Scoping Rules

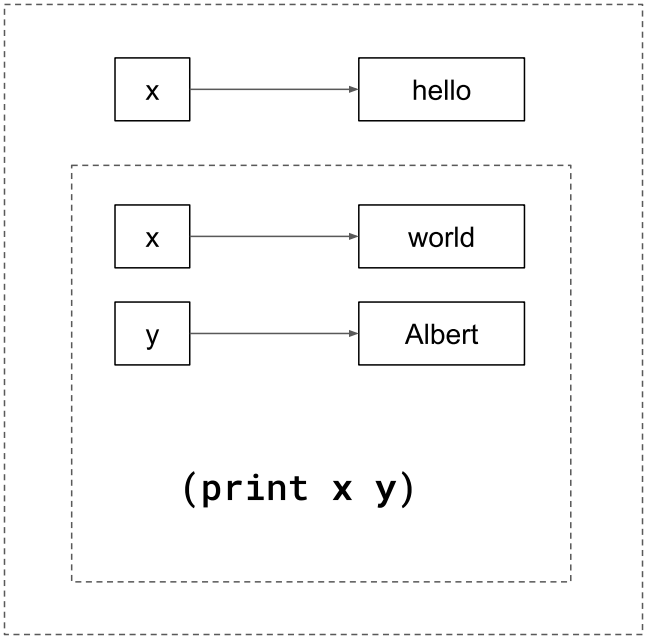
How do we resolve bindings for nested scopes

Nested Scopes

```
(def x "hello")

(let [x "world"
      y "Albert"]
  (print x y))
```

What is the binding for **x**?

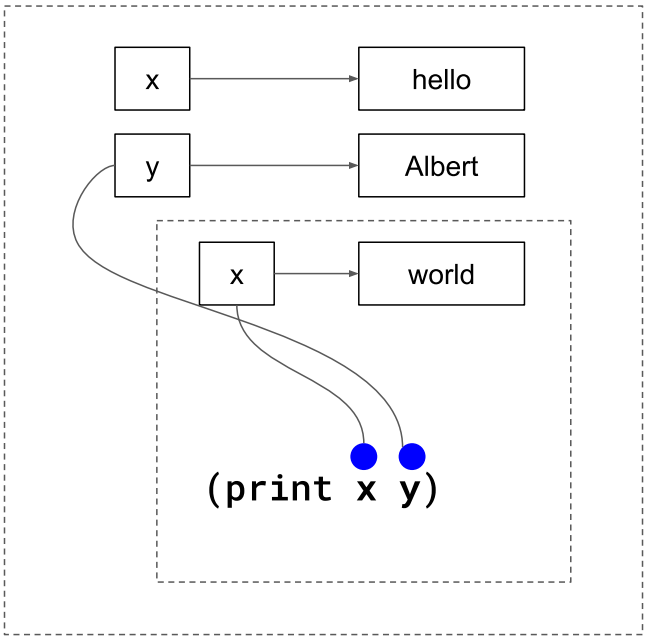


Nested Scopes

```
(def x "hello")
(def y "Albert")

(let [x "world"]
  (print x y))
```

What is the binding for **x**?

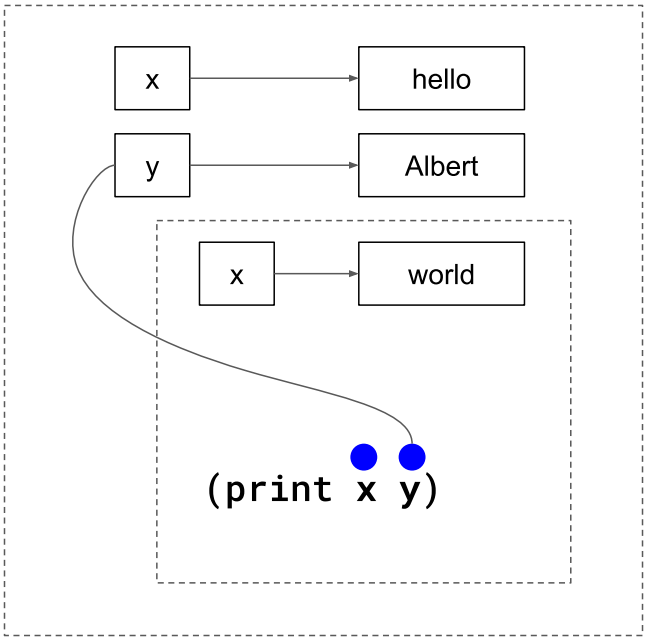


Nested Scopes

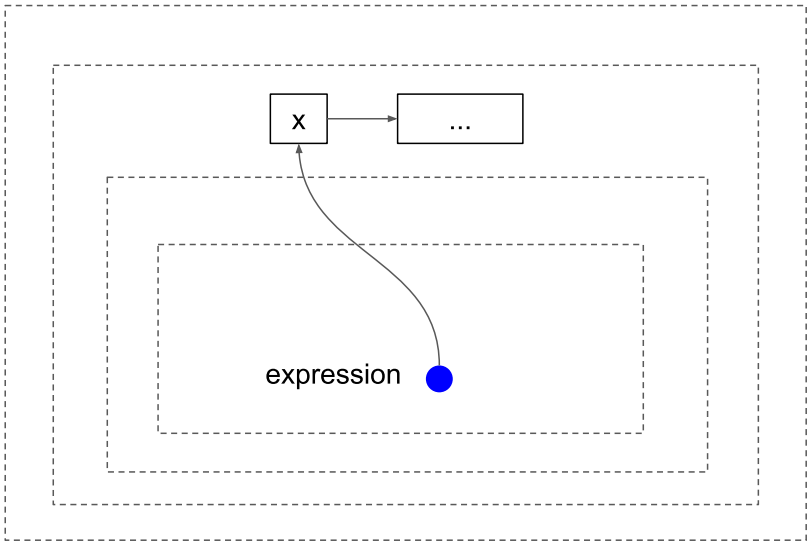
```
(def x "hello")
(def y "Albert")

(let [x "world"]
  (print x y))
```

What is the binding for **y**?



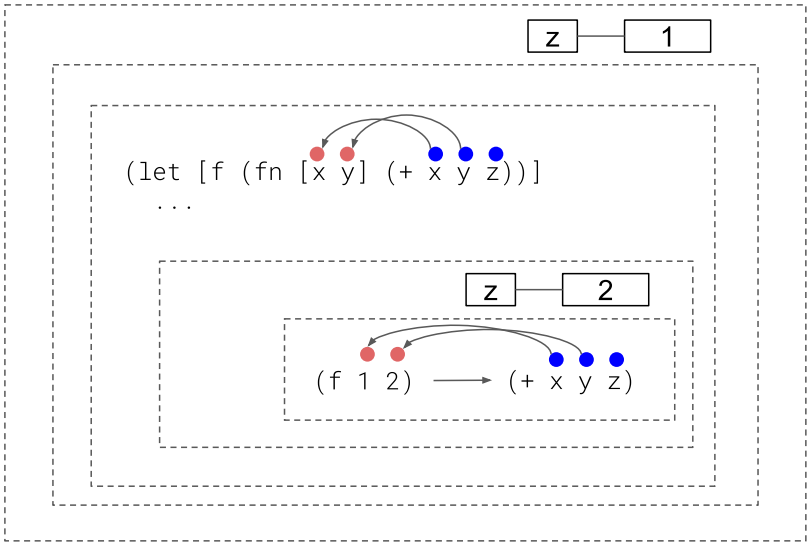
Closure of scopes



Functions as values

Functions are *weird*.

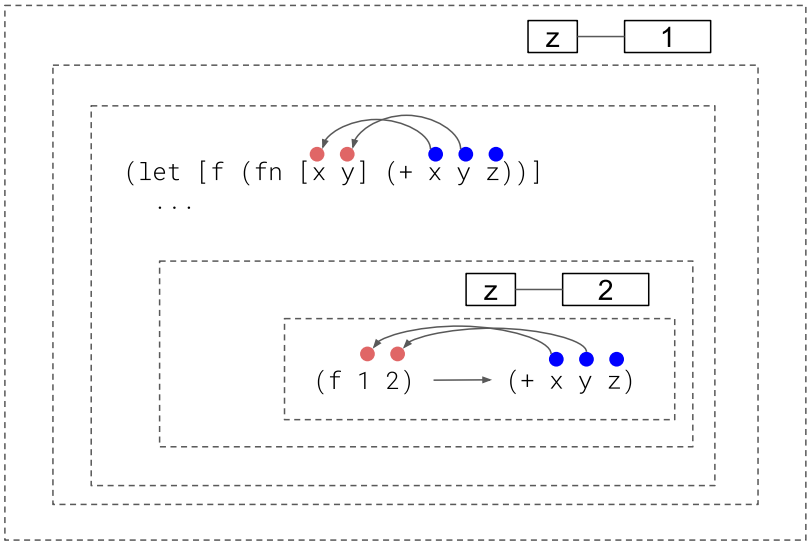
- Functions are **declared**.
- Functions are **invoked**.
- Functions can contain **free variables.**



Functions as values

What is the binding for **z**?

1. Closure of scope of function declaration.
2. Closure of scope of function invocation.

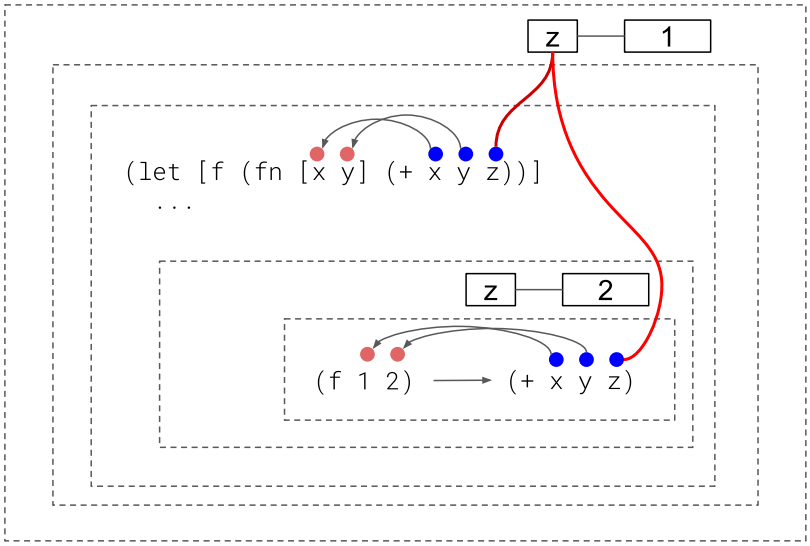


Functions as values

What is the binding for **z**?

- Closure of scope of function declaration.

Lexical scoping



Functions as values

What is the binding for **z**?

- Closure of scope of function invocation.

Dynamic scoping