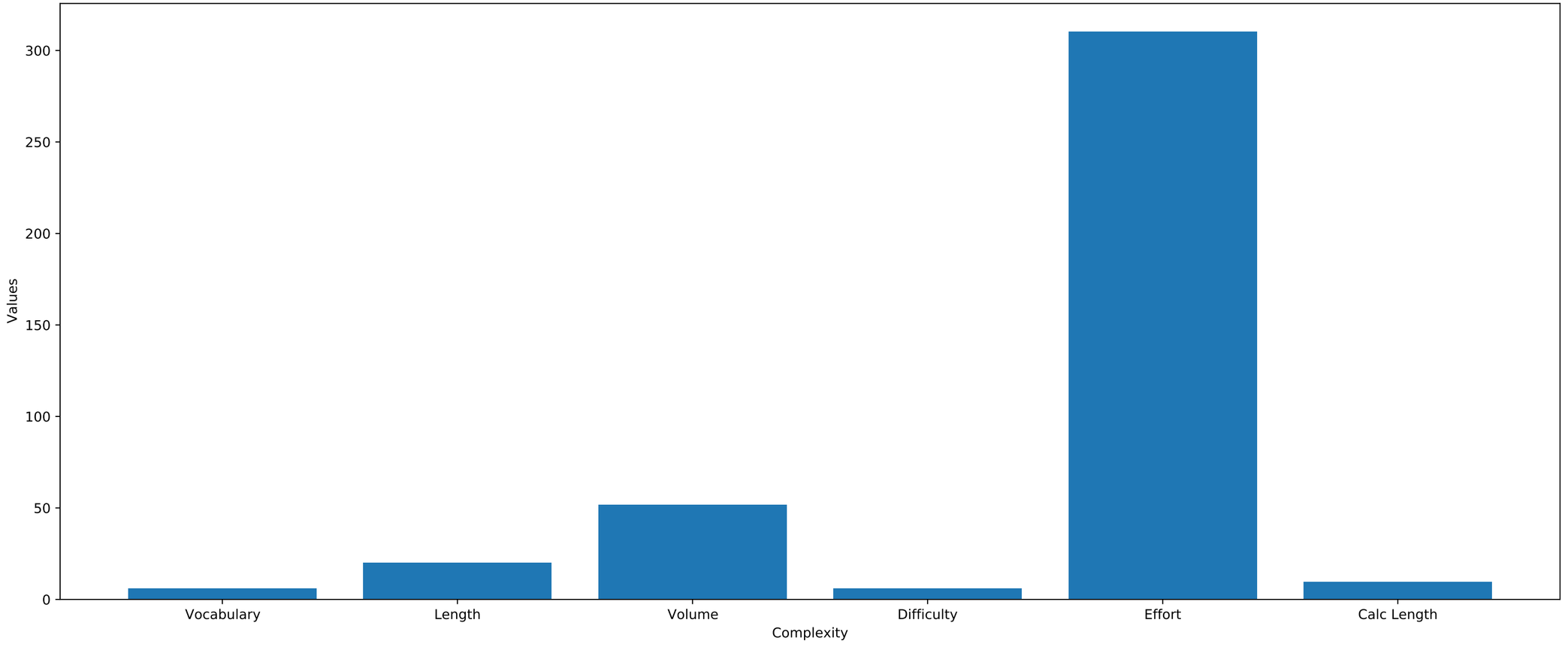

300
250
200
150
100
50
0
Values
Vocabulary
Length
Volume
Difficulty
Effort
Calc Length
Complexity

fooComplexity

Operator

if: 0

elif: 0

else: 0

try: 0

for: 0

with: 0

return: 0

def: 1

import: 0

calls: 4

arithmetic: 0

logic: 0

assign: 3

N1: 8

Program

Vocabulary: 6

Length: 20

Volume: 51.69925001442312

Difficulty: 6.0

Effort: 310.19550008653874

Difficulty: 9.509775004326938

fooObject

Name: foo

Type: <class 'function'>

Sign: ()

Args: {}

Doc:

None

Complx: {print: 0}

Source:

```
@report_object(output=True)
@report_complexity(operators=True, operands=False)
def foo():
    print("hello")
```

Output: hello

report_objectObject

Name: report_object

Type: <class 'function'>

Sign: (output)

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

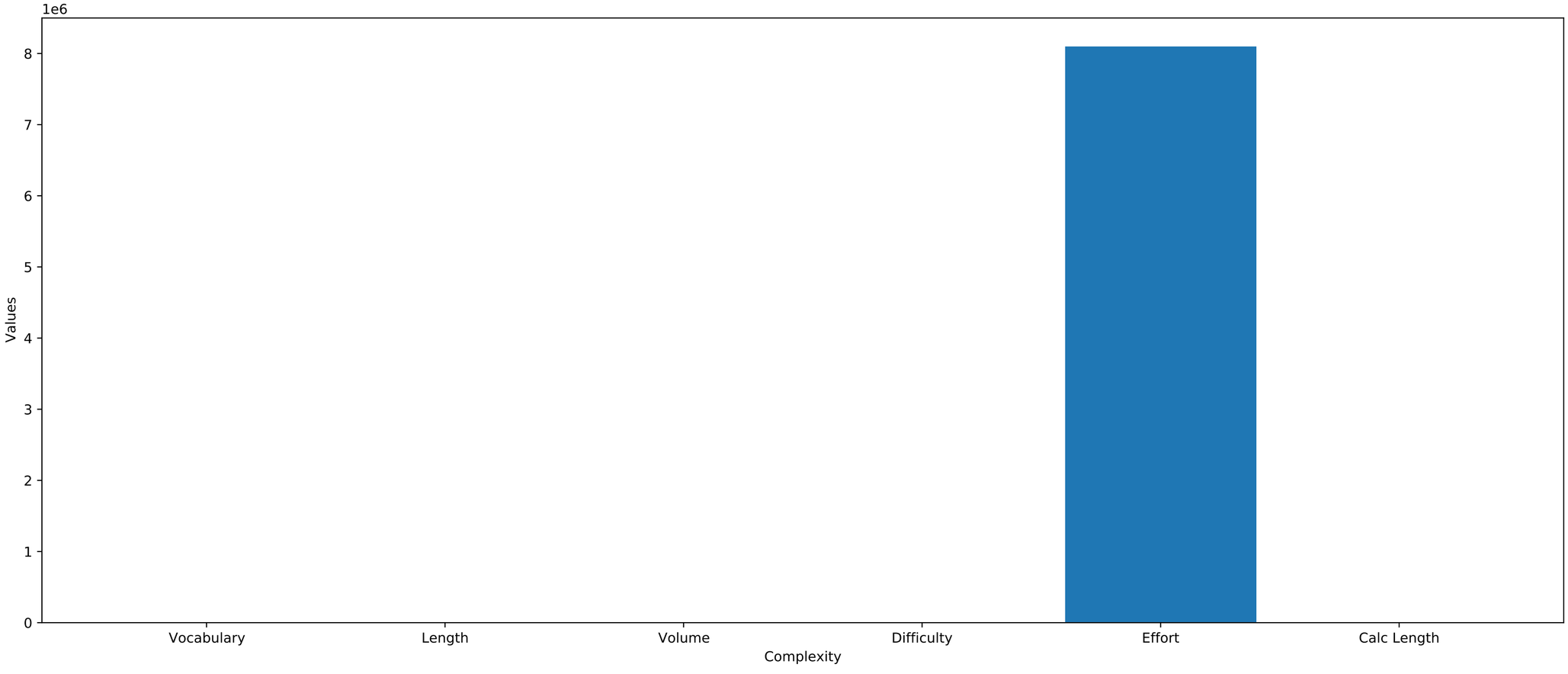
Complx: {print: 0}

Source:

```
def report_object(output):
    '''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 wrapper(func):

        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)
        if output:
            pdf.cell(200, 10, txt='Output: ' +
                     f.getvalue(), ln=1, align="L")
        filename = func.__name__+"_object.pdf"
        pdf.output(filename)
        filelist.append(filename)
        return func
    return helper
return wrapper
```

Output:


1e6
8
7
6
5
4
3
2
1
0
Values
Vocabulary
Length
Volume
Difficulty
Effort
Calc Length
Complexity

report_complexityComplexity

Operator

if: 28

elif: 0

else: 25

try: 0

for: 4

with: 1

return: 2

def: 2

import: 0

calls: 125

arithmetic: 93

logic: 28

assign: 188

N1: 496

Program

Vocabulary: 18

Length: 1330

Volume: 5546.000251918275

Difficulty: 1459.5

Effort: 8094387.367674723

Difficulty: 61.30296890880645