report_objectObject

Name: report_object

Type: <class 'function'>

Sign: (func)

Args: {}

Doc:

This method takes in a function as arguments
passes to the function below which reads the content
of the function argument and prints out the content

Complx: {print: 0}

Source:

```
def report_object(func):
    '''This method takes in a function as arguments
    passes to the function below which reads the content
    of the function argument and prints out the content
    '''
    def helper(*args, **kwargs):
        lines = inspect.getsource(func)
        sig = signature(func)
        f = io.StringIO()
        with redirect_stdout(f):
            str(func(*args, **kwargs))

        i = 0
        for line in lines.split('\n'):
            if 'print(' in line:
                i += 1
        i -= 1
        i = str(i)
        compl = '{print: ' + i + '}'

        pdf = FPDF()

        pdf.add_page()
        pdf.set_font("Arial", size=12)
        pdf.cell(200, 10, txt=func.__name__+'Object', ln=1, align="L")

        pdf.cell(200, 10, txt='Name: ' + func.__name__, ln=1, align="L")
        pdf.cell(200, 10, txt='Type: ' + str(type(func)), ln=1, align="L")
        pdf.cell(200, 10, txt='Sign: ' + str(sig), ln=1, align="L")
        pdf.cell(200, 10, txt='Args: ' + str({
            k: v.default
            for k, v in sig.parameters.items()
            if v.default is not inspect.Parameter.empty
        }), ln=1, align="L")
        pdf.cell(200, 10, txt='Doc: ', ln=1, align="L")
        pdf.multi_cell(0, 5, str(inspect.getdoc(func)))
        pdf.cell(200, 10, txt='Complx: ' + compl, ln=1, align="L")
        pdf.cell(200, 10, txt='Source: ', ln=1, align="L")
        pdf.multi_cell(0, 5, lines)

        pdf.cell(200, 10, txt='Output: ' +
                 f.getvalue(), ln=1, align="L")
        pdf.output(func.__name__+"_object.pdf")

    return helper
```

Output: