

# Rendering Codes Example Document

Marcobisky

November 25, 2024

## Abstract

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

## 1 Python highlighting

Listing 1: This is a literal python code

```
1 # Python code snippet to demonstrate various grammar and keywords
2
3 import math # Importing a library
4
5 class MyClass:
6     """This is a docstring for a sample class."""
7     def __init__(self, value=0):
8         self.value = value # Instance variable
9
10    def calculate(self, x):
11        """Method that uses a conditional, loop, and a lambda function."""
12        result = [y ** 2 for y in range(x) if y % 2 == 0] # List
13        return list(map(lambda z: math.sqrt(z + self.value), result)) #
14        Lambda and map
15
16 def main():
17     try:
18         obj = MyClass(value=10)
19         print("Square roots:", obj.calculate(5))
20     except ValueError as e:
21         print(f"An error occurred: {e}")
22     finally:
23         print("Execution finished!")
24
25 # Running the main function
26 if __name__ == "__main__":
27     main()
```

Listing 2: This is a python code from file

```
1 import torch
2 import torch.nn as nn
3 import torch.optim as optim
```

```

4 import numpy as np
5 # The architecture of the CNN
6 class SmallCNN(nn.Module):
7     # ... Defining the neural network
8
9 # Convert data to PyTorch tensors
10 data_tensor = torch.tensor(data, dtype=torch.float32)
11 labels_tensor = torch.tensor(labels, dtype=torch.long)
12
13 # Define the neural network, loss function, and optimizer, long ongsdj
    njdnflskjdbflsjahdbfjhadsfbgkjhd bvfjkhda fb ahf
14 model = SmallCNN()
15 criterion = nn.CrossEntropyLoss()
16 optimizer = optim.Adam(model.parameters(), lr=0.001)
17
18 # Train the neural network
19 def train_model(model, criterion, optimizer, data, labels, epochs=1000):
20     for epoch in range(epochs):
21         model.train()
22         optimizer.zero_grad()
23         outputs = model(data)
24         loss = criterion(outputs, labels)
25         loss.backward()
26         optimizer.step()
27         if (epoch+1) % 20 == 0:
28             print(f'Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}')
29
30 train_model(model, criterion, optimizer, data_tensor, labels_tensor)

```

## 2 MATLAB highlighting

Listing 3: This is a literal MATLAB code

```

1 % MATLAB code snippet to demonstrate various grammar and keywords
2
3 classdef MyClass
4     % A sample class demonstrating properties, methods, and control flow
5
6     properties
7         Value % Class property
8     end
9
10    methods
11        function obj = MyClass(val)
12            % Constructor method
13            if nargin > 0
14                obj.Value = val;
15            else
16                obj.Value = 0;
17            end
18        end
19    end

```

```

19         function result = calculate(obj, n)
20             % Method that performs a calculation using a loop and conditionals
21             result = zeros(1, n);
22             for i = 1:n
23                 if mod(i, 2) == 0
24                     result(i) = sqrt(i + obj.Value); % Square root for even
25 numbers
26                     else
27                         result(i) = i^2; % Square for odd numbers
28                     end
29             end
30         end
31     end
32 end
33
34 % Script to use the class
35 clc; clear;
36 try
37     obj = MyClass(10);
38     disp('Results:');
39     disp(obj.calculate(5));
40 catch ME
41     disp(['Error occurred: ', ME.message]);
42 end

```

Listing 4: This is a MATLAB code from file

```

1 % MATLAB code snippet to demonstrate functions, plotting, and control flow
2
3 function main()
4     % Main function to demonstrate MATLAB functionality
5
6     % Define an anonymous function
7     f = @(x) sin(x) + cos(x);
8
9     % Call a custom function and generate a plot
10    x = linspace(0, 2*pi, 100);
11    y = customFunction(x, f);
12
13    % Plot the result
14    figure;
15    plot(x, y, 'LineWidth', 1.5);
16    title('Plot of sin(x) + cos(x)');
17    xlabel('x');
18    ylabel('y');
19    grid on;
20 end
21
22 function y = customFunction(x, func)
23     % A custom function with a loop and a switch-case statement
24     y = zeros(size(x));

```

```

25     for i = 1:length(x)
26         switch true
27             case x(i) < pi
28                 y(i) = func(x(i)); % Apply the anonymous function
29             case x(i) >= pi
30                 y(i) = func(x(i)) * 2; % Double the result for x >= pi
31         end
32     end
33 end
34
35 % Call the main function
36 main();

```

### 3 Bash highlighting

```

1 $ sudo apt-get update
2 $ sudo apt-get install python3

```

### 4 C++ highlighting

Listing 5: This is a literal C++

```

1 #include <iostream>
2 #include <vector>
3 #include <cmath>
4 #include <stdexcept>
5
6 // A sample class to demonstrate properties, methods, and control flow
7 class MyClass {
8 private:
9     int value; // Private property
10
11 public:
12     // Constructor
13     MyClass(int val = 0) : value(val) {}
14
15     // Getter
16     int getValue() const {
17         return value;
18     }
19
20     // Method to perform calculations
21     std::vector<double> calculate(int n) const {
22         if (n <= 0) {
23             throw std::invalid_argument("n must be greater than 0");
24         }
25
26         std::vector<double> result(n);
27         for (int i = 0; i < n; ++i) {

```

```

28         if (i % 2 == 0) {
29             result[i] = std::sqrt(i + value); // Square root for even
indices
30         } else {
31             result[i] = std::pow(i, 2); // Square for odd indices
32         }
33     }
34     return result;
35 }
36 };
37
38 int main() {
39     try {
40         MyClass obj(10);
41         std::cout << "Value: " << obj.getValue() << std::endl;
42
43         auto results = obj.calculate(5);
44         std::cout << "Results: ";
45         for (const auto& val : results) {
46             std::cout << val << " ";
47         }
48         std::cout << std::endl;
49     } catch (const std::exception& e) {
50         std::cerr << "Error: " << e.what() << std::endl;
51     }
52
53     return 0;
54 }

```

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

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  #include <functional>
5
6  // A function to demonstrate lambdas and algorithms
7  std::vector<double> generateValues(int n, std::function<double(int)> func) {
8      std::vector<double> values(n);
9      for (int i = 0; i < n; ++i) {
10         values[i] = func(i);
11     }
12     return values;
13 }
14
15 // A function to demonstrate pointers and dynamic memory
16 double* computeSquares(int n) {
17     double* squares = new double[n];
18     for (int i = 0; i < n; ++i) {
19         squares[i] = i * i; // Compute square of i
20     }
21     return squares;

```

```

22 }
23
24 int main() {
25     // Demonstrating lambdas and STL
26     auto values = generateValues(10, [](int x) { return x * 2.5; });
27     std::cout << "Generated Values: ";
28     for (const auto& val : values) {
29         std::cout << val << " ";
30     }
31     std::cout << std::endl;
32
33     // Demonstrating dynamic memory and pointers
34     int n = 5;
35     double* squares = computeSquares(n);
36     std::cout << "Squares: ";
37     for (int i = 0; i < n; ++i) {
38         std::cout << squares[i] << " ";
39     }
40     std::cout << std::endl;
41
42     delete[] squares; // Free dynamically allocated memory
43
44     return 0;
45 }

```