```rust
fn example_1() {
    let r;

    {
        let x = 42;
        r = &x;
    }

    println!("{}", *r);
}
```

```
fn example_1() {
    let r;

    {
        let x = 42;     🟩 binding `x` declared here
        r = &x;     🟥 `x` does not live long enough
    }     🟩 `x` dropped here while still borrowed

    println!("{}", *r);     🟩 borrow later used here
}
```

```rust
fn example_1() {            //
    let r;                  // ----------+--- 'r
                            //           |
    {                       //           |
        let x = 42;         // --+-- 'x  |
        r = &x;             //   |       |
    }                       // --+       |
                            //           |
    println!("{}", *r);     //           |
}                           // ----------+
fn example_1() {
    let r;

    {
        let x = 42;
        r = &x;
    }

}
```

```
fn example_1() {            //
    let r;                  // ----------+-- 'r
                            //           |
    {                       //           |
        let x = 42;         // --+-- 'x  |
        r = &x;             //   |       |
    }                       // --+       |
                            //           |
}                           // ----------+
```

```rust
fn example_1() {            //
    let r;                  // ----------+--- 'r
                            //           |
    {                       //           |
        let x = 42;         // --+--- 'x |
        r = &x;             //   |       |
    }                       // --+       |
                            //           |
}                           // ----------+
```

Lifetimes are named regions of code that a reference must be valid for.

Rustonomicon 3.3

```
fn example_1() {            //
    let r;                  // ----------+-- 'r
                            //           |
    {                       //           |
        let x = 42;         // --+-- 'x  |
        r = &x;             //   |       |
    }                       // --+       |
                            //           |
    println!("{}", *r);     //           |
}                           // ----------+
```

```rust
fn example_1() {            //
    let r;                  // -----------+-- 'r
                            //            |
    {                       //            |
        let x = 42;         //            |
        r = &x;             //            |
    }                       //            |
                            //            |
    println!("{}", *r);     //            |
}                           // -----------+
fn example_1() {            //
    let r;                  //
                            //
    {                       //
        let x = 42;         //
        r = &x;             // ----------+--- 'r
    }                       //           |
                            //           |
    println!("{}", *r);     //           |
}                           // ----------+
fn example_1() {            //
    let r;                  //
                            //
    {                       //
        let x = 42;         //
        r = &x;             // ----------+--  'r
    }                       //           |
                            //           |
    println!("{}", *r);     // ----------+
}                           //
```

```
fn example_1() {                  //
    let r;                        //
                                  //
    {                             //
        let x = 42;               //
        r = &x;                   // ------------- 'r
    }                             //
                                  //
}                                 //
```

```rust
fn example_2() {
    let foo = 69;
    let mut r;

    {
        let x = 42;
        r = &x;

        println!("{}", *r);
    }

    r = &foo;

    println!("{}", *r);
}
```

```rust
fn example_2() {                         //
    let foo = 69;                        //
    let mut r;                           //
                                         //
    {                                    //
        let x = 42;                      //
        r = &x;                          // ----------+-- 'r
                                         //           |
        println!("{}", *r);              // ----------+
    }                                    //
                                         //
    r = &foo;                            // ----------+-- 'r
                                         //           |
    println!("{}", *r);                  // ----------+
}                                        //
```

A variable is live if its current value may be used later in the program.

every compiler textbook ever

```rust
fn example_2() {                //
    let foo = 69;               //
    let mut r;                  //
                                //
    {                           //
        let x = 42;             //
        r = &x;                 // -----------+-- 'r
                                //            |
        println!("{}", *r);     // -----------+
    }                           //
                                //
    r = &foo;                   // -----------+-- 'r
                                //            |
    println!("{}", *r);         // -----------+
}                               //
```

```rust
fn example_2() {                //
    let foo = 69;               //
    let mut r;                  //
                                //
    {                           //
        let x = 42;             //
        r = &x;                 //  ----------+-- 'r = { &x }
                                //            |
        println!("{}", *r);     //  ----------+
    }                           //
                                //
    r = &foo;                   //  ----------+-- 'r = { &foo }
                                //            |
    println!("{}", *r);         //  ----------+
}                               //
```

```rust
fn example_2() {                            //
    let foo = 69;                           //
    let mut r;                              //
                                            //
    {                                       //
        let x = 42;                         //
        r = &x;                             // ----------+-- 'r = { &x }
                                            //           |
        println!("{}", *r);                 //           |
    }                                       //           |
                                            //           |
    println!("{}", *r);                     // ----------+
}                                           //
```

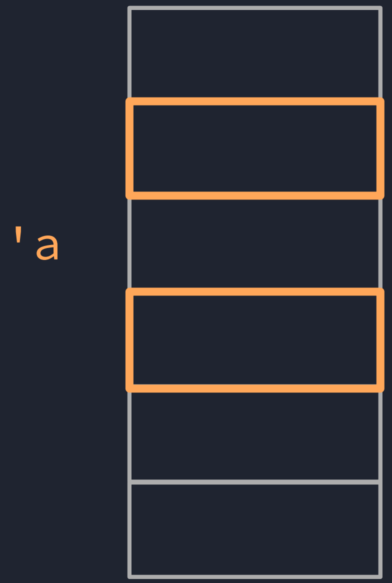
```rust
fn example_2() {                        //
    let foo = 69;                       //
    let mut r;                          //
                                        //
    {                                   //
        let x = 42;                     //
        r = &foo;                       // ----------+-- 'r = { &foo }
                                        //           |
        println!("{}", *r);             //           |
    }                                   //           |
                                        //           |
    println!("{}", *r);                 // ----------+
}                                       //
```

```rust
fn example_2() {                  //
    let foo = 69;                 //
    let mut r;                    //
                                  //
    {                             //
        let x = 42;               //
        r = &x;                   // ----------+-- 'r = { &x }
                                  //           |
        println!("{}", *r);       // ----------+
    }                             //
                                  //
    r = &foo;                     // ----------+-- 'r = { &foo }
                                  //           |
    println!("{}", *r);           // ----------+
}                                 //
```

```rust
fn example_2() {                //
    let foo = 69;               //
    let mut r;                  //
                                //
    {                           //
        let x = 42;             //
        r = &x;                 // -----------+-- 'r = { &x }
                                //            |
        println!("{}", *r);     //            |
    }                           //            |
                                //            |
    if random_bool() {          //            |
        r = &foo;               //            +-- 'r = { &x, &foo }
    }                           //            |
                                //            |
    println!("{}", *r);         // -----------+
}                               //
```

```rust
fn longest<'a>(
    s1: &'a str,
    s2: &'a str)
    -> &'a str
{
    if s1.len() > s2.len() {
        return s1;
    }
    else {
        return s2;
    }
}
```

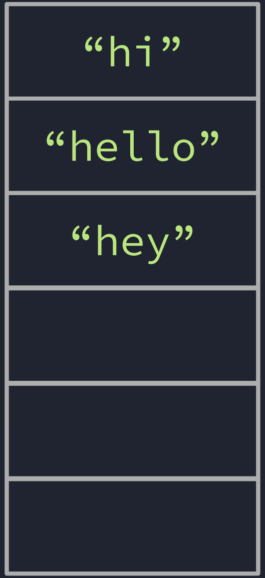
```rust
fn longest<'a>(
    s1: &'a str,
    s2: &'a str)
    -> &'a str
{
    if s1.len() > s2.len() {
        return s1;
    }
    else {
        return s2;
    }
}
```



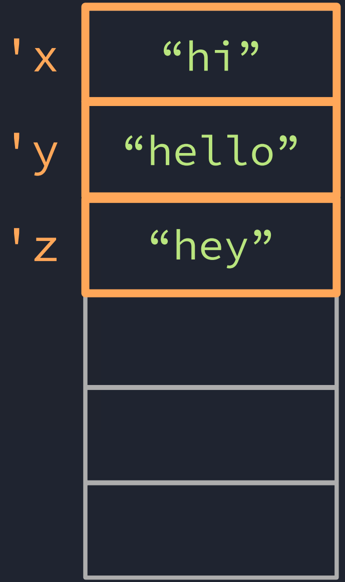
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x = "hi";
    let y = "hello";
    let z = "hey";

    let l1 = longest(x, y);
    let l2 = longest(l1, z);
}
```

```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```

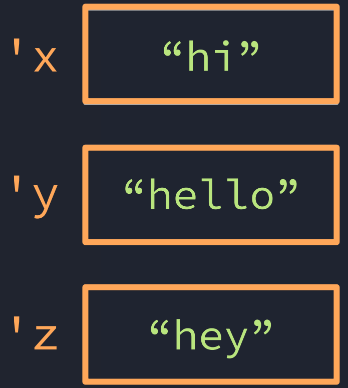
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```



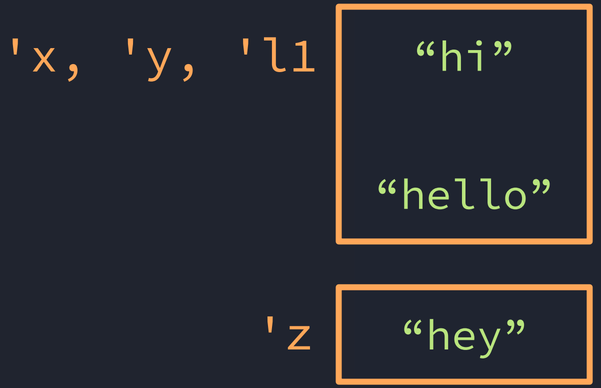
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
     let x: &'x str = "hi";
     let y: &'y str = "hello";
     let z: &'z str = "hey";

     let l1: &'l1 str = longest(x, y);
     let l2: &'l2 str = longest(l1, z);
}
```



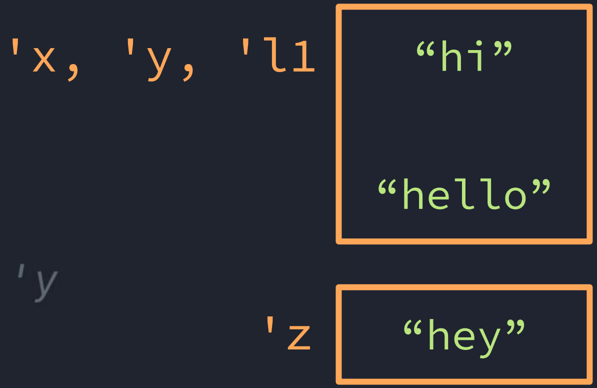
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```

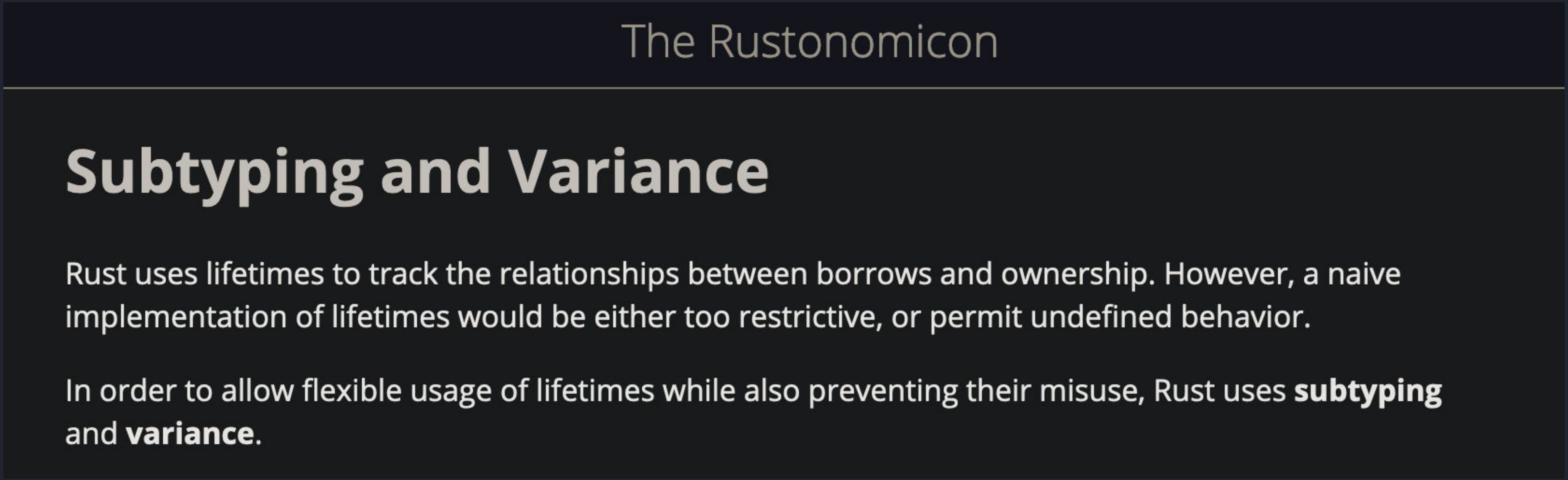


```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```



```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: String  = "hello".into();
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, &y); // 'y
    let l2: &'l2 str = longest(l1, z);
}
```




The Rustonomicon

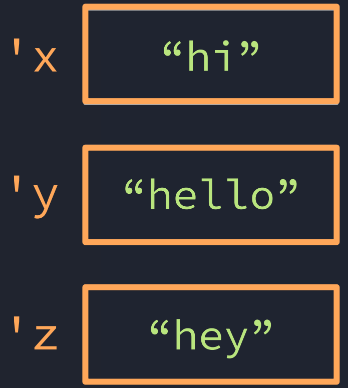
Subtyping and Variance

Rust uses lifetimes to track the relationships between borrows and ownership. However, a naive implementation of lifetimes would be either too restrictive, or permit undefined behavior.

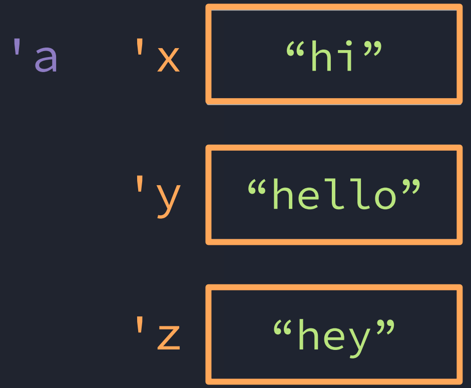
In order to allow flexible usage of lifetimes while also preventing their misuse, Rust uses **subtyping** and **variance**.

read more at: https://doc.rust-lang.org/nomicon/subtyping.html

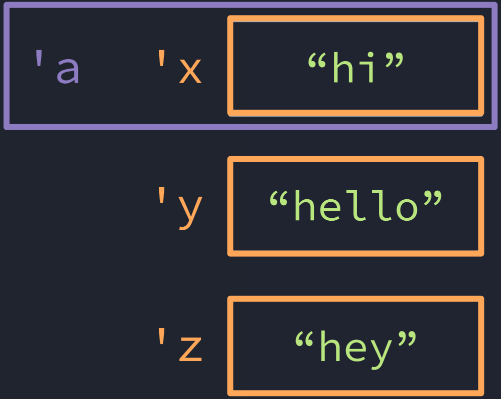
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```



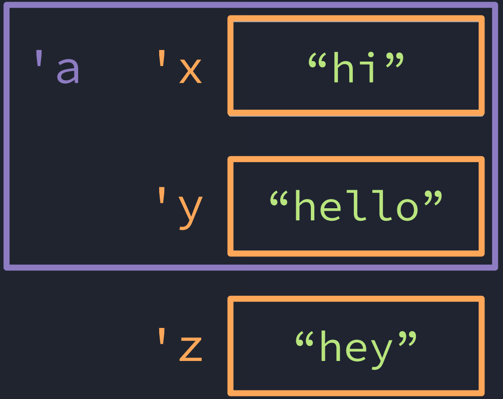
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```



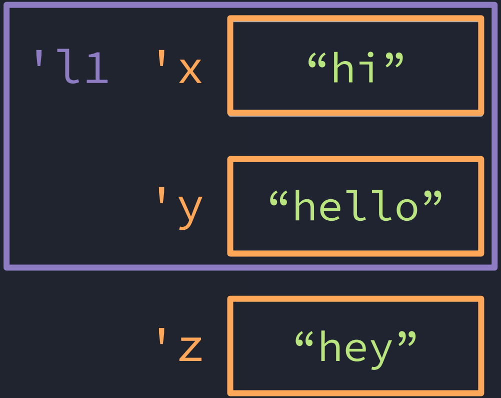
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```



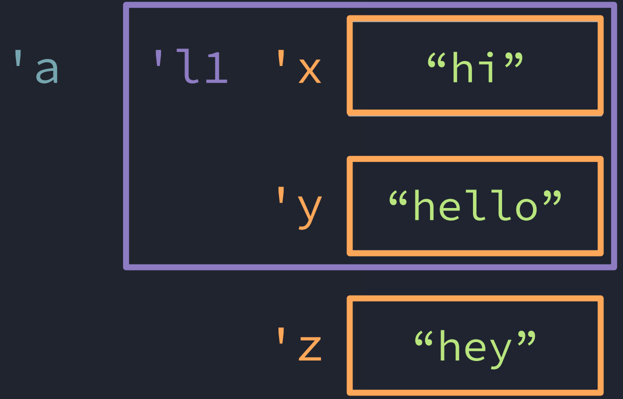
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```



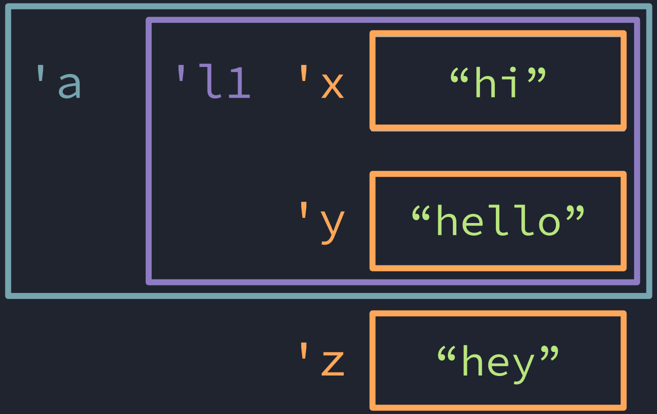
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```



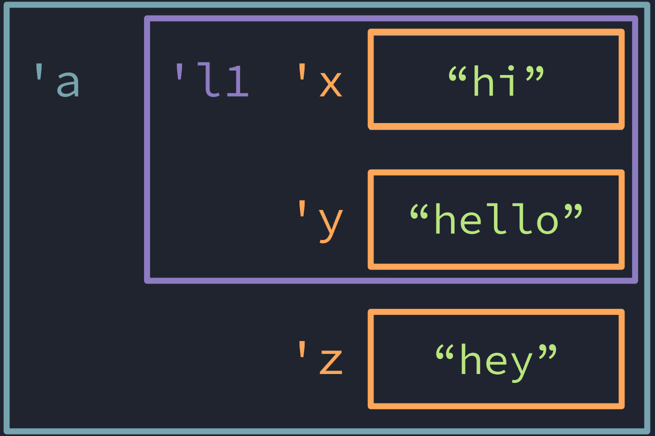
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```



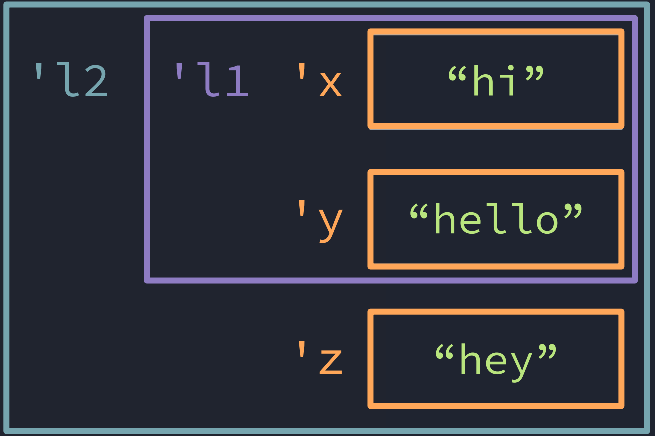
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```



```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```



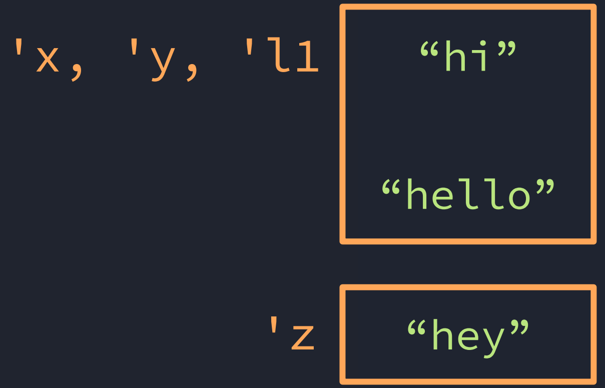
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```



```
'a: 'b
```

```rust
fn foo<'a, 'b>(a: &'a i32, b: &'b i32)
where 'a: 'b
```

```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {}


fn main() {
    let x: &'x str = "hi";
    let y: &'y str = "hello";
    let z: &'z str = "hey";

    let l1: &'l1 str = longest(x, y);
    let l2: &'l2 str = longest(l1, z);
}
```

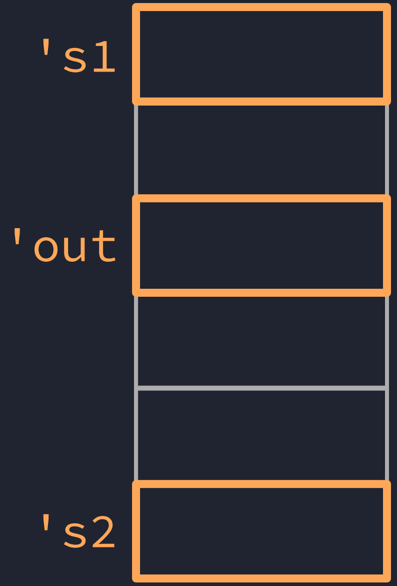


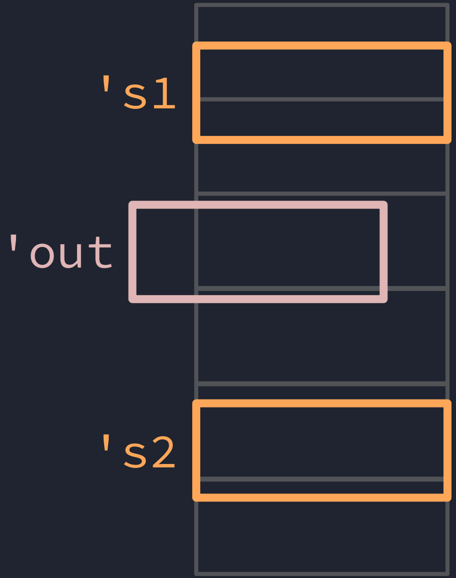
```rust
fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str {
    if s1.len() > s2.len() {
        return s1;
    }
    else {
        return s2;
    }
}
```

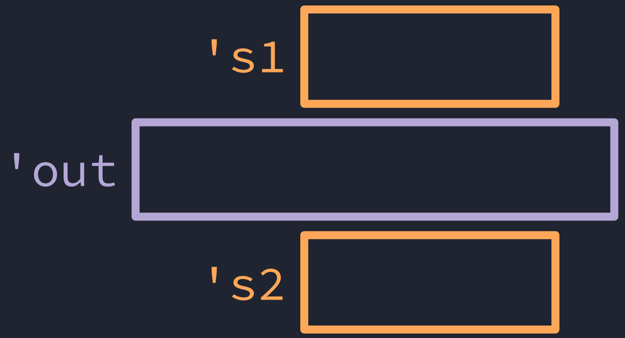
```rust
fn longest<'s1, 's2, 'out>(s1: &'s1 str, s2: &'s2 str) -> &'out str {
    if s1.len() > s2.len() {
        return s1;
    }
    else {
        return s2;
    }
}
```

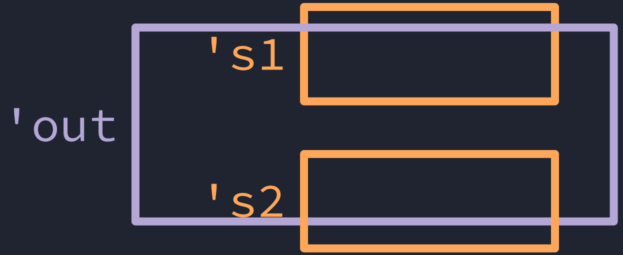
```rust
fn longest<'s1, 's2, 'out>(s1: &'s1 str, s2: &'s2 str) -> &'out str
    where 'x: 'y, ...
{
    if s1.len() > s2.len() {
        return s1;
    }
    else {
        return s2;
    }
}
```

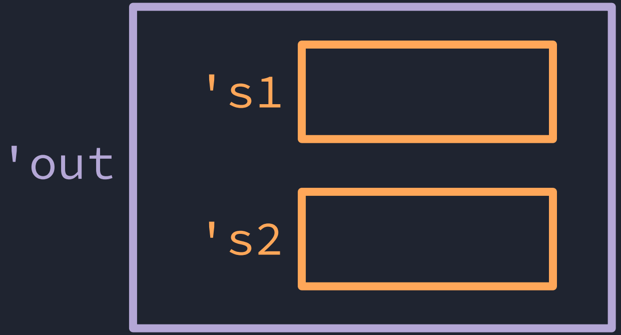
```
'a: 'b
```

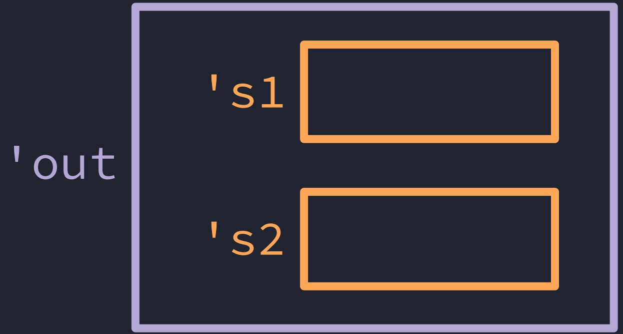

's1
'out
's2


's1
'out
's2


's1
'out
's2

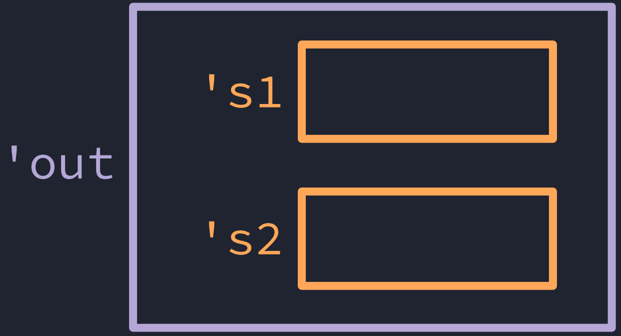

's1
'out
's2


'out
's1
's2



's1 ⊆ 'out

's2 ⊆ 'out



```
's1: 'out
's2: 'out
```

```
's1 ⊆ 'out

's2 ⊆ 'out
```

```
's1: 'out

's2: 'out
```

```rust
fn longest<'s1, 's2, 'out>(s1: &'s1 str, s2: &'s2 str) -> &'out str
    where 's1: 'out, 's2: 'out
{
    if s1.len() > s2.len() {
        return s1;
    }
    else {
        return s2;
    }
}
```

```rust
fn longest<'s1, 's2, 'out>(s1: &'s1 str, s2: &'s2 str) -> &'out str
    where 's1: 'out, 's2: 'out
```

```rust
fn longest<'s1, 's2, 'out>(s1: &'s1 str, s2: &'s2 str) -> &'out str
    where 's1: 'out, 's2: 'out

fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str
```

'a = 🤔