

1 \KKvUsePreset{TeXStyle} Test

```
Math Mode Test: $a + b = c$  
無視されない  
% 1行目のコメント保護とインデントのテスト  
\long\def\myprog#1{  
    \ifx#1\empty  
        \relax  
    \else  
        \message{Processing...}%  
        #1  
    \fi  
}  
無視されない  
\let\test\expandafter  
ここでも使えます*1。
```

1. 箇条書きの項目の中でも：

```
1 \begin{itemize}  
2     \item nested item  
3 \end{itemize}
```

```
1 % ここは行番号が出るはず  
2 \usepackage{xcolor}  
3 \usepackage{KKluaverb}  
4  
5 \begin{document}  
6 Hello, KKTTeX!  
7 \end{document}
```

1行完結のテスト：`\long\def\test#1{#1}`です。

記号のテスト：

```
{ [ \ ( \% \$ # & _ ^ ) / ] }
```

^{*1} 脚注の中でも

```
1 \def\bar{baz}  
2 これは\bar です
```

と書けるのが KKluaverb の強みです。

2 \KKvUsePreset{LuaStyle} Test

```
-- test comment
function KKV.encode_tail(str)
    local s = (str .. "\n"):gsub(ltjflg, "\n")
    return KKV.encode(s)
end

-- 制御構文と論理値のテスト
local function check_status(flag)
    if flag == true then
        print("Status is OK! ")
    elseif flag == false then
        print("Status is Error... ")
    else
        print("Status is nil.")
    end
end

1 -- ループと標準ライブラリのテスト
2 local data = { "Apple", "Banana", "Cherry" }
3
4 for i, v in ipairs(data) do
5     local msg = string.format("Item %d: %s", i, v)
6     table.insert(results, msg)
7     print(msg)
8 end

-- 演算子と文字列操作のテスト
function KKV.process(text)
    local len = #text
    local result = "{ " .. text .. " }" -- 連結演算子
    return result
end

local my_table = { key = "value", num = 123 }

1 -- KKV の内部ロジックをシミュレート
2 function KKV.encode(str)
3     local res = str:gsub(".", function(c)
4         return string.format("%%02X", string.byte(c))
5     end)
6     return res
end
```

```
7 end
8
9 local output = KKV.encode("Hello KKTTeX!")
10 print(output)
```