Rendering Codes Example Document

Marcobisky

February 2, 2025

Abstract

This is an example document to demonstrate different code highlighting effects.

1 Python highlighting

Listing 1: This is a literal python code

```python
# Python code snippet to demonstrate various grammar and keywords

import math  # Importing a library

class MyClass:
    """This is a docstring for a sample class."""
    def __init__(self, value=0):
        self.value = value  # Instance variable

    def calculate(self, x):
        """Method that uses a conditional, loop, and a lambda function."""
        result = [y ** 2 for y in range(x) if y % 2 == 0]  # List comprehension
        return list(map(lambda z: math.sqrt(z + self.value), result))  # Lambda and map

def main():
    try:
        obj = MyClass(value=10)
        print("Square roots:", obj.calculate(5))
    except ValueError as e:
        print(f"An error occurred: {e}")
    finally:
        print("Execution finished!")

# Running the main function
if __name__ == "__main__":
    main()
```

Listing 2: This is a python code from file

```python
import torch
import torch.nn as nn
import torch.optim as optim
```

```
import numpy as np
# The architecture of the CNN
class SmallCNN(nn.Module):
    # ... Defining the neural network

# Convert data to PyTorch tensors
data_tensor = torch.tensor(data, dtype=torch.float32)
labels_tensor = torch.tensor(labels, dtype=torch.long)

# Define the neural network, loss function, and optimizer, long ongosdj njdnflskjdbflsjahdbfjhadsfbgkjhd bvfjkhdafb ahf
model = SmallCNN()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Train the neural network
def train_model(model, criterion, optimizer, data, labels, epochs=1000):
    for epoch in range(epochs):
        model.train()
        optimizer.zero_grad()
        outputs = model(data)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        if (epoch+1) % 20 == 0:
            print(f'Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}')

train_model(model, criterion, optimizer, data_tensor, labels_tensor)
```

2 MATLAB highlighting

Listing 3: This is a literal MATLAB code

```
% MATLAB code snippet to demonstrate various grammar and keywords

classdef MyClass
    % A sample class demonstrating properties, methods, and control flow

    properties
        Value % Class property
    end

    methods
        function obj = MyClass(val)
            % Constructor method
            if nargin > 0
                obj.Value = val;
            else
                obj.Value = 0;
            end
        end
```

```
        function result = calculate(obj, n)
            % Method that performs a calculation using a loop and conditionals
            result = zeros(1, n);
            for i = 1:n
                if mod(i, 2) == 0
                    result(i) = sqrt(i + obj.Value); % Square root for even
   numbers
                else
                    result(i) = i^2; % Square for odd numbers
                end
            end
        end
    end
end

% Script to use the class
clc; clear;
try
    obj = MyClass(10);
    disp('Results:');
    disp(obj.calculate(5));
catch ME
    disp(['Error occurred: ', ME.message]);
end
```

Listing 4: This is a MATLAB code from file

```
% MATLAB code snippet to demonstrate functions, plotting, and control flow

function main()
    % Main function to demonstrate MATLAB functionality

    % Define an anonymous function
    f = @(x) sin(x) + cos(x);

    % Call a custom function and generate a plot
    x = linspace(0, 2*pi, 100);
    y = customFunction(x, f);

    % Plot the result
    figure;
    plot(x, y, 'LineWidth', 1.5);
    title('Plot of sin(x) + cos(x)');
    xlabel('x');
    ylabel('y');
    grid on;
end

function y = customFunction(x, func)
    % A custom function with a loop and a switch-case statement
    y = zeros(size(x));
```

```
    for i = 1:length(x)
        switch true
            case x(i) < pi
                y(i) = func(x(i)); % Apply the anonymous function
            case x(i) >= pi
                y(i) = func(x(i)) * 2; % Double the result for x >= pi
        end
    end
end

% Call the main function
main();
```

3 Bash highlighting

```
$ sudo apt-get update
$ sudo apt-get install python3
```

4 C++ highlighting

Listing 5: This is a literal C++

```
#include <iostream>
#include <vector>
#include <cmath>
#include <stdexcept>

// A sample class to demonstrate properties, methods, and control flow
class MyClass {
private:
    int value; // Private property

public:
    // Constructor
    MyClass(int val = 0) : value(val) {}

    // Getter
    int getValue() const {
        return value;
    }

    // Method to perform calculations
    std::vector<double> calculate(int n) const {
        if (n <= 0) {
            throw std::invalid_argument("n must be greater than 0");
        }

        std::vector<double> result(n);
        for (int i = 0; i < n; ++i) {
```

```
            if (i % 2 == 0) {
                result[i] = std::sqrt(i + value); // Square root for even
   indices
            } else {
                result[i] = std::pow(i, 2); // Square for odd indices
            }
        }
        return result;
    }
};

int main() {
    try {
        MyClass obj(10);
        std::cout << "Value: " << obj.getValue() << std::endl;

        auto results = obj.calculate(5);
        std::cout << "Results: ";
        for (const auto& val : results) {
            std::cout << val << " ";
        }
        std::cout << std::endl;
    } catch (const std::exception& e) {
        std::cerr << "Error: " << e.what() << std::endl;
    }

    return 0;
}
```

Listing 6: This is a C++ code from file

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

// A function to demonstrate lambdas and algorithms
std::vector<double> generateValues(int n, std::function<double(int)> func) {
    std::vector<double> values(n);
    for (int i = 0; i < n; ++i) {
        values[i] = func(i);
    }
    return values;
}

// A function to demonstrate pointers and dynamic memory
double* computeSquares(int n) {
    double* squares = new double[n];
    for (int i = 0; i < n; ++i) {
        squares[i] = i * i; // Compute square of i
    }
    return squares;
```

```cpp
}

int main() {
    // Demonstrating lambdas and STL
    auto values = generateValues(10, [](int x) { return x * 2.5; });
    std::cout << "Generated Values: ";
    for (const auto& val : values) {
        std::cout << val << " ";
    }
    std::cout << std::endl;

    // Demonstrating dynamic memory and pointers
    int n = 5;
    double* squares = computeSquares(n);
    std::cout << "Squares: ";
    for (int i = 0; i < n; ++i) {
        std::cout << squares[i] << " ";
    }
    std::cout << std::endl;

    delete[] squares; // Free dynamically allocated memory

    return 0;
}
```

5 VHDL highlighting

Listing 7: This is a literal VHDL

```vhdl
-- Dff.vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

entity Dff is
    port (
        D   : in std_logic;  -- Data input
        clk : in std_logic;  -- Clock input
        Q   : out std_logic  -- Output
    );
end Dff;

architecture Behavioral of Dff is
    signal Q_internal : std_logic := '0';  -- Internal signal for flip-flop
    state
begin
    process(clk)
    begin
        if rising_edge(clk) then
            Q_internal <= D;  -- Update internal state on clock's rising
   edge
        end if;
```

```
    end process;

    Q <= Q_internal;  -- Assign internal state to output
end Behavioral;
```

Listing 8: This is a VHDL code from file

```
-- adder2bit.vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

entity adder2bit is
    port (
        Ah: in std_logic;
        Al: in std_logic;
        Bh: in std_logic;
        Bl: in std_logic;
        C: out std_logic;
        Q1: out std_logic;
        Q0: out std_logic);
end adder2bit;

architecture Behavioral of adder2bit is
    signal sel: std_logic_vector(3 downto 0); -- Combine inputs into a
   single signal
begin
    -- Combine inputs into a 4-bit signal
    sel <= Ah & Al & Bh & Bl;

    -- Enumerate all possible input combinations and assign Q0
    Q0 <= '1' when sel = "0001" or
                  sel = "0011" or
                  sel = "0100" or
                  sel = "0110" or
                  sel = "1001" or
                  sel = "1011" or
                  sel = "1100" or
                  sel = "1110" else
          '0';

    -- Enumerate all possible input combinations and assign Q1
    Q1 <= '1' when sel = "0010" or
                  sel = "0011" or
                  sel = "0101" or
                  sel = "0110" or
                  sel = "1000" or
                  sel = "1001" or
                  sel = "1100" or
                  sel = "1111" else
          '0';

    -- Enumerate all possible input combinations and assign C
```

```
45      C <= '1' when sel = "0111" or
46                    sel = "1010" or
47                    sel = "1011" or
48                    sel = "1101" or
49                    sel = "1110" or
50                    sel = "1111" else
51              '0';
52  end Behavioral;
```