**ЛАБОРАТОРНА РОБОТА № 1**

З курсу

«Автоматизація тестування»

**Студент:** Однокозов О. Д.

**Група:** 41ІПЗ

**Факультет:** Математики, інформатики та фізики

**Викладач:** Кархут В.

Київ, 2022

math\_operations.js

const sum = (a,b) => (a+b);

const dif = (a,b) => (a-b);

const pow = (a,b) => (a\*b);

const div = (a,b) => (a/b);

module.exports.sum = sum;

module.exports.dif = dif;

module.exports.pow = pow;

module.exports.div = div;

math\_operations.test.js

var math\_operations = require('./math\_operations.js')

describe('sum = ', () => {

    test('Sum positive numbers', () => {

        expect(math\_operations.sum(1,2)).toBe(3);

    });

    test('Sum negative numbers', () => {

        expect(math\_operations.sum(1,-2)).toBe(-1);

    });

});

describe('dif = ', () => {

    test('Dif positive numbers', () => {

        expect(math\_operations.dif(15,2)).toBe(13);

    });

    test('Dif negative numbers', () => {

        expect(math\_operations.dif(-30,-15)).toBe(-15);

    });

});

describe('pow = ', () => {

    test('Pow positive numbers', () => {

        expect(math\_operations.pow(2,2)).toBe(4);

    });

    test('Pow negative numbers', () => {

        expect(math\_operations.pow(4,-2)).toBe(-8);

    });

});

describe('div = ', () => {

    test('Div positive numbers', () => {

        expect(math\_operations.div(4,2)).toBe(2);

    });

    test('Div negative numbers', () => {

        expect(math\_operations.div(6,-2)).toBe(-3);

    });

});

math\_operations\_param.test.js

var math\_operations = require('./math\_operations.js')

describe('sum = ', () => {

    test.each([[1,2,3],[2,3,5],[-1,-1,-2],])('sum test each', (a,b,expected) => {

        expect(math\_operations.sum(a,b)).toBe(expected);

    });

});

describe('dif = ', () => {

    test.each([[5,2,3],[10,3,7],[-5,-1,-4],])('dif test each', (a,b,expected) => {

        expect(math\_operations.dif(a,b)).toBe(expected);

    });

});

describe('pow = ', () => {

    test.each([[1,2,2],[14,3,42],[-1,-1,1],])('pow test each', (a,b,expected) => {

        expect(math\_operations.pow(a,b)).toBe(expected);

    });

});

describe('div = ', () => {

    test.each([[6,2,3],[15,5,3],[-15,-3,5],])('div test each', (a,b,expected) => {

        expect(math\_operations.div(a,b)).toBe(expected);

    });

});

array\_utils.js

const arrSum = (array) => {

    let result = 0;

    array.forEach(element => {

        result += element;

    });

    return result;

}

module.exports.arrSum = arrSum;

const arrPos = (array) => {

    let newArray = array.filter(value => (value >= 0));

    return newArray;

}

module.exports.arrPos = arrPos;

const arrNeg = (array) => {

    let newArray = array.filter(value => (value <= 0));

    return newArray;

}

module.exports.arrNeg = arrNeg;

array\_utils.test.js

var array\_utils = require('./array\_utils.js')

describe('array functions = ', () => {

    let array = [];

    for (i = 0; i < 50; i++) {

        let num = Math.floor(Math.random() \* 10);

        num = num > 5 ? num \* -1 : num;

        array.push(num);

    }

    test('array arrSum = ', () => {

        expect(array\_utils.arrSum(array)).toEqual(expect.any(Number));

    });

    test('array arrPos = ', () => {

        array\_utils.arrPos(array).forEach(x => {

            expect(x).toBeGreaterThanOrEqual(0);

        });

    });

    test('array arrNeg = ', () => {

        let arrNeg = array\_utils.arrNeg(array);

        arrNeg.forEach(x => {

            expect(x).toBeLessThanOrEqual(0);

        });

    });

});

string\_operations.test.js

describe("string tests: ", ()=> {

    test("palindrome?", ()=> {

        expect("kekekek".split("").reverse().join("")).toEqual("kekekek");

    })

    test("anagram", ()=> {

        const first = "anagram".split("");

        const result = "maranag".split("");

        expect(first).toEqual(expect.arrayContaining(result))

    })

})

grouped.test.js

var array\_utils = require('./array\_utils.js')

describe('array functions: ', () => {

    let array = [];

    for (i = 0; i < 50; i++) {

        let num = Math.floor(Math.random() \* 10);

        num = num > 5 ? num \* -1 : num;

        array.push(num);

    }

    test('array arrSum = ', () => {

        expect(array\_utils.arrSum(array)).toEqual(expect.any(Number));

    });

    test('array arrPos = ', () => {

        array\_utils.arrPos(array).forEach(x => {

            expect(x).toBeGreaterThanOrEqual(0);

        });

    });

    test('array arrNeg = ', () => {

        let arrNeg = array\_utils.arrNeg(array);

        arrNeg.forEach(x => {

            expect(x).toBeLessThanOrEqual(0);

        });

    });

});

describe("string tests: ", ()=> {

    test("palindrome?", ()=> {

        expect("kekekek".split("").reverse().join("")).toEqual("kekekek");

    })

    test("anagram", ()=> {

        const first = "anagram".split("");

        const result = "maranag".split("");

        expect(first).toEqual(expect.arrayContaining(result))

    })

})

functions.js

const isContain = (arr, obj) => {

    let answer = false;

    arr.forEach(element => {

        let keys\_array = Object.keys(element);

        let countOfСoincidence = 0;

        keys\_array.forEach((key) => {

            if(element[key] === obj[key] && obj.hasOwnProperty(key)) countOfСoincidence++;

        })

        if(keys\_array.length === countOfСoincidence) answer = true;

    });

    return answer;

}

const isContainSubString = (str,subStr) => {

    if(str.includes(subStr)) return true;

}

const returnObj = () => ({type: "shooter", game: "CS:GO"});

const callWithFuncOverArray = (arr, func) => {

    for (let i = 0; i < arr.length; i++) {

        arr[i] = func(arr[i]);

    }

    return arr;

}

const callWithFuncOverString = (str, func) => {

    str = str.split("");

    for (let i = 0; i < str.length; i++) {

        func(str[i]);

    }

}

module.exports.isContain = isContain;

module.exports.isContainSubString = isContainSubString;

module.exports.returnObj = returnObj;

module.exports.callWithFuncOverArray = callWithFuncOverArray;

module.exports.callWithFuncOverString = callWithFuncOverString;

functions.test.js

var functions = require('./functions.js')

describe("testing the final\_operations functions : ", ()=> {

    let arr = [

        {year: 4, status : "student", group : "121"},

        {animal: "dog", age : "14"},

        {type: "simulator", game: "life"}

    ];

    test("test the functions.isContainObj", ()=> {

        expect(functions.isContain(arr, {animal: "dog", age : "14"})).toBe(true);

    });

    test("test the functions.isContainSubString", ()=> {

        expect(functions.isContainSubString("hello", "ell")).toBe(true);

    });

    test("test the functions.returnObj", ()=> {

        expect(functions.returnObj()).toHaveProperty("type","shooter");

    });

    test("test the functions.callWithFuncOverArray", ()=> {

        const func = jest.fn(val => (val + 3));

        functions.callWithFuncOverArray([1,3,5,2,6], func);

        expect(func.mock.calls.length).toBe(5);

    });

    test("test the functions.callWithFuncOverString", ()=> {

        const func = jest.fn(val => (val));

        functions.callWithFuncOverString("Hello", func);

        console.log(func.mock.calls);

    });

    let objString = {

        firstName : "Volodymyr",

        secondName : "Zelensky",

        introduce : function(){console.log("Hello, I am ",this.firstName," ",this.secondName)}

    }

    test("test", ()=> {

        const func = jest.spyOn(objString, 'introduce');

        objString.introduce();

        objString.introduce();

        objString.introduce();

        console.log(func.mock.calls.length);

    });

})

Результат усіх тестів:

Text

Description automatically generated

GitHub Repository: <https://github.com/ElbruzOne/Automation-Testing>