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.

```julia
#= A comment that consits 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 contain exactly 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: €, α, γ, w², Δₓ, ϔ, x̃
# Mathematical characters: ∈, ∪, ∩, ∇, ⊗, ⊕, ≈, ‖, ..., ⋯, ⋮, ⋱, ⋰
# 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 = "String: \"Übergrößengeschäft\", α, π, ∪, φ̇  and the + operator."
    myset = Set( [2, 9, 12, 25, 33])
    x_in_myset = x ∈ myset
    myset⁽²⁾ = myset ∪ Set( [4, 8, 12, 33])
    z1vec = rand(Int8, 3)
    z₂vec = Array{Int8}(undef, 3)
    z₂vec[1:2] = [x % y, y \ x]
    t = x % 2 == 0 ? x : x + 1
    t̄ = ~(t & x | y) ⊻ y
    myτ̂var = t̄ & t
    α = √0.3
    βᵅ = 3.2e+5^α
    myβvar = ∛0.12E-2 * βᵅ
    z₂vec[3] = y^2 + 3.4x*y - (α + myβvar) * t/2
    z₂vec = (z₂vec + z1vec).^2
    if !(0.1 ≤ norm(z1vec') < norm(z₂vec') + e ÷ pi + γ + ϕ)
        mystr = String( mystr, " signed ")
        println( mystr)
        return true;
    elseif 3.2 ≥ norm(z₂vec - z1vec) > 2.69
        if norm(z₂vec - z1vec) ≠ 3.0
            println( String( "Error in ", myver, "!"))
        end
        return false;
    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)
  approx4π = 3.142

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

  # single characters
  mychar = 'W'
  mychar⁽²⁾ = '€'
  mychar⁽³⁾ = 'ϰ'

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:
  my$\beta$2ndvar = 2 * 0.12E-2 * x$^y$

  # identifier name, which contains a $\gamma$, $\pi$ or $\phi$:
  my$\phi$var$_+$ = sqrt(2)
  approx4$\pi$ = 3.142

  # numbers in E-notation without using a + sign:
  evar = 3.99e400
  evar$_2$ = 3.99E400

  # single characters
  mychar = 'W'
  mychar$^{(2)}$ = '€'
  mychar$^{(3)}$ = '$\varkappa$'

end
```