Currying

lambda calculus ML Haskell currying Haskell

```
f x y = x + y
```

2

```
map (f 2) [1, 2, 3]
```

```
[3, 4, 5]
```

`f (f 2) f` Haskell “currying”Currying “” `f` Scheme

```
(define f
  (lambda (x)
    (lambda (y)
      (+ x y))))
```

`f x y x + y(f 2)` 2 map `[1, 2, 3] [3, 4, 5]`

currying “” currying

```
map (\y->f 2 y) [1, 2, 3]
```

Haskell ML currying lambda calculus lambda calculus Haskell Curry

Haskell Curry currying currying

currying `(\y->f 2 y)` currying `(f 2)`“\y->f 2 y”””

“f 2 [1, 2, 3] f y f ”

`(f 2)` “”

`(f 2) \y->f 2 y (f 2)` “f ” (f 2) f 2345..... 3`map (f 2) [1, 2, 3] [(\z->f 2 1 z), (\z->f 2 2 z), (\z->f 2 3 z)]`

`(f 2)` “” `f` “”

currying “” curry f

```
f x y = x / y
```

`[1, 2, 3]` 2

`map (f 2) [1, 2, 3]` 2 Haskell

```
map (flip f 2) [1, 2, 3]
```

`flip` “”

```
flip f x y = f y x
```

f 3 2 `map f`

```
f x y z = (x - y) / z
```

```
map (flip (f 1) 2) [1, 2, 3]
```

“”””””

```
map (\y -> f 1 y 2) [1, 2, 3]
```

`(flip (f 1) 2)`

`(flip (f 1) 2) \y-> f 1 y 2` 1 2 `f` map `[1, 2, 3]`

currying

currying `flip` f map `[1, 2, 3]`

```
map (\y -> f y 1 2) [1, 2, 3]
```

“” `map` currying “”

currying currying currying ..... currying

ML Haskell currying