

```

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

```

Listing 1: Example - Julia (TreeSitter powered highlighting)

```

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

```

Listing 1: Example - Julia (minted powered highlighting)

```

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

```

Listing 2: Example - Python (TreeSitter powered highlighting)

```

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

```

Listing 2: Example - Python (`minted` powered highlighting)

```

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

Listing 3: Example - Rust (TreeSitter powered highlighting)

```

1  use std::collections::HashMap;
2  use std::ops::{Add, Mul};
3
4  #[derive(Debug)]
5  struct Point<T> {
6      x: T,
7      y: T,
8  }
9
10 impl<T> Point<T> {
11     fn new(x: T, y: T) -> Self {
12         Point { x, y }
13     }
14 }
15
16 #[derive(Debug)]
17 enum Shape {
18     Circle(f64),
19     Rectangle(f64, f64),
20     Triangle(f64, f64, f64),
21 }
22
23 fn area(shape: Shape) -> f64 {
24     match shape {
25         Shape::Circle(radius) => std::f64::consts::PI * radius.powi(2),
26         Shape::Rectangle(width, height) => width * height,
27         Shape::Triangle(a, b, c) => {
28             let s = (a + b + c) / 2.0;
29             (s * (s - a) * (s - b) * (s - c)).sqrt()
30         }
31     }
32 }
33
34 fn main() {
35     let mut scores = HashMap::new();
36     scores.insert("Alice", 95);
37     scores.insert("Charlie", 85);
38     println!("Scores: {:?}", scores); // here is some comment
39
40     let point = Point::new(10, 20);
41     println!("Point: {:?}", point);
42
43     let circle = Shape::Circle(5.0);
44     let rectangle = Shape::Rectangle(4.0, 6.0);
45     let triangle = Shape::Triangle(3.0, 4.0, 5.0);
46
47     println!("Circle area: {}", area(circle));
48     println!("Rectangle area: {}", area(rectangle));
49
50     let numbers = [1, 2, 3, 4, 5];
51     println!("Sum: {}", numbers.iter().sum::<i32>());
52 }
53

```

Listing 3: Example - Rust (minted powered highlighting)