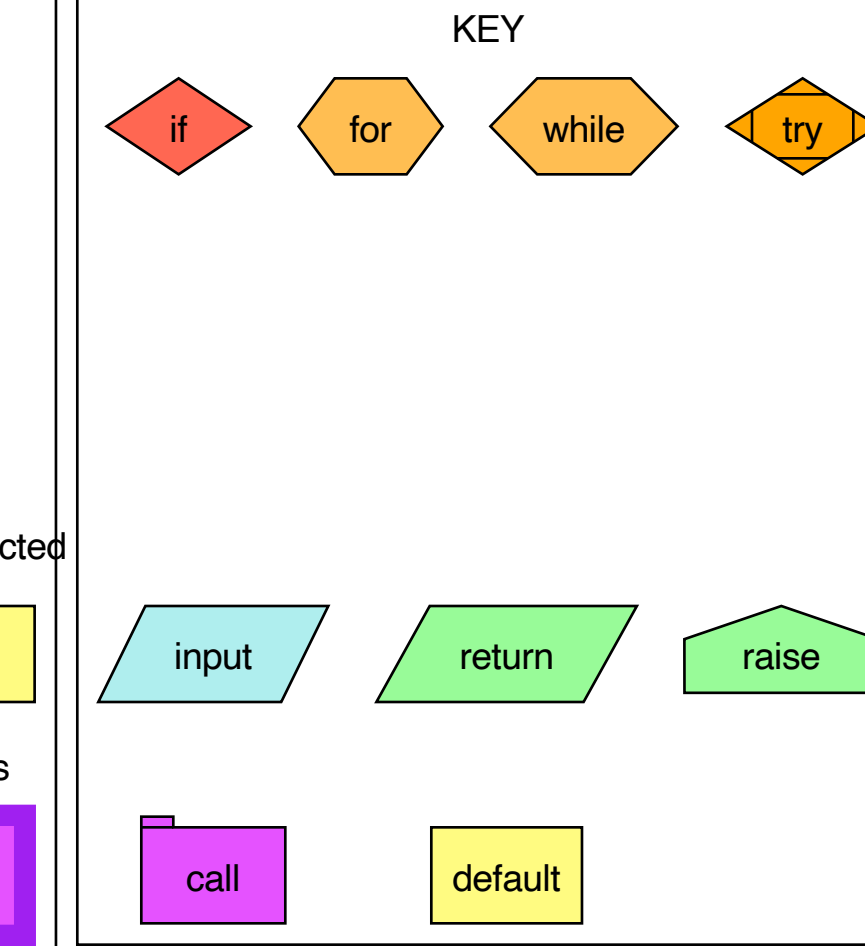
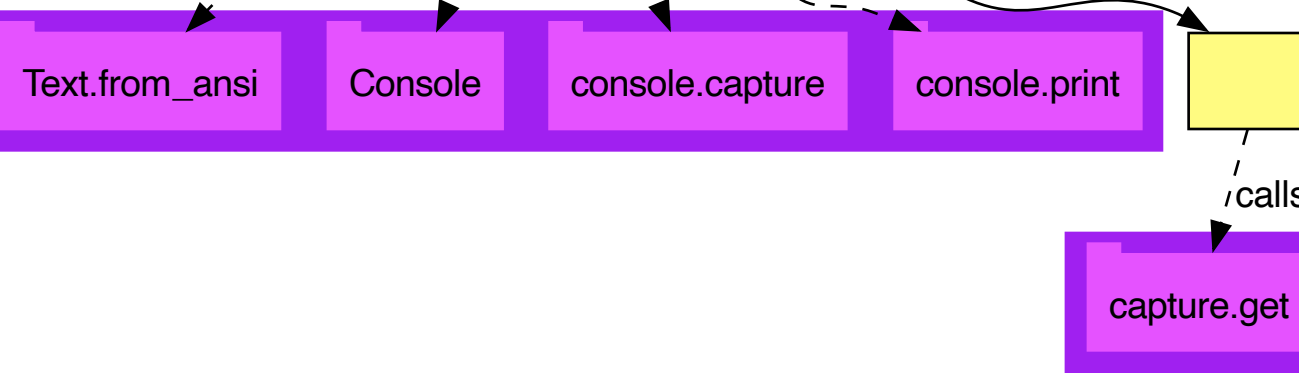
```
ansi_bytes = (
    b"x1b[01mx1b[KC:\Us...x1b[01mx1b[Kmainx1b[mx1b[K':
x1b[01mx1b[KC:...x1b[01mx1b[Kax1b[mx1b[K' x1b[01;35mx1b[K-..."
)
ansi_text = ansi_bytes.decode('utf-8')
text = Text.from_ansi(ansi_text)
console = Console(force_terminal=True, legacy_windows=False, color_system='truecolor')
console.print(text)
result = capture.get()
print(repr(result))
expected = ""
1mC:\Users\stefa...1mmain 0m':
1mC:\Users\stefa\AppData\Local\Temp\tmp3ydingba:3:5: 0m 1;35mwarning: 0munused variable ' 1ma 0m'
1;35m-Wunused-variable0m
3 | int 1;35ma0m0m=1;
| 1;35m^0m0m
""
assert result == expected
```



```
text = Text.from_ansi(ansi_bytes.decode())
assert str(text) == expected_text
```



```
text = Text.from_ansi(f'x1b{code}x')
console = Console(force_terminal=True)
console.print(text)
expected = 'x'
assert capture.get() == expected
```



progress