Date	01/08/2021, 20:14:50
Test	test write codeblock
Description	This test creates a PDF with a CodeBlock element in it.

```
import unittest
from datetime import datetime
from pathlib import Path

from borb.io.read.types import Decimal
from borb.pdf.canvas.layout.page_layout.multi_column_layout import SingleColumnLayout
from borb.pdf.canvas.layout.table.fixed_column_width_table import (
    FixedColumnWidthTable as Table,
)
from borb.pdf.canvas.layout.text.codeblock import CodeBlock
from borb.pdf.canvas.layout.text.paragraph import Paragraph
from borb.pdf.document import Document
from borb.pdf.page.page import Page
from borb.pdf.pdf import PDF


class TestWriteCodeblock(unittest.TestCase):
    """
    This test creates a PDF with a CodeBlock element in it.
    """

    def __init__(self, methodName="runTest"):
        super().__init__(methodName)
        # find output dir
        p: Path = Path(__file__).parent
        while "output" not in [x.stem for x in p.iterdir() if x.is_dir()]:
            p = p.parent
        p = p / "output"
        self.output_dir = Path(p, Path(__file__).stem.replace(".py", ""))
        if not self.output_dir.exists():
            self.output_dir.mkdir()

    def test_write_document(self):

        # create document
        pdf = Document()

        # add page
        page = Page()
        pdf.append_page(page)

        # layout
        layout = SingleColumnLayout(page)

        # add test information
        layout.add(
            Table(number_of_columns=2, number_of_rows=3)
            .add(Paragraph("Date", font="Helvetica-Bold"))
            .add(Paragraph(datetime.now().strftime("%d/%m/%Y, %H:%M:%S")))
            .add(Paragraph("Test", font="Helvetica-Bold"))
            .add(Paragraph(Path(__file__).stem))
            .add(Paragraph("Description", font="Helvetica-Bold"))
            .add(Paragraph("This test creates a PDF with a CodeBlock element in it."))
            .set_padding_on_all_cells(Decimal(2), Decimal(2), Decimal(2), Decimal(2))
        )

        # read self
        with open(__file__, "r") as self_file_handle:
            file_contents = self_file_handle.read()

        layout.add(
            CodeBlock(
                file_contents,
                font_size=Decimal(5),
            )
        )

        # determine output location
        out_file = self.output_dir / "output.pdf"

        # attempt to store PDF
        with open(out_file, "wb") as in_file_handle:
            PDF.dumps(in_file_handle, pdf)
```