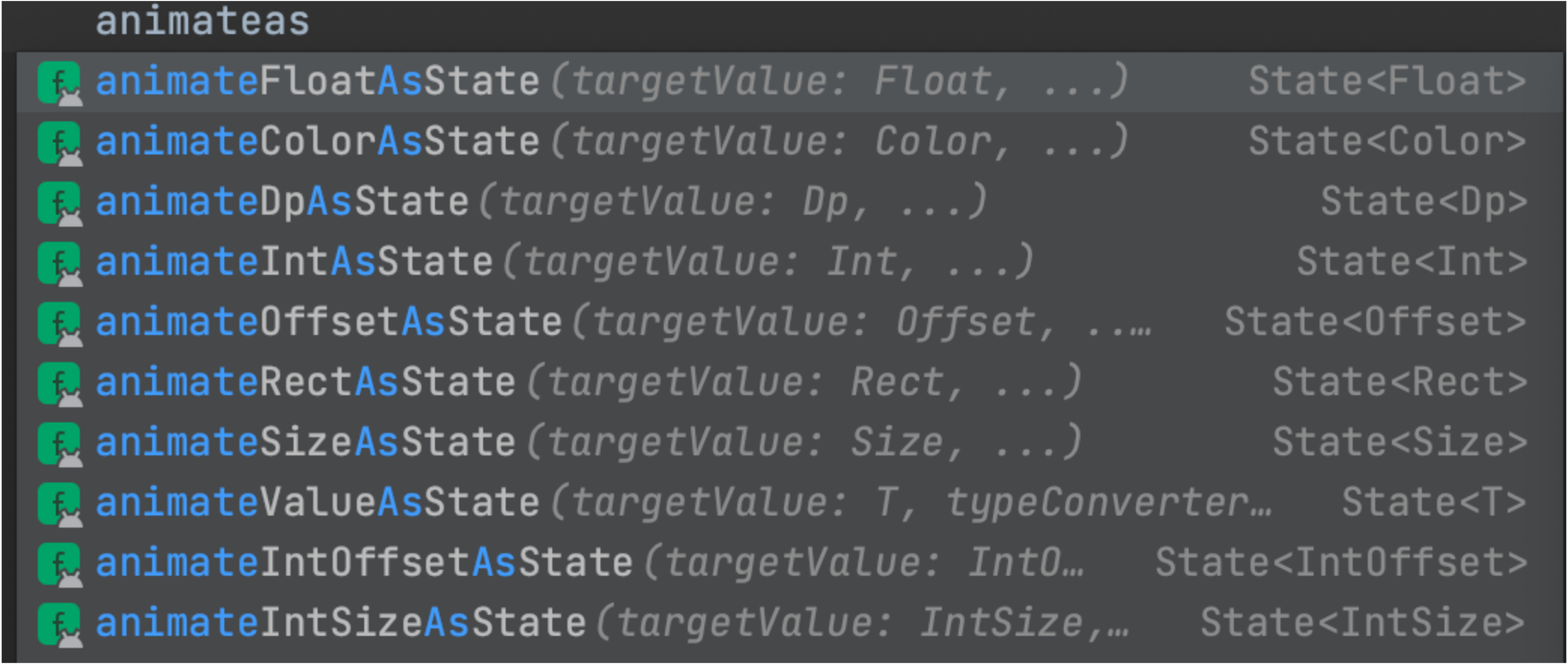
#AMxDC24

@ElizaCamber

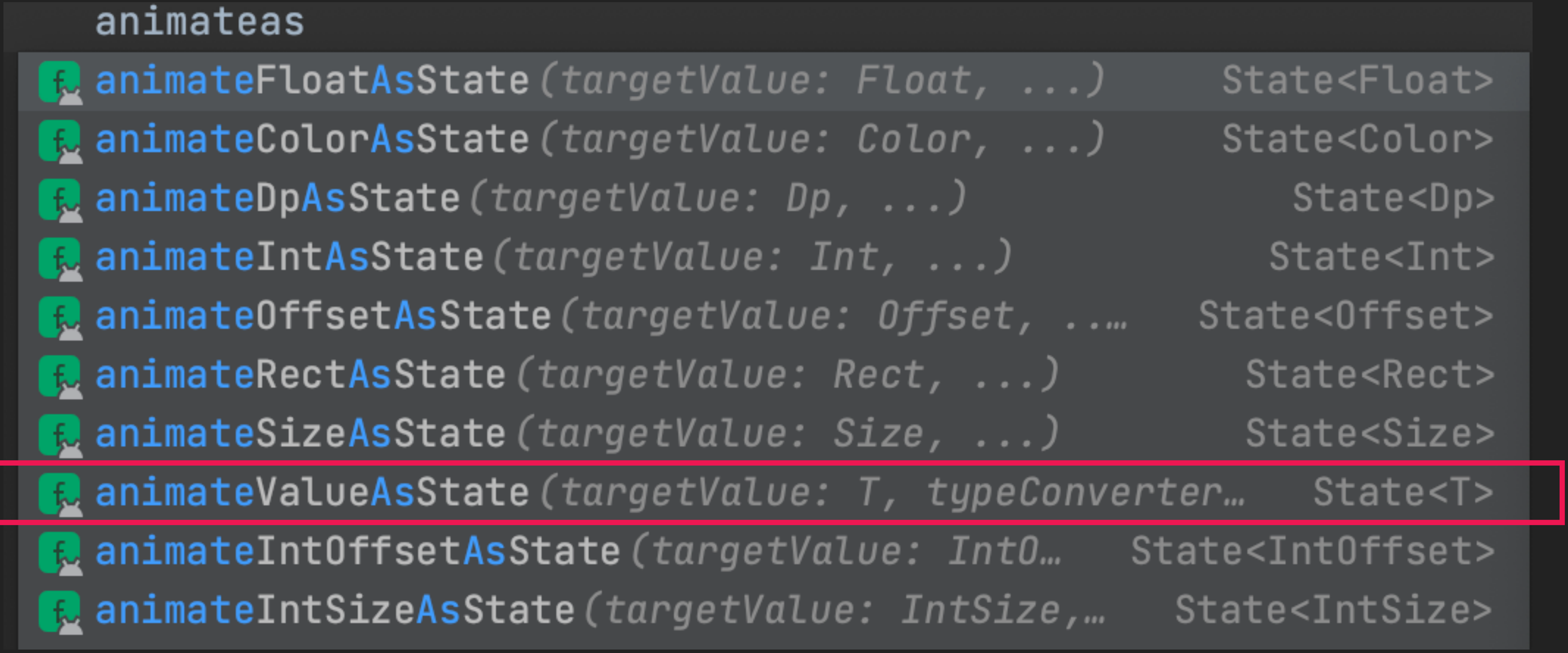
mkodo

animate*AsState



```
animateas
animateFloatAsState(targetValue: Float, ...)          State<Float>
animateColorAsState(targetValue: Color, ...)          State<Color>
animateDpAsState(targetValue: Dp, ...)                   State<Dp>
animateIntAsState(targetValue: Int, ...)                State<Int>
animateOffsetAsState(targetValue: Offset, ..…        State<Offset>
animateRectAsState(targetValue: Rect, ...)             State<Rect>
animateSizeAsState(targetValue: Size, ...)             State<Size>
animateValueAsState(targetValue: T, typeConverter…        State<T>
animateIntOffsetAsState(targetValue: IntO…        State<IntOffset>
animateIntSizeAsState(targetValue: IntSize,…        State<IntSize>
```

animate*AsState



```
animateas
animateFloatAsState(targetValue: Float, ...)          State<Float>
animateColorAsState(targetValue: Color, ...)          State<Color>
animateDpAsState(targetValue: Dp, ...)                State<Dp>
animateIntAsState(targetValue: Int, ...)              State<Int>
animateOffsetAsState(targetValue: Offset, ...…        State<Offset>
animateRectAsState(targetValue: Rect, ...)            State<Rect>
animateSizeAsState(targetValue: Size, ...)            State<Size>
animateValueAsState(targetValue: T, typeConverter…    State<T>
animateIntOffsetAsState(targetValue: IntO…            State<IntOffset>
animateIntSizeAsState(targetValue: IntSize,…          State<IntSize>
```

animate*AsState

```
@Composable
fun Heart() {
    var animated by remember { mutableStateOf(false) }
    val color = if (animated) Color.Red else Color(0xFFFB79A8)

    ...

  Image(
    ...
    colorFilter = ColorFilter.tint(color)
)
```