```
# Function with optional argument
1
   function greet(name="User")
2
        println("Hello, $name")
3
4
5
   # struct and its usage
6
   struct Point{T}
7
        x::T
8
9
        y::T
   end
10
   p1 = Point(1.0, 3.7)
11
   println("Point: [$(p1.x), $(p1.y)]")
12
13
   # Array and array comprehension
14
   numbers = [1, 2, 3, 4, 5]
15
   squared_numbers = [x^2 for x in numbers]
16
17
   # Dictionaries and their usage
18
   person = Dict("name" ⇒ "John", "age" ⇒ 30)
19
    for (key, value) in person
20
        println("$key: $value")
21
   end
22
23
   # Try-catch block
24
25
        result = 10 / 0
26
   catch e
27
        result = "Error: $e"
28
29
30
   # While loop with break statement
31
   i = 1
32
   while i \leq 5
33
        if i = 3
34
35
            break
36
        end
37
        println(i)
        i += 1
38
   end
39
40
41
   # Anonymous function (lambda) and map function
   double = x \rightarrow 2x
42
   numbers = [1, 2, 3, 4, 5]
43
   doubled_numbers = map(double, numbers)
44
45
   # Set data structure
46
   fruits = Set(["apple", "banana", Nothing])
47
48
   # Tuple unpacking and multi-line string
49
   (x, y) = (10, 20)
50
   multi_line_string = """
51
   This is a multi-line
52
   string in Julia.
53
54
```

Listing 1: Example - Julia (TreeSitter powered highlighting)

```
# Function with optional argument
    function greet(name="User")
2
        println("Hello, $name")
3
    end
4
    # struct and its usage
    struct Point{T}
        x::T
        y::T
    end
10
    p1 = Point(1.0, 3.7)
11
    println("Point: [$(p1.x), $(p1.y)]")
12
    # Array and array comprehension
14
    numbers = [1, 2, 3, 4, 5]
15
    squared_numbers = [x^2 for x in numbers]
16
17
    # Dictionaries and their usage
18
    person = Dict("name" ⇒ "John", "age" ⇒ 30)
19
    for (key, value) in person
20
        println("$key: $value")
21
    end
22
23
    # Try-catch block
24
25
        result = 10 / 0
26
    catch e
27
        result = "Error: $e"
28
    end
29
30
    # While loop with break statement
32
    while i \leq 5
33
        if i = 3
34
            break
35
        end
        println(i)
37
        i += 1
38
    end
39
40
    # Anonymous function (lambda) and map function
41
    double = x \rightarrow 2x
42
    numbers = [1, 2, 3, 4, 5]
43
    doubled_numbers = map(double, numbers)
44
45
    # Set data structure
46
    fruits = Set(["apple", "banana", Nothing])
47
    # Tuple unpacking and multi-line string
49
    (x, y) = (10, 20)
50
    multi_line_string = """
51
    This is a multi-line
52
    string in Julia.
53
    0.000
```

Listing 1: Example - Julia (minted powered highlighting)

```
from math import cos
1
   # Class with magic methods and inheritance
2
   class Shape:
3
4
5
        This is doc comment for Shape with some escape sequence \t\t in it.
        Second line here.
6
7
        def __init__(self, size: int, color: str = "black"):
8
            self.color = color
9
            self.size = abs(size)
10
            print(f"creating {self.color} shape of size: ", self.size)
11
            if self.size < 5:</pre>
12
                print("shape is quite small")
13
14
        def \_str\_(self) \rightarrow str:
15
            return f"{self.color} shape" # here some direct string formatting
16
17
        def truthy(self, a: int) \rightarrow bool:
18
            print(f"foo\t{a}\tbar") # here some more string shenanigans
19
            return True
20
21
        def is_larger_than(self, other: "Shape") → bool:
22
            # yes, this could be single line
23
            if self.size > other.size:
24
25
                return True
            else:
26
                return False
27
28
   def foo(i: int = 5) \rightarrow None:
29
30
        Here is some random function with comment string
31
32
        SOME_CONSTANT = 42 # constants are recognized too!
33
        print(f"the constant is {SOME_CONSTANT}.", end="\n\n")
34
        print("input i: ", i)
35
        return None
36
37
   foo(8) # actually call the function
38
   circle = Shape(size=3, color="red") # call some constructors
39
   square = Shape(size=5)
40
41
   if circle.is_larger_than(square):
        print(f"the {circle.color} circle is larger than {square.color} square")
42
   else:
43
        print(f"the {circle.color} circle is smaller than {square.color} square")
44
```

Listing 2: Example - Python (TreeSitter powered highlighting)

```
from math import cos
    # Class with magic methods and inheritance
    class Shape:
3
        0.00
4
        This is doc comment for Shape with some escape sequence \t\t in it.
        Second line here.
6
        def __init__(self, size: int, color: str = "black"):
            self.color = color
            self.size = abs(size)
10
            print(f"creating {self.color} shape of size: ", self.size)
11
             if self.size < 5:</pre>
12
                 print("shape is quite small")
14
        def _str_(self) \rightarrow str:
15
             return f"{self.color} shape" # here some direct string formatting
17
        def truthy(self, a: int) \rightarrow bool:
18
             print(f"foo\t{a}\tbar") # here some more string shenanigans
19
             return True
20
21
        def is_larger_than(self, other: "Shape") → bool:
22
            # yes, this could be single line
23
            if self.size > other.size:
24
                 return True
25
            else:
26
                 return False
27
28
    def foo(i: int = 5) \rightarrow None:
29
30
        Here is some random function with comment string
31
32
        SOME_CONSTANT = 42 # constants are recognized too!
33
        print(f"the constant is {SOME_CONSTANT}.", end="\n\n")
34
        print("input i: ", i)
35
        return None
36
37
    foo(8) # actually call the function
38
    circle = Shape(size=3, color="red") # call some constructors
39
    square = Shape(size=5)
40
    if circle.is_larger_than(square):
41
        print(f"the {circle.color} circle is larger than {square.color} square")
42
    else:
43
        print(f"the {circle.color} circle is smaller than {square.color} square")
44
```

Listing 2: Example - Python (minted powered highlighting)

```
use std::collections::HashMap;
1
    use std::ops::{Add, Mul};
2
3
    #[derive(Debug)]
4
5
    struct Point<T> {
        x: T,
6
7
        y: T,
8
9
    impl<T> Point<T> {
10
        fn new(x: T, y: T) \rightarrow Self {
11
            Point { x, y }
12
13
14
15
    #[derive(Debug)]
16
17
    enum Shape {
        Circle(f64),
18
        Rectangle(f64, f64),
19
        Triangle(f64, f64, f64),
20
21
22
    fn area(shape: Shape) \rightarrow f64 {
23
        match shape {
24
25
            Shape::Circle(radius) \Rightarrow std::f64::consts::PI * radius.powi(2),
            Shape::Rectangle(width, height) ⇒ width * height,
26
            Shape::Triangle(a, b, c) \Rightarrow {
27
                 let s = (a + b + c) / 2.0;
28
                 (s * (s - a) * (s - b) * (s - c)).sqrt()
29
30
        }
31
    }
32
33
    fn main() {
34
        let mut scores = HashMap::new();
35
        scores.insert("Alice", 95);
36
        scores.insert("Charlie", 85);
37
        println!("Scores: {:?}", scores); // here is some comment
38
39
40
        let point = Point::new(10, 20);
41
        println!("Point: {:?}", point);
42
        let circle = Shape::Circle(5.0);
43
        let rectangle = Shape::Rectangle(4.0, 6.0);
44
        let triangle = Shape::Triangle(3.0, 4.0, 5.0);
45
46
        println!("Circle area: {}", area(circle));
47
        println!("Rectangle area: {}", area(rectangle));
48
49
        let numbers = [1, 2, 3, 4, 5];
50
        println!("Sum: {}", numbers.iter().sum::<i32>());
51
    }
52
```

Listing 3: Example - Rust (TreeSitter powered highlighting)

```
use std::collections::HashMap;
    use std::ops::{Add, Mul};
2
3
    #[derive(Debug)]
4
    struct Point<T> {
        x: T,
6
        y: T,
    }
    impl<T> Point<T> {
10
        fn new(x: T, y: T) \rightarrow Self {
11
             Point { x, y }
12
        }
13
    }
14
15
    #[derive(Debug)]
16
    enum Shape {
17
        Circle(f64),
18
        Rectangle(f64, f64),
19
        Triangle(f64, f64, f64),
20
    }
21
22
    fn area(shape: Shape) \rightarrow f64 {
23
        match shape {
24
             Shape::Circle(radius) ⇒ std::f64::consts::PI * radius.powi(2),
25
             Shape::Rectangle(width, height) ⇒ width * height,
26
             Shape::Triangle(a, b, c) \Rightarrow {
27
                 let s = (a + b + c) / 2.0;
28
                 (s * (s - a) * (s - b) * (s - c)).sqrt()
29
             }
30
        }
31
    }
32
33
    fn main() {
34
        let mut scores = HashMap::new();
35
        scores.insert("Alice", 95);
36
        scores.insert("Charlie", 85);
37
        println!("Scores: {:?}", scores); // here is some comment
38
39
        let point = Point::new(10, 20);
40
        println!("Point: {:?}", point);
41
42
        let circle = Shape::Circle(5.0);
43
        let rectangle = Shape::Rectangle(4.0, 6.0);
44
        let triangle = Shape::Triangle(3.0, 4.0, 5.0);
45
46
        println!("Circle area: {}", area(circle));
47
        println!("Rectangle area: {}", area(rectangle));
49
        let numbers = [1, 2, 3, 4, 5];
50
        println!("Sum: {}", numbers.iter().sum::<i32>());
51
    }
52
53
```

Listing 3: Example - Rust (minted powered highlighting)