# Function Definitions

Functions are not only about organizing code or making it easier to reuse code, but provide a powerful means of **abstraction**!

# Iffy Lang

$$b ::= \mathsf{T} \mid \mathsf{F} \mid \mathsf{if}\ b_1\ \mathsf{then}\ b_2\ \mathsf{else}\ b_3$$

# Iffy Lang

$p$ ::= func $name(x_1, \ldots, x_i)\{body\}$

$body$ ::= return $b$

$b$ ::= $x$ | $name(b_1, \ldots, b_i)$ | T | F | if $b_1$ then $b_2$ else $b_3$

# Example Program: Negation

```
func not(x) {
  ?
}
```

# Example Program: Negation

```
func not(x) {
  return (if x
          then F
          else T)
}
```

# Example Program: And

```
func and(x1, x2) {
  ?
}
```

# Example Program: And

```
func and(x1, x2) {
  return (if x1
          then x2
          else F)
}
```

# Example Program: And

```
func or(x1, x2) {
  ?
}
```

# Example Program: And

```
func or(x1, x2) {
  return (if x1
          then T
          else x2)
}
```

# Example Program: Implication

```
func implies(x1, x2) {
  ?
}
```

# Example Program: Implication

```
func implies(x1, x2) {
  return or(not(x1),x2)
}
```

# Example Program: Necessary and Sufficient

```
func iff(x1, x2) {
  ?
}
```

# Example Program: Necessary and Sufficient

```
func iff(x1, x2) {
  return and(implies(x1,x2),implies(x2,x1))
}
```

# Example Program: Exclusive-Or

```
func xor(x1, x2) {
  ?
}
```

# Example Program: Exclusive-Or

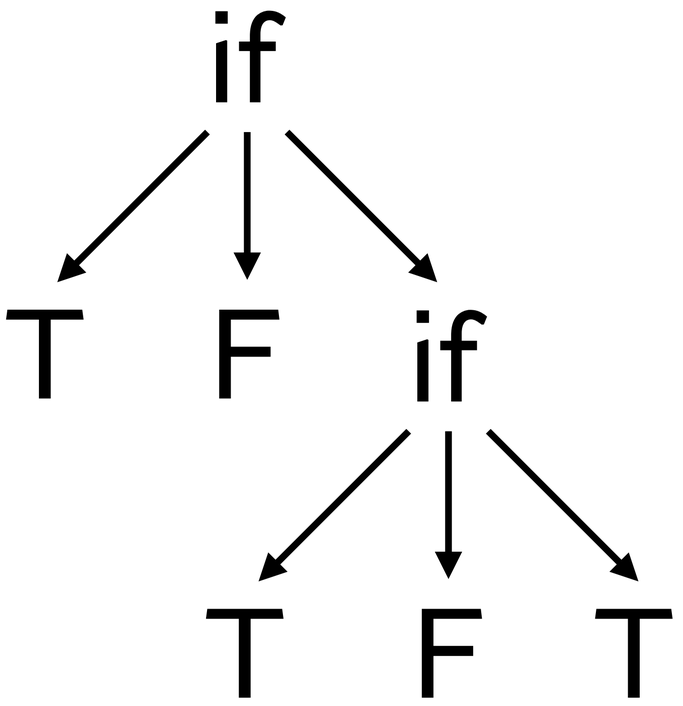
```
func xor(x1, x2) {
  return or(and(x1,not(x2)),and(not(x1),x2))
}
```

# Syntax Trees

**Syntax trees** are a tree representation of a syntactical expression.

# Example Syntax Tree

if F then T else (if F then T else T)

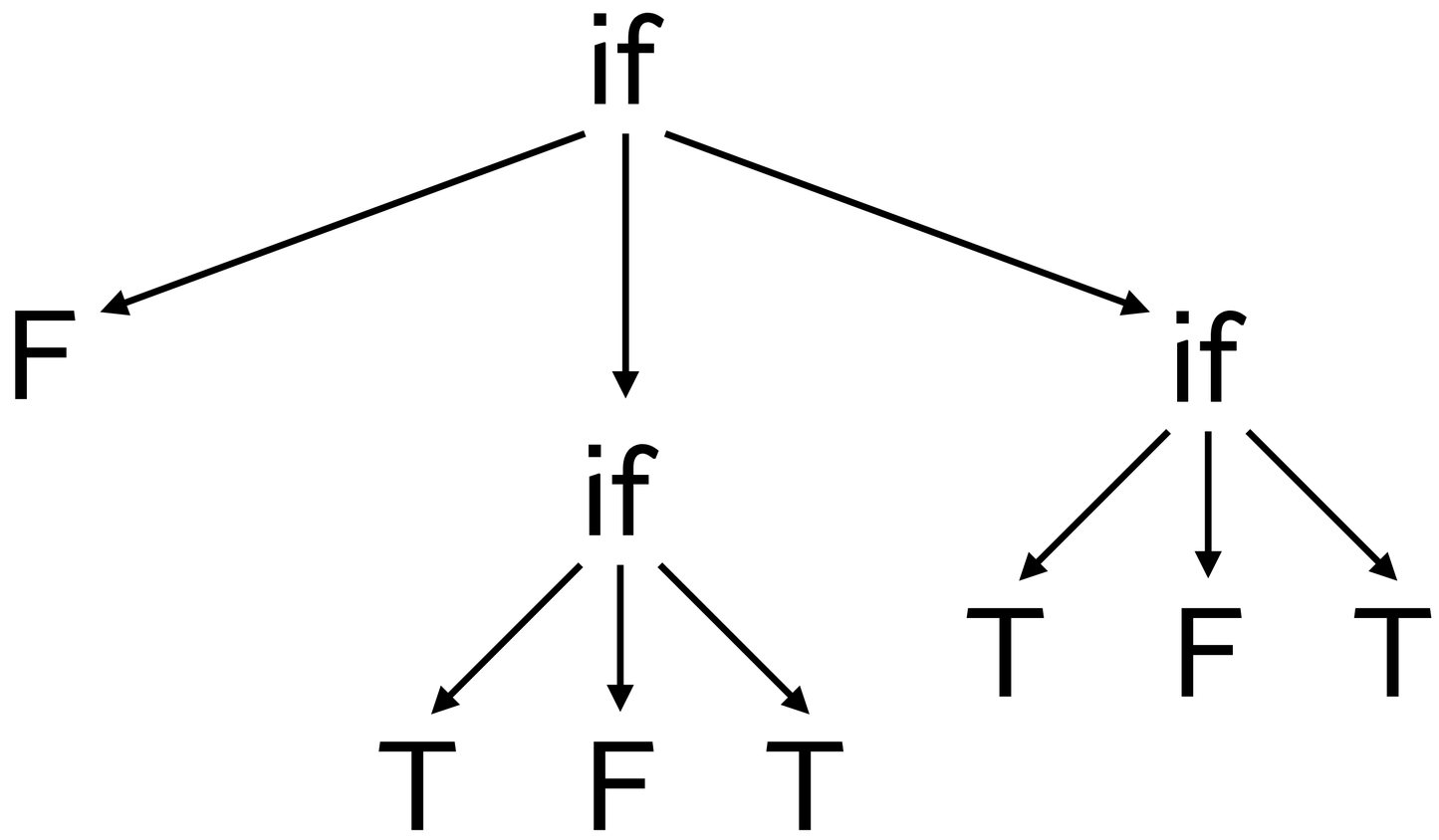


# Example Syntax Tree

if (if F then T else T) then F else (if F then T else T)

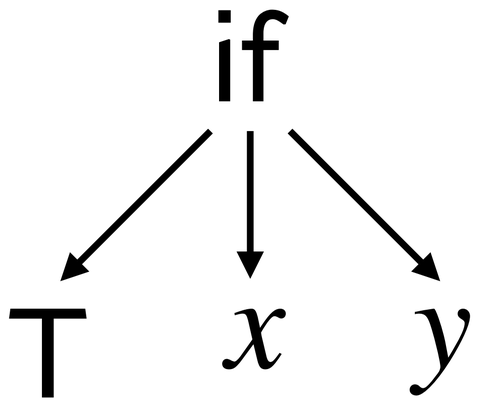
# Example Syntax Tree

if (if F then T else T) then F else (if F then T else T)



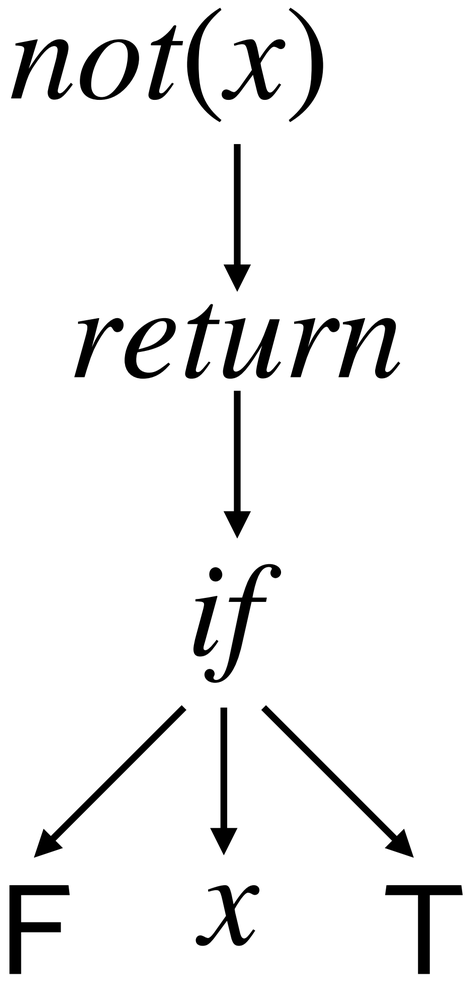
# Example Syntax Tree

if $x$ then T else $y$



# Syntax Trees for Function Definitions

```
func not(x) {
  return (if x
            then F
            else T)
}
```

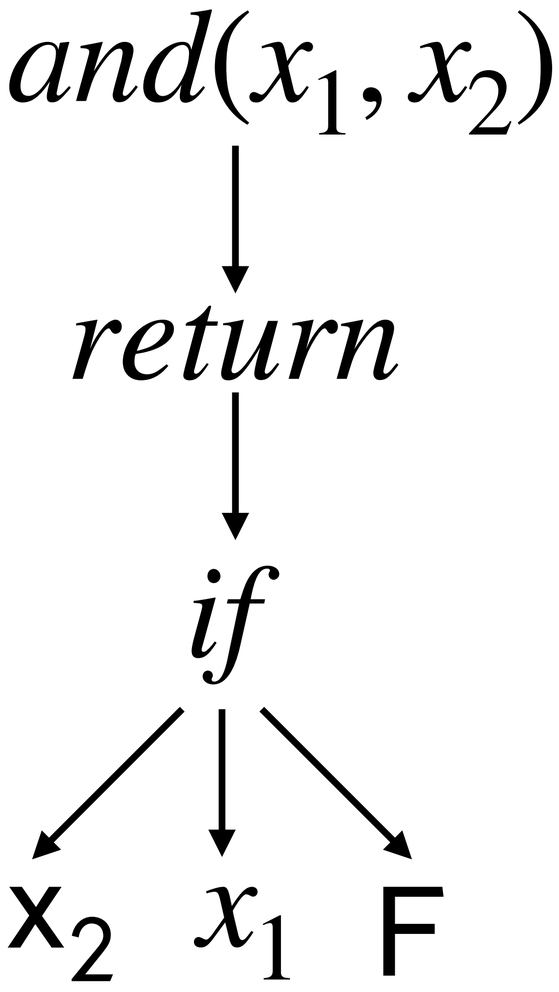


# Syntax Trees for Function Definitions

```
func and(x1, x2) {
  return (if x1
          then x2
          else F)
}
```

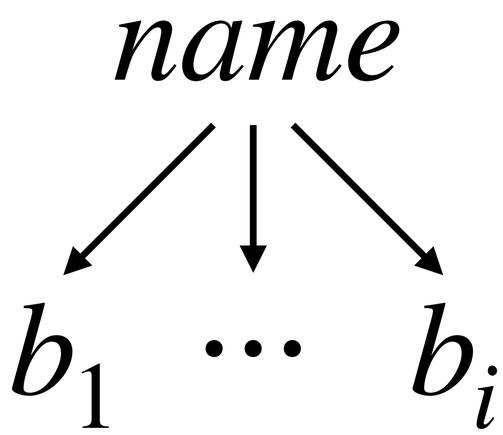
# Syntax Trees for Function Definitions

```
func and(x1, x2) {
  return (if x1
            then x2
            else F)
}
```



# Syntax Trees for Function Application

$name(b_1, \ldots, b_i)$



# Syntax Trees for Function Application

```
func implies(x1, x2) {
  return or(not(x1),x2)
}
```