
yes23

We acknowledge, celebrate and pay our respects to the Ngunnawal and Ngambri people of the Canberra region and to all First Nations Australians on whose traditional lands we meet and work, and whose cultures are among the oldest continuing cultures in human history.

Read more:

- the Voice to Parliament,
- the Yes23 campaign,
- ANU's support for the Voice
- mis and disinformation

Anonymous Functions

Contents

Anonymous functions

(Thompson: §11.2)

- Function **without a name**
- Useful when we use higher order functions
- Related to the **lambda calculus** (the math behind Haskell)

Example: addOne

(Thompson: §11.2)

```
1  -- The function
2  addOne :: Int -> Int
3  addOne x = 1 + x
4
5  -- is the same as the anonymous function
6  \x -> 1 + x
```

In `ghci` we now have:

```
ghci> addOne 4
5
ghci> (\x -> 1 + x) 4
5
ghci> (\x -> 1 + x) 100
101
```

Multiple arguments

(Thompson: §11.2)

```
1  -- The function
2  add :: Int -> Int -> Int
3  add x y = x + y
4
5  -- is the same as the anonymous function
6  \x y -> x + y
```

In `ghci` we now have:

```
ghci> add 4 5
9
ghci> (\x y -> x + y) 4 5
9
ghci> (\x y -> x + y) 100 6
106
```

Try it yourself!

Write the following as an anonymous function

```
1  multPlusOne :: Int -> Int -> Int
2  multPlusOne x y = (x * y) + 1
```

```
3  \x y -> (x * y) + 1
```