

The Output of a Quite General Code Example

The following Julia code contains all operators, some strings and some comments. Moreover a few names of variables consist of greek letters, superscripts or subscripts. However the code itself is rather useless unless you want to test how Julia code is displayed by the jlcode package.

```
#= A comment that consists of several lines.
Hence it has to be a little longer, of course.
Otherwise it would be silly as it could fit in one line. =#

# This line will be my reference line, which will exactly contain 80 characters.

# This line is a comment containing operators like &, - and %
# A comment with the German word "Übergrößengeschäft" (store for oversizes)
# This line contains some special unicode characters:  $\alpha$ ,  $\gamma$ ,  $w^2$ ,  $\Delta_x$ 
# A comment with some numbers: 424, 1.23, 0.2E-5, -9.9e+9


# defining a useless testfunction
function Style_3rd_Test(x, y)

    myver = v"2.00"
    mystr = "A string with \"Übergrößengeschäft\",  $\alpha$ ,  $\pi$  and the + operator."
    mychar = 'W'
    mychar(2) = '€'
    mychar(3) = '⌘'
    z1vec = rand{Int8, 3}
    z2vec = Array{Int8}(3)
    z2vec[1:2] = [x % y, y \ x]
    t = x % 2 == 0 ? x : x + 1
    t = ~(t & x | y)
     $\alpha$  = 0.3
     $\beta^\alpha$  = 3.2e+5^ $\alpha$ 
    my $\beta$ var = 0.12E-2 *  $\beta^\alpha$ 
    z2vec[3] = y^2 + 3.4x*y - ( $\alpha$  + my $\beta$ var) * t/2
    z2vec = (z2vec + z1vec).^2
    if !(norm(z1vec) < norm(z2vec) + e + pi +  $\gamma$  +  $\phi$ )
        mystr = String( mystr, " signed ", mychar)
        println( mystr)
        return true;
    elseif norm(z2vec - z1vec) > 2.69
        println( String( "Error in ", myver, "!"))
        return false;
    end
end

end
```

Known Managable Issues

The following Julia code presents the known issues that can appear due to the nature of the listings package. Right now the jlcode package is not able to handle theses cases. However the issues can be fixed by the user himself.

Without fixing the issues one gets the following output:

```
function KnownManageableIssues(x, y)

    # KNOWN MANAGEABLE ISSUES:

    # identifier name with a number that follows
    # directly behind a special unicode character:
    myβ2ndvar = 2 * 0.12E-2 * xy

    # identifier name, which contains a γ, π or φ:
    myφvar+ = sqrt(2)

    # numbers in E-notation without using a + sign:
    evar = 3.99e400
    evar2 = 3.99E400

end
```

After fixing the issues the output looks as follows:

```
function KnownManageableIssues(x, y)

    # KNOWN MANAGEABLE ISSUES:

    # identifier name with a number that follows
    # directly behind a special unicode character:
    myβ2ndvar = 2 * 0.12E-2 * xy

    # identifier name, which contains a γ, π or φ:
    myφvar+ = sqrt(2)

    # numbers in E-notation without using a + sign:
    evar = 3.99e400
    evar2 = 3.99E400

end
```