

FOUNDATION



Function as output

Functions as output

```
def format(scale: Int, number: Double): String
```

Scale = 8



1234.12345678

Functions as output

```
def format(scale: Int, number: Double): String
```

```
format(2, 1234.12345678)  
// res0: String = "1234.12"
```

```
format(5, 1234.12345678)  
// res1: String = "1234.12345"
```

Scale = 8

←-----→

1234.12345678

Functions as output

```
def format(scale: Int, number: Double): String =  
  ...  
  
def format2D(number: Double): String = format(2, number)  
def format5D(number: Double): String = format(5, number)
```

```
format2D(1234.12345678)  
// res2: String = "1234.12"  
  
format5D(1234.12345678)  
// res3: String = "1234.12345"
```

Functions as output

```
def publishEvent(  
  hostname: String,  
  port      : Int,  
  topic     : String,  
  key       : String,  
  event     : Event  
): Unit
```

Functions as output

```
def publishEvent(  
  hostname: String,  
  port      : Int,  
  topic     : String,  
  key       : String,  
  event     : Event  
): Unit
```

```
def publishLocalEvent(topic: String, key: String, event: Event): Unit =  
  publishEvent("localhost", 9000, topic, key, event)
```

```
def publishUatEvent(topic: String, key: String, event: Event): Unit =  
  publishEvent("uat-acme.com", 12450, topic, key, event)
```

Functions as output

```
def publishEvent(  
  hostname: String,  
  port      : Int,  
  topic     : String,  
  key       : String,  
  event     : Event  
): Unit
```

```
def publishLocalEvent = publishEvent("localhost", 9000)  
def publishUatEvent   = publishEvent("uat-acme.com", 12450)
```

Pseudocode

Functions as output

```
def format(scale: Int, number: Double): String
```

```
format(2, 1234.12345678)  
// res5: String = "1234.12"
```

```
def format(scale: Int): Double => String
```

```
format(2)(1234.12345678)  
// res7: String = "1234.12"
```

Functions as output

```
def format(scale: Int, number: Double): String
```

```
format(2, 1234.12345678)  
// res5: String = "1234.12"
```

```
def format(scale: Int): Double => String
```

```
format(2)(1234.12345678)  
// res7: String = "1234.12"
```

Currying

```
val function3: (Int , Int , Int) => Int
```

```
val function3: Int => (Int => (Int => Int))
```

Functions as output

```
def format(scale: Int, number: Double): String
```

```
format(2, 1234.12345678)  
// res9: String = "1234.12"
```

```
def format(scale: Int): Double => String
```

```
format(2)(1234.12345678)  
// res11: String = "1234.12"
```

Currying

```
val function3: (Int , Int , Int) => Int
```

```
val function3: Int => Int => Int => Int
```

Partial function application

```
def format(scale: Int): Double => String
```

```
val format2D = format(2)  
val format5D = format(5)
```

Partial function application

```
def format(scale: Int): Double => String
```

```
val format2D = format(2)  
val format5D = format(5)
```

```
format2D(1234.12345678)  
// res12: String = "1234.12"  
  
format5D(1234.12345678)  
// res13: String = "1234.12345"
```

Syntax

Uncurried

```
def format(scale: Int, number: Double): String
```

Curried

```
def format(scale: Int)(number: Double): String
```

```
def format(scale: Int): Double => String
```

```
val format: Int => Double => String
```

Conversion (Currying)

```
def format(scale: Int, number: Double): String
```

Conversion (Currying)

```
def format(scale: Int, number: Double): String
```

```
format _  
// res15: (Int, Double) => String = <function2>
```


Conversion (Currying)

```
def format(scale: Int, number: Double): String
```

```
format _  
// res15: (Int, Double) => String = <function2>
```

```
(format _).curried  
// res16: Int => Double => String = scala.Function2$$Lambda$7196/0x00000000102221040@27721800
```

Summary

- A curried function can be partially applied
- Only works from left to right arguments
- One more thing ...

Exercise 2: Functions as output

`exercises.function.FunctionExercises.scala`