Exercise 4 (non-mandatory)

Part 1 - OOP

Task 1

testPoint.html and *point.js* are files in the same folder:

Script i *testPoint.html*

```
<script src="point.js"></script>

<script>

    let p = createPoint(10, 5);
    let dist = p.distance(60, 20);
    let ang = p.angle(60, 20);

    console.log(dist);
    console.log(ang);
    console.log(p.x, p.y);

</script>
```

Script i *point.js*

```
let createPoint = function(x1, y1) {

    let point = {};
    point.x = x1;
    point.y = y1;

    // ----------------------------------------
    point.distance = function(x2, y2) {
        return Math.sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1));
    };

    // ----------------------------------------
    point.angle = function (x2, y2) {
        return 90 - radToDeg(Math.atan2(x2 - x1, y2 - y1));
    }

    // ----------------------------------------
    function radToDeg(rad) {
        return 180*rad/Math.PI;
    }

    return point;
}
```

a) Explain the code (you don't have to explain the inner workings of the functions in *point.js*). What is the advantage(s) of organizing the code in this way?
b) Which of the three functions inside *createPoint* are accessible on the outside (are public)?
c) What changes do you have to do in *point.js* to change it to a constructor-function (instead of a factory-function)?

Task 2

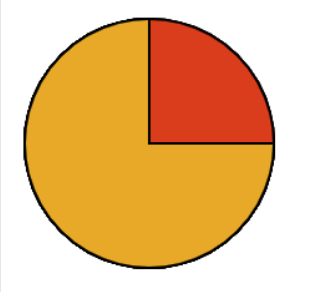
We have an HTML-file which contains a canvas-element with the id *cnv*. Create a constructor-function in a separate js-file (*pie.js*) so that we can use it like this in the HTML-file.

```
<script src="pie.js"></script>
<script>

    let cnv = document.getElementById("cnv");

    let pie = new Pie(cnv);
    pie.setValue(75); // 75 %
    pie.setBackColor("red");
    pie.setPieColor("orange");

</script>
```



Part 2 - GIT

Task 3

Create a project-folder on your computer and create some files - e.g. an HTML-file and js-file (you can use the folder and the files in the previous task if you want to). Also, create an empty repository on GitHub and copy the web URL for the repository. Use Git commands (in the console-window) to:

1. Create a Git repository in the project folder
2. Commit all the files in the project
3. Create a new remote with the name *origin* (use the URL from GitHub)
4. Push your master branch to GitHub
5. Create a new branch called *test* and make it the current branch
6. Add a new file to your project, e.g. a new HTML-file.
7. Commit and push the *test* branch to GitHub
8. Check that you have both the branches on GitHub.
9. Do some changes in your local *test* branch and do a commit.
10. Do some changes in your files in the *test* branch on GitHub and do a commit (simulating that some other member of the project has changed the files).
11. Pull the *test* branch, resolve any conflicts and do a commit.
12. Push the updated *test* branch to GitHub.

Task 4

Explain the terms: *commit*, *branch, checkout*, *push*, *pull* and *pull request*

Part 3 – Finding and correcting errors

Task 5

a) Read/run the code below. What's wrong? What kind of error message do you get – if any?

```
let myNumber = 5;
myNubmer += 2;
console.log(myNumber);
```

b) Read/run the code below. Assume that *btn* refers to a button on the webpage. What's wrong? What kind of error message do you get – if any?

```
let btn = document.getElementByID("btn");
console.log(btn.innerHTML);
```

c) Read/run the code below. What's wrong? What kind of error message do you get – if any?

```
let myNumber = 4;

if (myNumber < 2) {
   console.log("small number");
}
else (myNumber > 2) {
   console.log("big number");
}
```

d) Read/run the code below. What's wrong? What kind of error message do you get – if any?

```
for (let i; i < 10; i++) {
   console.log(i);
}
```

e) Read/run the code below. What's wrong? What kind of error message do you get – if any?

```
for (let i = 0; i > 10; i++) {
   console.log(i);
}
```

f) Read the code below. What's wrong? What kind of error message do you get – if any?

```
for (let i = 0; i < 10; i--) {
   console.log(i);
}
```

g) Read/run the code below. What's wrong? What kind of error message do you get – if any?

```
let myNumber;
myNumber += 2;
console.log(myNumber);
```

h) Read/run the code below. What's wrong? What kind of error message do you get – if any?

```
let myNumber1 = 2;
let myNumber2 = 0;
myNumber1 /= myNumber2;
console.log(myNumber1);
```

i) Read/run the code below. What's wrong? What kind of error message do you get – if any?

```
let myNumber = 2 * "45,5";
console.log(myNumber);
```

j) Read/run the code below. What's wrong? What kind of error message do you get – if any?

```
let myObj = {};
console.log(myObj.name);
```

k) In the code below, we have forgot to declare the variable with *let* or *var*. Do you get any error messages? Is there a difference between *let* and *var*? What happens if you add **`"use strict"`** at the top of the script?

```
myNumber = 5;
console.log(myNumber);
```

l) Open *errors1.html*. The program should convert meters to mm, but there are several errors in the program. Find and correct the errors so that the program works.

m) Open *errors2.html*. The program should draw 3 rows with 4 circles (using two while-loops), but it only draws one row. Find and correct the error (it's only one).