1 IJ macro code

```
print("hello world");
```

2 Inline code

The code `run("Gaussian Blur...", "sigma=2");` blurrs a image by sigma=2.

3 Include a code from a separate ijm file

```
/*

************* Temporal-Color Coder *******************************
Color code the temporal changes.

Kota Miura (miura@embl.de)
Centre for Molecular and Cellular Imaging, EMBL Heidelberg, Germany

If you publish a paper using this macro, please acknowledge.


---- INSTRUCTION ----

1. Open a stack (8 bit or 16 bit)
2. Run the macro
3. In the dialog choose one of the LUT for time coding.
        select frame range (default is full).
        check if you want to have color scale bar.

History

080212        created ver1 K_TimeRGBcolorcode.ijm
080213        stack slice range option added.
                time color code scale option added.

                future probable addiition: none-linear assigning of gray intensity to color intensity
                --> but this is same as doing contrast enhancement before processing.
```

```
101122  plugin'ified it
101123        fixed for cases when slices > 1 and frames == 1
*****************************************************************************
*/

var Glut = "Fire";        //default LUT
var Gstartf = 1;
var Gendf = 10;
var GFrameColorScaleCheck = 1;

macro "Time-Lapse Color Coder" {
        Stack.getDimensions(ww, hh, channels, slices, frames);
        if (channels > 1)
                exit("Cannot color-code multi-channel images!");
        //swap slices and frames in case:
        if ((slices > 1) && (frames == 1)) {
                frames = slices;
                slices = 1;
                Stack.setDimensions(1, slices, frames);
                //print("slices and frames swapped");
        }
        Gendf = frames;
        showDialog();
        if (Gstartf <1) Gstartf = 1;
        if (Gendf > frames) Gendf = frames;
        totalframes = Gendf - Gstartf + 1;
        calcslices = slices * totalframes;
        imgID = getImageID();

        calledFromBatchMode = is("Batch Mode");
        if (!calledFromBatchMode)
                setBatchMode(true);

        newImage("colored", "RGB White", ww, hh, calcslices);
        run("Stack to Hyperstack...", "order=xyczt(default)
         ↪  channels=1 slices="
                + slices + " frames=" + totalframes + "
                 ↪  display=Color");
        newimgID = getImageID();

        selectImage(imgID);
        run("Duplicate...", "duplicate");
        run("8-bit");
        imgID = getImageID();

        newImage("stamp", "8-bit White", 10, 10, 1);
        run(Glut);
        getLut(rA, gA, bA);
        close();
        nrA = newArray(256);
        ngA = newArray(256);
        nbA = newArray(256);

        newImage("temp", "8-bit White", ww, hh, 1);
        tempID = getImageID();

        for (i = 0; i < totalframes; i++) {
                colorscale = floor((256 / totalframes) * i);
                for (j = 0; j < 256; j++) {
                        intensityfactor = j / 255;
                        nrA[j] = round(rA[colorscale] *
                        ↪   intensityfactor);
                        ngA[j] = round(gA[colorscale] *
                        ↪   intensityfactor);
                        nbA[j] = round(bA[colorscale] *
                        ↪   intensityfactor);
                }

                for (j = 0; j < slices; j++) {
                        selectImage(imgID);
                        Stack.setPosition(1, j + 1, i +
                        ↪   Gstartf);
                        run("Select All");
                        run("Copy");

                        selectImage(tempID);
                        run("Paste");
                        setLut(nrA, ngA, nbA);
                        run("RGB Color");
                        run("Select All");
                        run("Copy");
                        run("8-bit");

                        selectImage(newimgID);
                        Stack.setPosition(1, j + 1, i + 1);
                        run("Select All");
                        run("Paste");
                }
        }

        selectImage(tempID);
        close();

        selectImage(imgID);
        close();

        selectImage(newimgID);

        run("Stack to Hyperstack...", "order=xyctz channels=1
        ↪  slices="
                + totalframes + " frames=" + slices + "
                ↪  display=Color");
        op = "start=1 stop=" + Gendf + " projection=[Max
        ↪  Intensity] all";
        run("Z Project...", op);
        if (slices > 1)
                run("Stack to Hyperstack...",
                ↪  "order=xyczt(default) channels=1 slices=" +
                ↪  slices
                        + " frames=1 display=Color");
        resultImageID = getImageID();

        selectImage(newimgID);
        close();

        selectImage(resultImageID);
        if (!calledFromBatchMode)
                setBatchMode("exit and display");

        if (GFrameColorScaleCheck)
                CreateScale(Glut, Gstartf, Gendf);
}

function showDialog() {
        lutA = getList("LUTs");
         Dialog.create("Color Code Settings");
        Dialog.addChoice("LUT", lutA);
        Dialog.addNumber("start frame", Gstartf);
        Dialog.addNumber("end frame", Gendf);
        Dialog.addCheckbox("Create Time Color Scale Bar",
        ↪  GFrameColorScaleCheck);
        Dialog.show();
         Glut = Dialog.getChoice();
        Gstartf = Dialog.getNumber();
        Gendf = Dialog.getNumber();
        GFrameColorScaleCheck = Dialog.getCheckbox();
}

function CreateScale(lutstr, beginf, endf){
        ww = 256;
        hh = 32;
        newImage("color time scale", "8-bit White", ww, hh, 1);
        for (j = 0; j < hh; j++) {
                for (i = 0; i < ww; i++) {
                        setPixel(i, j, i);
                }
        }
        run(lutstr);
        run("RGB Color");
        op = "width=" + ww + " height=" + (hh + 16) + "
         ↪  position=Top-Center zero";
        run("Canvas Size...", op);
        setFont("SansSerif", 12, "antiliased");
        run("Colors...", "foreground=white background=black
         ↪  selection=yellow");
        drawString("frame", round(ww / 2) - 12, hh + 16);
        drawString(leftPad(beginf, 3), 0, hh + 16);
        drawString(leftPad(endf, 3), ww - 24, hh + 16);

}

function leftPad(n, width) {
    s = "" + n;
    while (lengthOf(s) < width)
        s = "0" + s;
    return s;
}
```