

# DEUCE

## A Lightweight UI For Structured Editing

Brian Hempel, Justin Lubin, Grace Lu, and Ravi Chugh



THE UNIVERSITY OF  
CHICAGO

# **Unstructured text is clumsy**

ñjò-| I [AÍÓÓnLE...@iq6Èûâ8€tù.À, :◊ÿc; :2πU''LZ™Yz±ii  
ÒJ#| suurü= {~' ÍðÀûÛ ``<> bN™‡EvPèÆêÛN, Ω≥>n° fTπc"  
¤/≤õõ...+Tã@\$TwhJ<ÍÚ% (8" ÍÛFz5£:Ø< i I-^ð9\*™s\* ^ . Íuj  
±[ i !ùØ\$3áUà · Í • 'Œ◊≥≥ `ë\ ÍÌ\Ä" dfi<u ¥v£1 Ýzáõ (cõ÷◊·≥  
'o¶ç±d, ¥fl3° BH? \_i ~ ' √ <Œz¢§"ÀK\*Ã«? <` öÓæN(HRXU-â`á  
ÜÍIzÅ: ¶i dSÝì0πJFíi"÷∞: ^ "†V» ' , "" ôdåuTΩ{<héäÄË );  
o, U) è#æ · J a fít Í Óí2 ÜdHÃì2 ° @' j À= · Ú° @\*qx] b èuSzÃµxKø  
&PkY• VÒV¢I | fÉI i ~ È»ùÙûó2 i > { j ÜdÜß≤ 'QC; L · } n' I" • T≤

ñjò-| I [AÍÓÓnLE...@iq6Èûâ8€tù.À, :◊ÿc; :2πU''LZ™Yz±ii  
ÒJ#| suurü= {~' ÍðÀÛÛ ``<> bN™‡EvPèÆêÛN, Ω≥>n° fTπc"  
¤/≤õõ...+Tã@\$TwhJ<ÍÚ% (8" ÍÛFz5£:Ø< i I-^ð9\*™s\* ^ . Íuj  
±[ ï!ùØ\$3áUà · Í • ' E◊≥≥ ` ë\ ÍI\Ä" dfiu¥v£1Ýzáõ(cô÷◊·≥  
' o¶ç±d, ¥fl3° BH? \_ i ~ ' ✓<Œz¢§"ÀK\*Ã«? < öÓæN(HRXU-â`á  
ÜÍIzÅ: ¶idSÝì0πJFíi"÷∞: "†V» ' , "" ôdåuTΩ{<héäÄË);  
o, U) è#æ · J a fit Í Óí2ÜdHÃì2° @' j À= · Ú°@\*qx] bëuSzÃµxKø  
&PkY•VÒV¢Ì|fÉI i ~ È»ùÙûó2i > {j ÜdÜß≤'QC; L · }n'I" • T≤

GHOST: So art thou to revenge, when thou shalt hear.

HAMLET: What?

GHOST: I am thy father's spirit,  
Doom'd for a certain term to walk the night,  
And for the day confin'd to fast in fires,  
Till the foul crimes done in my days of nature  
Are burnt and purged away. But that I am forbid  
To tell the secrets of my prison-house,  
I could a tale unfold whose lightest word  
Would harrow up thy soul; freeze thy young blood,

ñjò-| I [AÍÓÓnLE...@iq6Èûâ8€tù.À, :◊ÿc; :2πU''LZ™Yz±ï/ØJ#| suurü= {~' ÌøÀûÛ` «<> · bN™‡EvPèÆêÛN, Ø≥>n°fTπc"·  
¤√≤ñO...+Tä@\$TwhJ<ÌÚ% (8" ÌÛFz5£:Ø< i I-^ø9\*™s\*` · Ìuj·  
±[i!ùØ\$3áUà · Ì• 'Œ◊≥≥' ë\Ìì\Ä"dfi¥v£1Ýzáõ(cõ÷◊·≥·  
'o¶ç±d, ¥fl3°ßH? \_i ~ ' √<Œz¢§“ÀK\*Ã«? <` öÓæN( HRXU-â` ád  
ÜÏIzÅ:¶idSÝì0πJFíi"÷∞: " " †V»' , "" ôdâuTΩ{<héääË );  
o, U) è#æ · J¤fitÌÖí2ÜdHÃì2°@' jÀ= · Ú°@\*qx] bëuSzÃµxKø  
&PkY•vòv¢Ì| fÉI i~Ê»ùÙûó2i> {jÛdÜß≤'ΩC; L· }n'I"· T≤

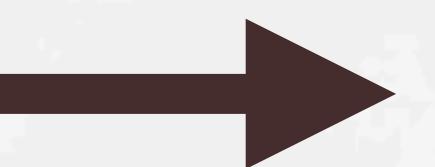
GHOST: So art thou to revenge, when thou shalt hear.

HAMLET: What?

GHOST: I am thy father's spirit,  
Doom'd for a certain term to walk the night,  
And for the day confin'd to fast in fires,  
Till the foul crimes done in my days of nature  
Are burnt and purged away. But that I am forbid  
To tell the secrets of my prison-house,  
I could a tale unfold whose lightest word  
Would harrow up thy soul; freeze thy young blood,

```
maybeZip : List a -> List b -> Maybe (List (a,b))
maybeZip xs ys = case (xs, ys) of
    (x::xs_, y::ys_) -> case maybeZip xs_ ys_ of
        Nothing -> Nothing
        Just xys -> Just ((x,y) :: xys)
    ([] , [])          -> Just []
    _                  -> Nothing
```

ñjò-| I [AÍÓÓnLE...@iq6Èûâ8€tù.À, :◊ÿc; :2πU''LZ™Yz±ii  
ÒJ#| suurü= {~' ÍðÀÛÛ ``<> ` bN™‡EvPèÆêÛN, Ω≥>n° fTπc"  
¤/≤õ...+Tã@\$TwhJ<ÍÛ% (8" ÍÛFz5£:Ø< i I-^ð9\*™s\* ^ . Íuj  
±[ i !ùØ\$3áUà · Í • 'Œ◊≥≥ ` ë\ ÍI\Ä" dfi<u¥v£1Ýzáõ(cõ÷◊·≥  
'o¶ç±d, ¥fl3° BH? \_i ~' √<Œz¢§"ÀK\*Ã«? < öÓæN(HRXU-â`á  
ÜÍIzÅ: ¶idSÝì0πJFíi"÷∞: ^ "†V» ' , "" ôdåuTΩ{<héäÄË);  
o, U) è#æ · Jäfit ÍÖí2ÜdHÄì2° @' jÀ= · Ú°@\*qx] bèuSzÄµxKø  
&PkY•VÒV¢Ì|fÉI i ~Ê»ùÙûó2i > {jÙdÜß≤'QC; L · }n'I" •T≤



# Not a program

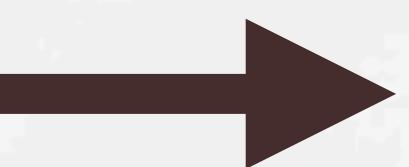
GHOST: So art thou to revenge, when thou shalt hear.

HAMLET: What?

GHOST: I am thy father's spirit,  
Doom'd for a certain term to walk the night,  
And for the day confin'd to fast in fires,  
Till the foul crimes done in my days of nature  
Are burnt and purged away. But that I am forbid  
To tell the secrets of my prison-house,  
I could a tale unfold whose lightest word  
Would harrow up thy soul; freeze thy young blood,

```
maybeZip : List a -> List b -> Maybe (List (a,b))
maybeZip xs ys = case (xs, ys) of
  (x::xs_, y::ys_) -> case maybeZip xs_ ys_ of
    Nothing -> Nothing
    Just xys -> Just ((x,y) :: xys)
  ([] , [])
  -> Just []
  -> Nothing
```

ñjò-| I [AÍÓÓnLE...@iq6Èûâ8€tù.À, :◊ÿc; :2πU''LZ™Yz±ii  
ÒJ#| suurü= {~' ÍðÀÛÛ ``<> ` bN™‡EvPèÆêÛN, Ω≥>n° fTπc"  
¤/≤õ...+Tã@\$TwhJ<ÍÛ% (8" ÍÛFz5£:Ø< i I-^ð9\*™s\* ^ . Íuj  
±[ i !ùØ\$3áUà · Í • 'Œ◊≥≥ ` ë\ ÍÍ\Ä" dfi<u¥v£1Ýzáõ(cõ÷◊·≥  
'o¶ç±d, ¥fl3° BH? \_i ~' √<Œz¢§"ÀK\*Ã«? < öÓæN(HRXU-â`á  
ÜÍIzÅ: ¶idSÝì0πJFíi"÷∞: ^ "†V» ' , "" ôdåuTΩ{<héäÄË);  
o, U) è#æ · Jäfit ÍÖí2ÜdHÄì2° @' jÀ= · Ú°@\*qx] bëuSzÄµxKø  
&PkY•VÒV¢Ì|fÉI i ~Ê»ùÙûó2i > {jÙdÜß≤'QC; L · }n'I" •T≤

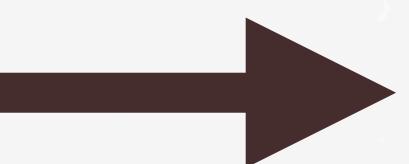


# Not a program

GHOST: So art thou to revenge, when thou shalt hear.

HAMLET: What?

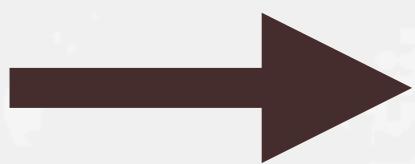
GHOST: I am thy father's spirit,  
Doom'd for a certain term to walk the night,  
And for the day confin'd to fast in fires,  
Till the foul crimes done in my days of nature  
Are burnt and purged away. But that I am forbid  
To tell the secrets of my prison-house,  
I could a tale unfold whose lightest word  
Would harrow up thy soul; freeze thy young blood,



# Not a program

```
maybeZip : List a -> List b -> Maybe (List (a,b))
maybeZip xs ys = case (xs, ys) of
  (x::xs_, y::ys_) -> case maybeZip xs_ ys_ of
    Nothing -> Nothing
    Just xys -> Just ((x,y) :: xys)
  ([] , [])
  -> Just []
  -> Nothing
```

ñjò-| I [AÍÓÓnLE...@iq6Èûâ8€tù.À, :◊ÿc; :2πU''LZ™Yz±ii  
ÒJ#| suurü= {~' ÍðÀÛÛ ``<> ` bN™‡EvPèÆêÛN, Ω≥>n° fTπc"  
¤/≤õ...+Tã@\$TwhJ<ÍÛ% (8" ÍÛFz5£:Ø< i I-^ð9\*™s\*` Íuj'  
±[ i !ùØ\$3áUà· Í• 'Œ◊≥≥` ë\ ÍI\Ä" dfi<u¥v£1Ýzáõ(cõ÷◊·≥  
'o¶ç±d, ¥fl3° BH?\_i ~' √<Œz¢§"ÀK\*Ã«? < öÓæN(HRXU-â`á  
ÜÍIzÅ:¶idSÝì0πJFíi"÷∞: "†V»' ,"" ôdåuTΩ{<héäÄE);  
o, U) è#æ· JäfitÍÖí2ÜdHÄì2°@' jÀ=· Ú°@\*qx] bëuSzÄµxKø  
&PkY•VÒV¢Ì|fÉI i ~È»ùÙûó2i>{jÙdÜß≤'QC;L· }n'I"•T≤

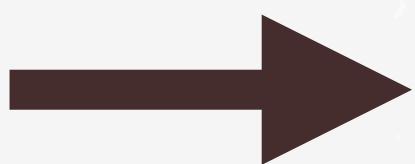


# Not a program

GHOST: So art thou to revenge, when thou shalt hear.

HAMLET: What?

GHOST: I am thy father's spirit,  
Doom'd for a certain term to walk the night,  
And for the day confin'd to fast in fires,  
Till the foul crimes done in my days of nature  
Are burnt and purged away. But that I am forbid  
To tell the secrets of my prison-house,  
I could a tale unfold whose lightest word  
Would harrow up thy soul; freeze thy young blood,



# Not a program

```
maybeZip : List a -> List b -> Maybe (List (a,b))
maybeZip xs ys = case (xs, ys) of
  (x:xs_, y:ys_) -> case maybeZip xs_ ys_ of
    Nothing -> Nothing
    Just xys -> Just ((x,y) :: xys)
  [] , []           -> Just []
  _                -> Nothing
```



# Not a program (hidden syntax error)

# Problem for Beginners

# Problem for Beginners

```
-- SYNTAX PROBLEM ----- ./Utils.elm
Arrows are reserved for cases and anonymous functions. Maybe you want > or >=
instead?
46| Nothing -> Nothing
          ^
```

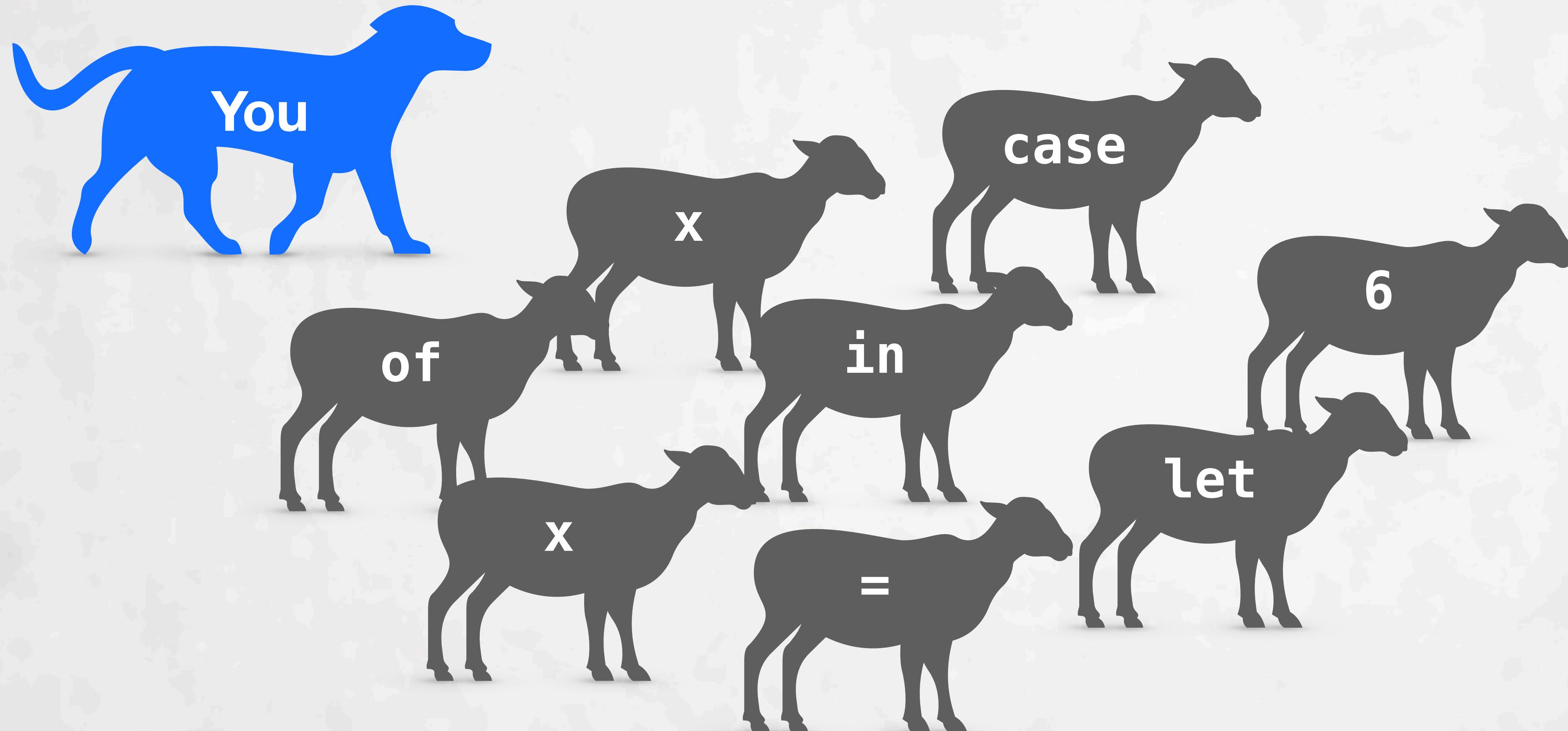
# Problem for Beginners

```
-- SYNTAX PROBLEM ----- ./Utils.elm
Arrows are reserved for cases and anonymous functions. Maybe you want > or >=
instead?
46| Nothing -> Nothing
          ^
```

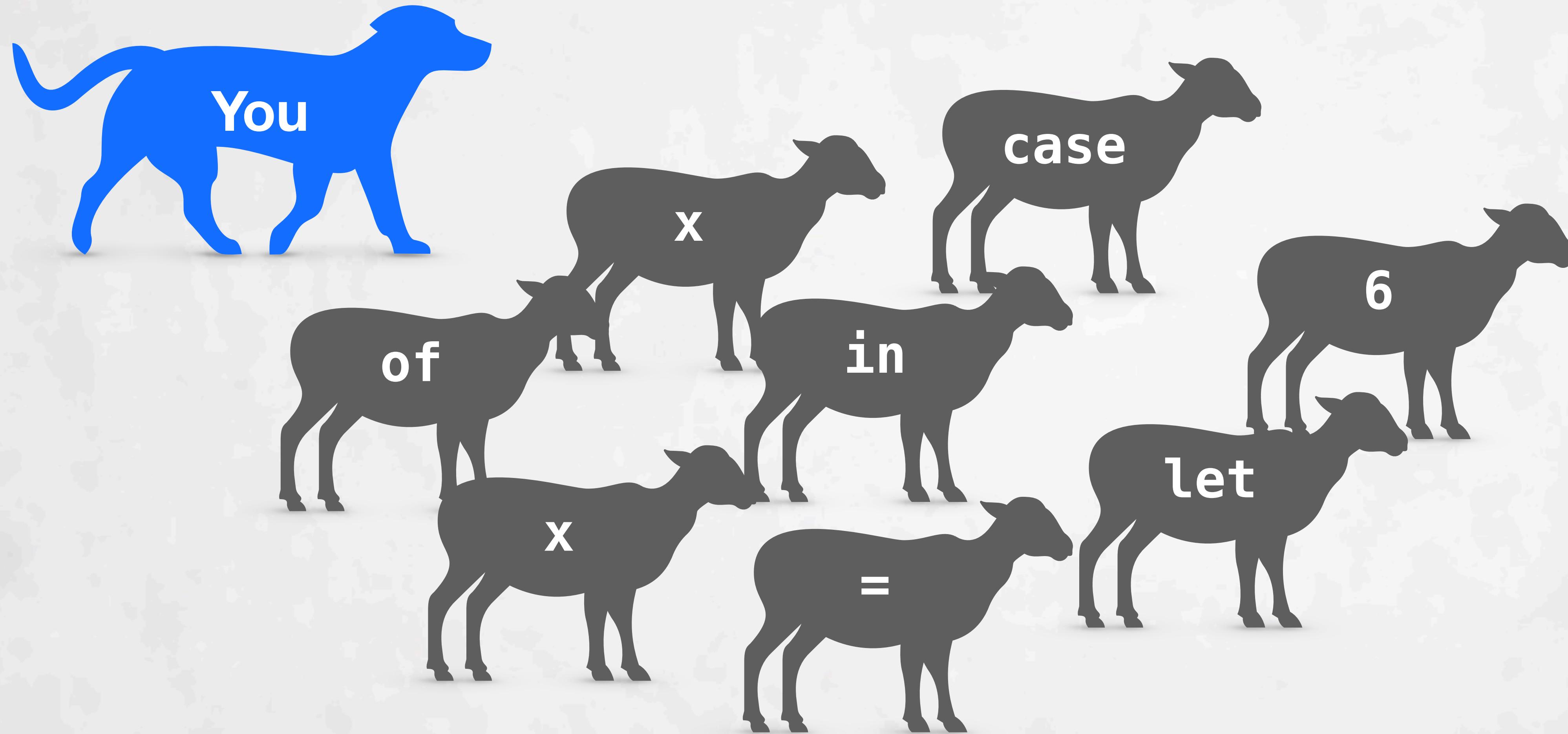
**Structure must come from your head!**

# Problem for Experts

# Problem for Experts



# Problem for Experts



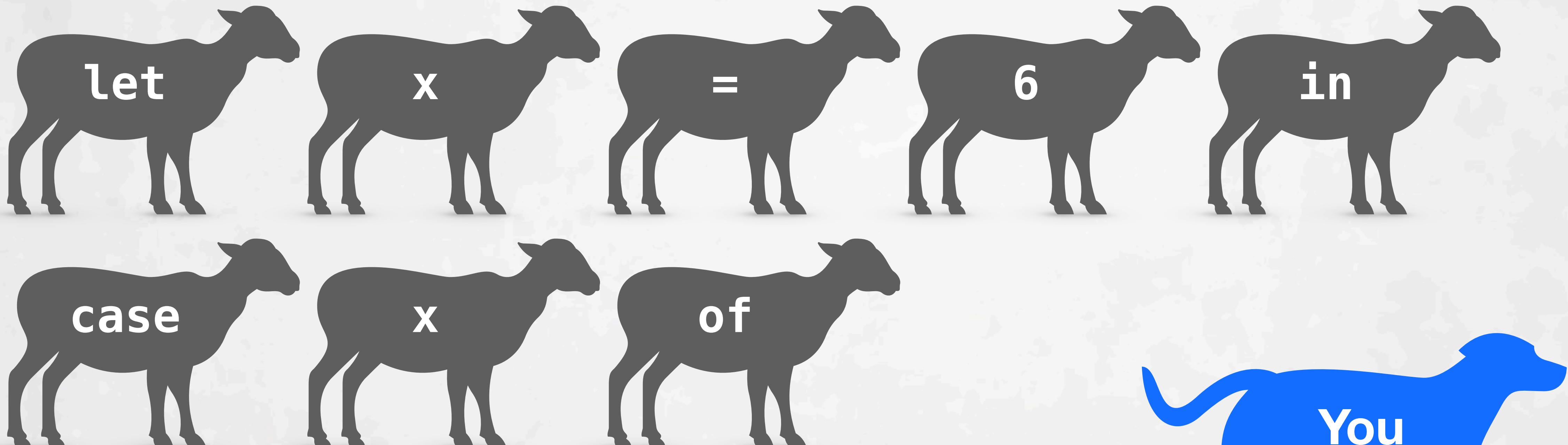
**Spend time herding text!**

# Problem for Experts



**Spend time herding text!**

# Problem for Experts



**Spend time herding text!**



# Your program isn't text.

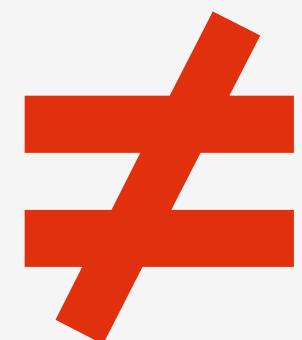
Your program isn't text.  
It's an AST.

# Text Changes

# Text Changes

≠

# Text Changes



# AST Changes

# Structured Editors

# Structured Editors



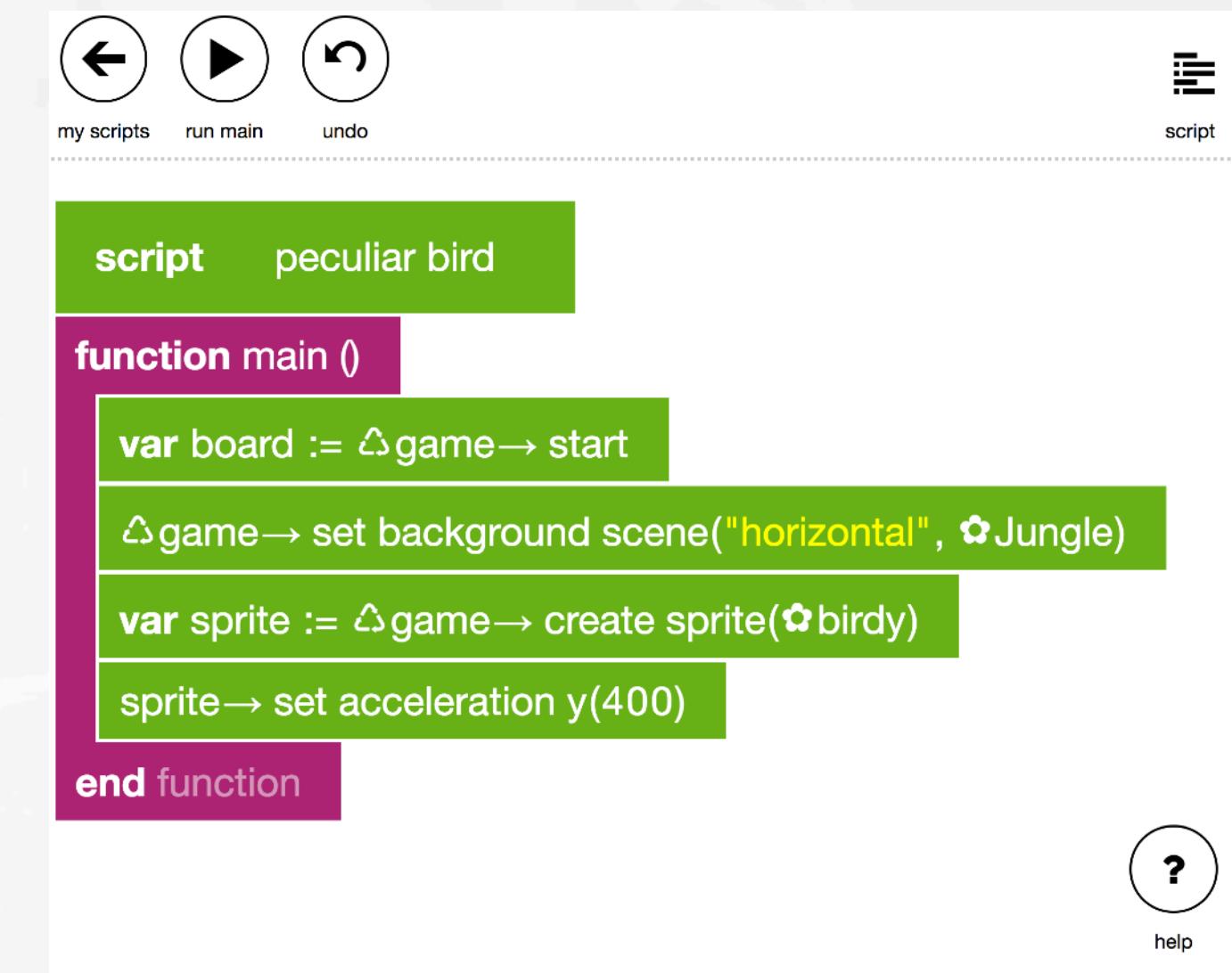
**Scratch**

(Maloney et al. 2010; Resnick et al. 2009)

# Structured Editors



**Scratch**  
(Maloney et al. 2010; Resnick et al. 2009)

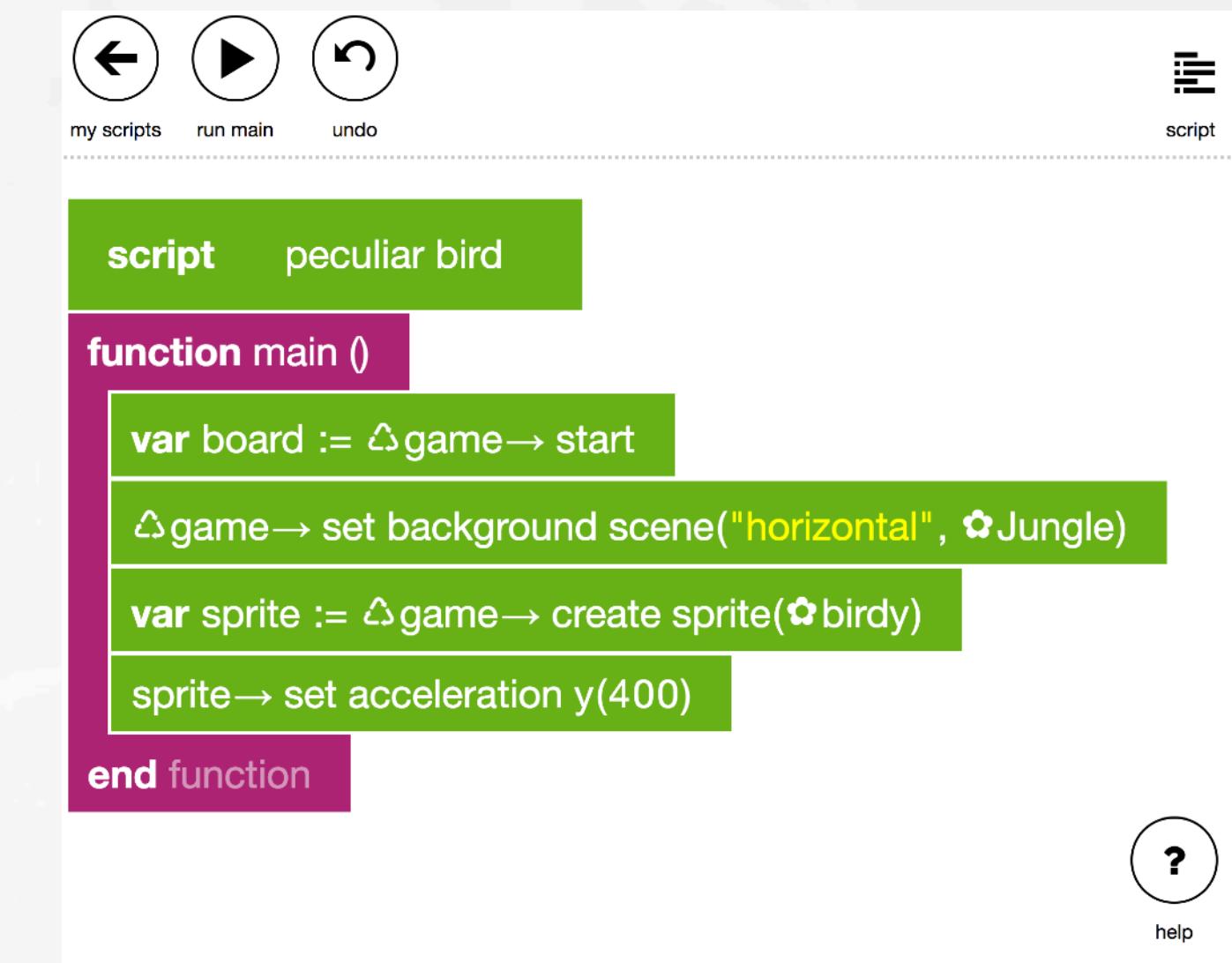


**TouchDevelop**  
(Tillmann et al. 2012)

# Structured Editors



**Scratch**  
(Maloney et al. 2010; Resnick et al. 2009)



**TouchDevelop**  
(Tillmann et al. 2012)

**UI challenges; Experts still use plain text**

# Traditional Refactoring

# Traditional Refactoring

**Text-Select      Menu      Configure**

# Traditional Refactoring

## Text-Select      Menu      Configure

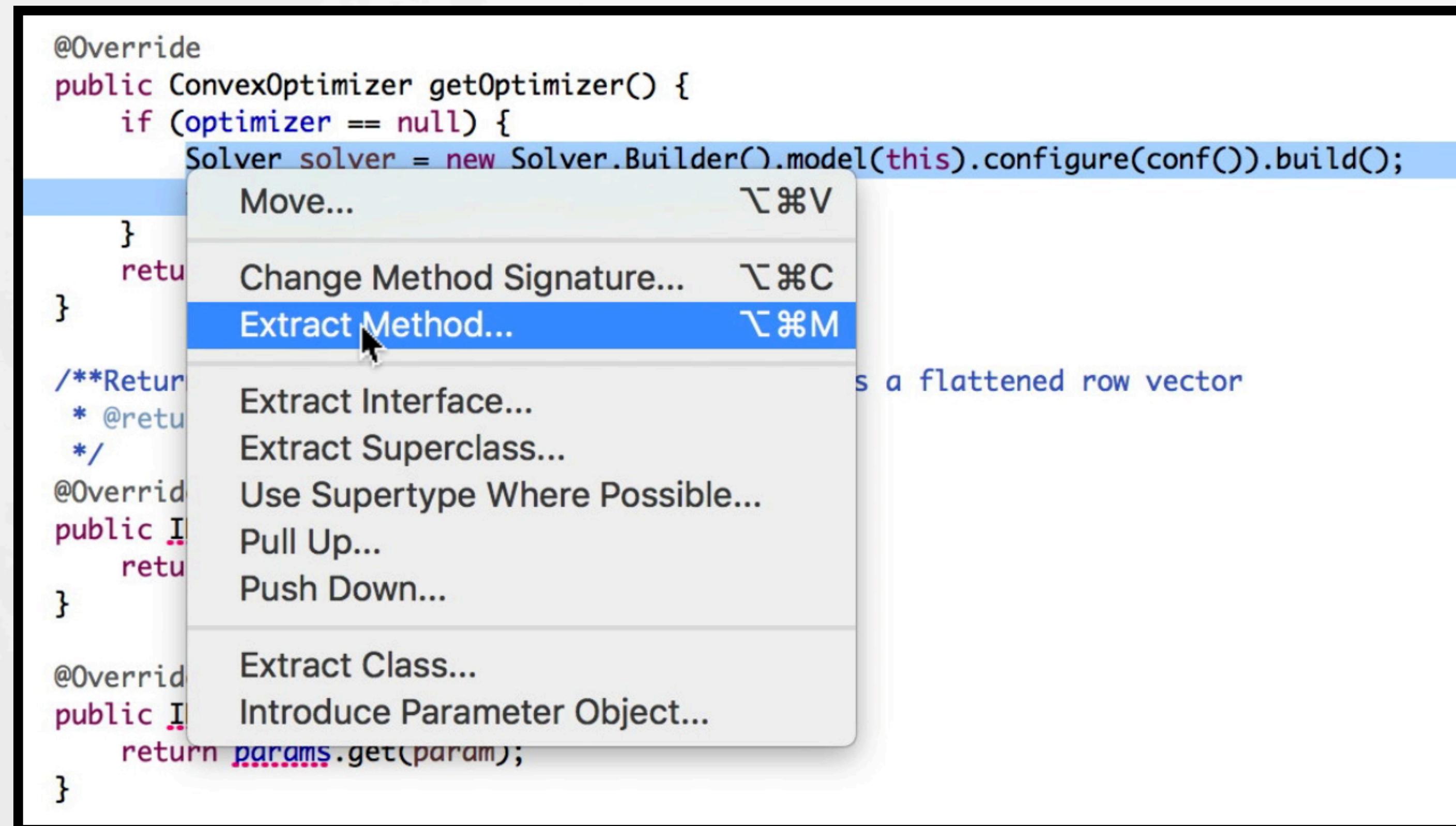
```
@Override
public ConvexOptimizer getOptimizer() {
    if (optimizer == null) {
        Solver solver = new Solver.Builder().model(this).configure(conf()).build();
        this.optimizer = solver.getOptimizer();
    }
    return optimizer;
}

/**Returns the parameters of the neural network as a flattened row vector
 * @return the parameters of the neural network
 */
@Override
public INDArray params() {
    return paramsFlattened;
}

@Override
public INDArray getParam(String param) {
    return params.get(param);
}
```

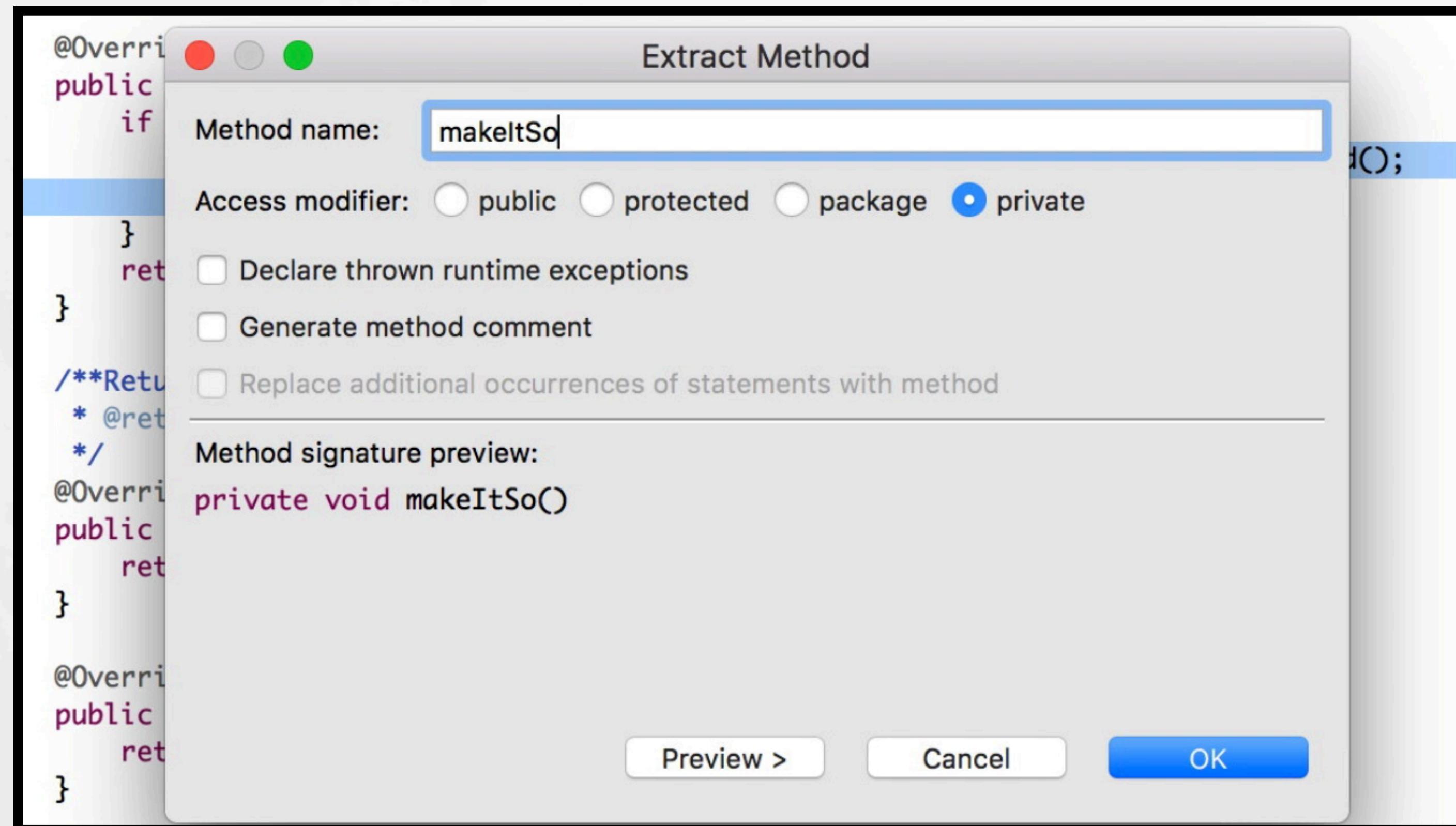
# Traditional Refactoring

## Text-Select      Menu      Configure



# Traditional Refactoring

## Text-Select      Menu      Configure



# Traditional Refactoring

**Text-Select      Menu      Configure**

# Traditional Refactoring

Text-Select

Menu

Configure

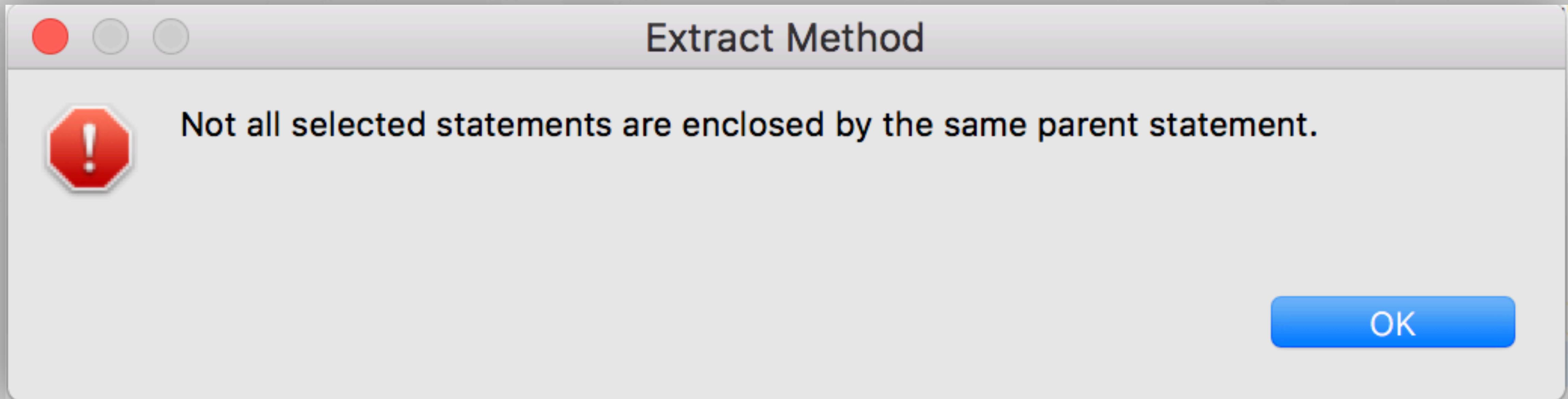
 Awkward

# Traditional Refactoring

Text-Select

Menu

Configure



# Traditional Refactoring

Text-Select

Menu

Configure

 Awkward

# Traditional Refactoring

Text-Select

Menu

Configure

✗ Awkward

✗ Multiple  
Selections

# Traditional Refactoring

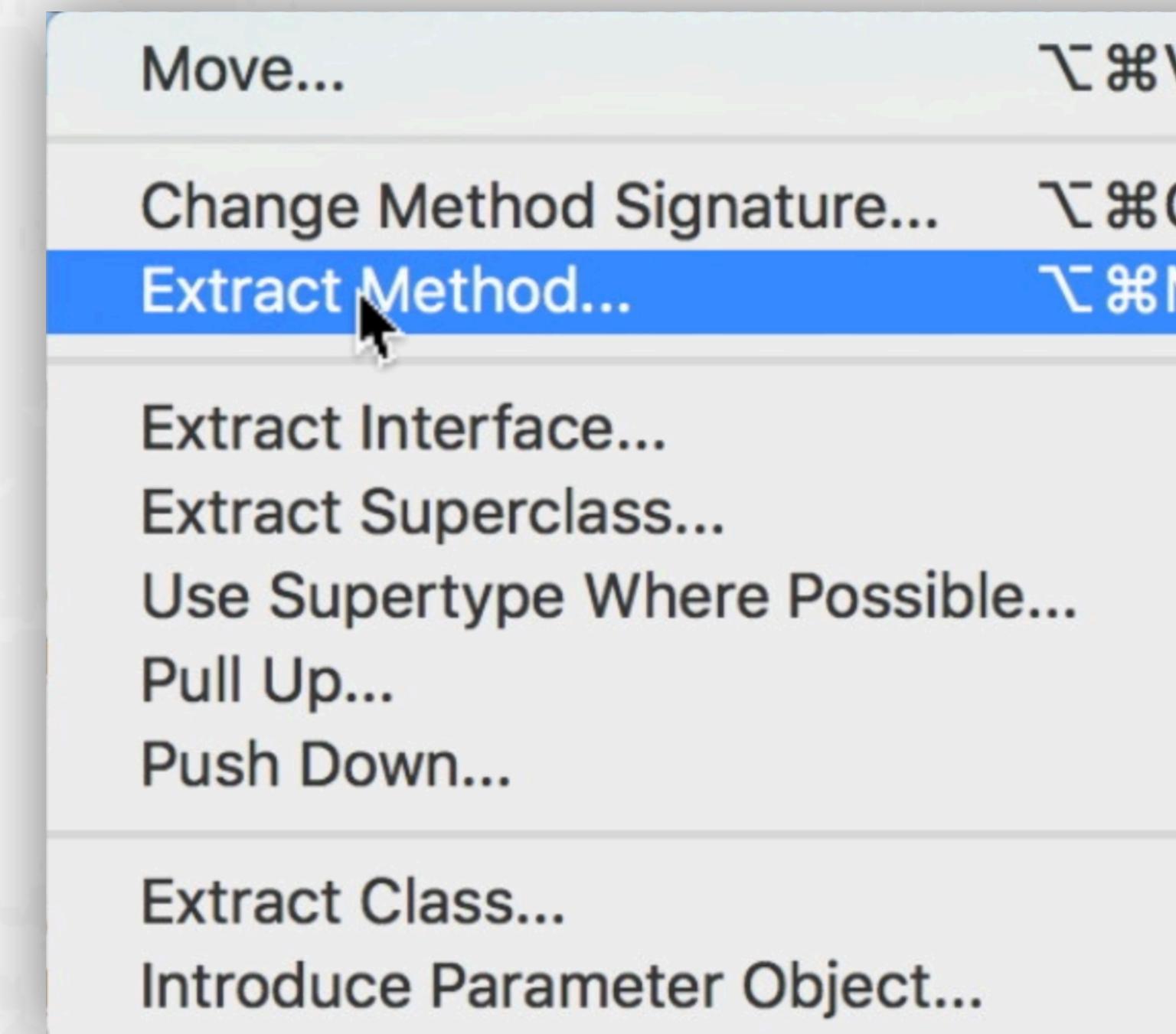
Text-Select

Menu

Configure

✗ Awkward

✗ Multiple  
Selections



# Traditional Refactoring

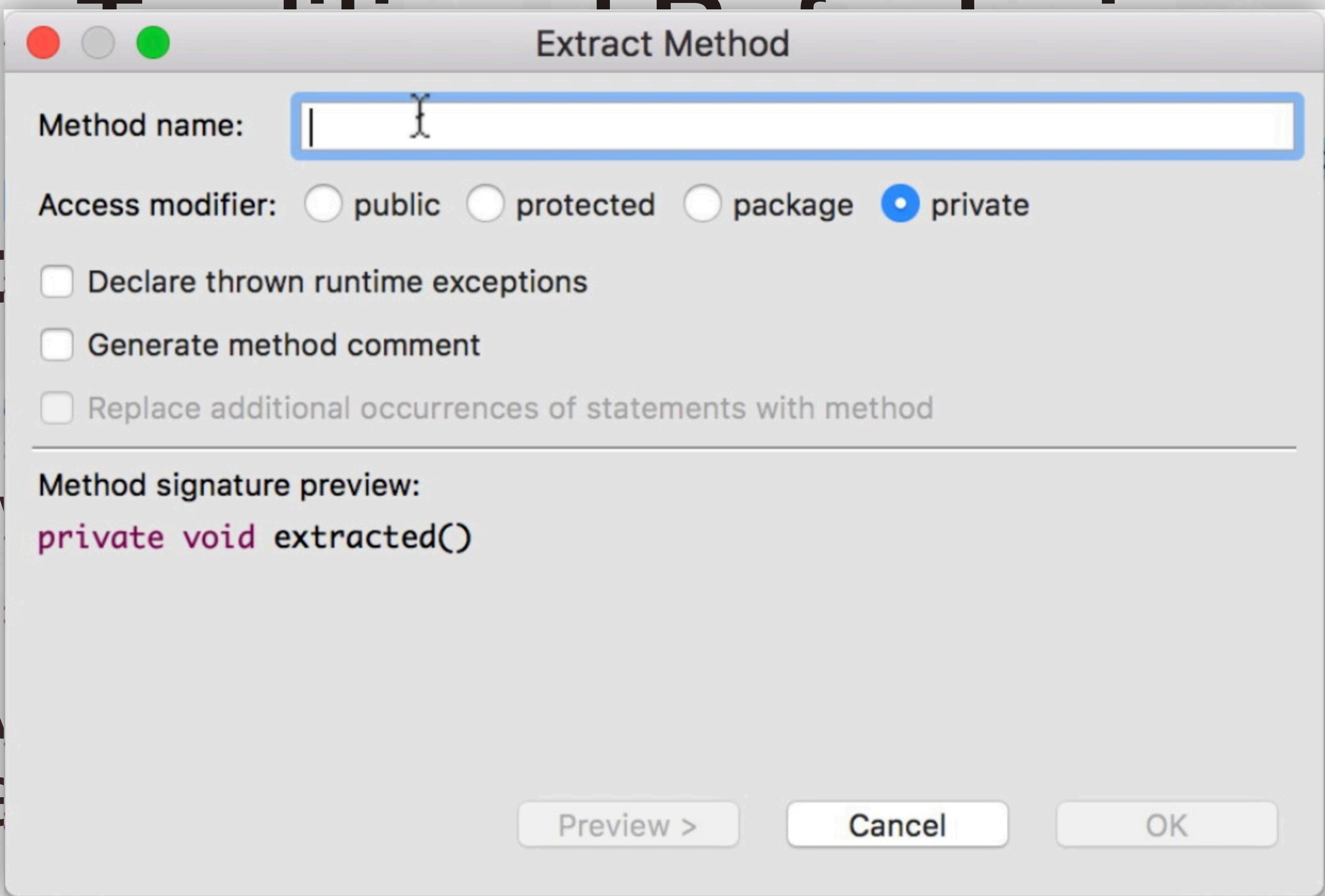
Text-Select

Menu

Configure

- ✗ Awkward
- ✗ Multiple Selections

- ✗ Many Options



Text

gure

X A

X M  
Se

# Traditional Refactoring

Text-Select

Menu

Configure

- ✗ Awkward
- ✗ Multiple Selections

✗ Many Options

✗ Dialogs

# Traditional Refactoring

Text-Select

Menu

Configure

- ✗ Awkward
- ✗ Multiple Selections

- ✗ Many Options

- ✗ Dialogs

# Deuce

Text-Select

Menu

Configure

✗ Awkward  
✗ Multiple  
Selections

✗ Many  
Options

✗ Dialogs

# Deuce

## Structure Select

```
(def image1
  (let [width height]
    (let [x y] [50 65]
      (image "lightgrey"))

  (def main
    (draw (concat [ image1
```

## Menu

✗ Many Options

## Configure

✗ Dialogs

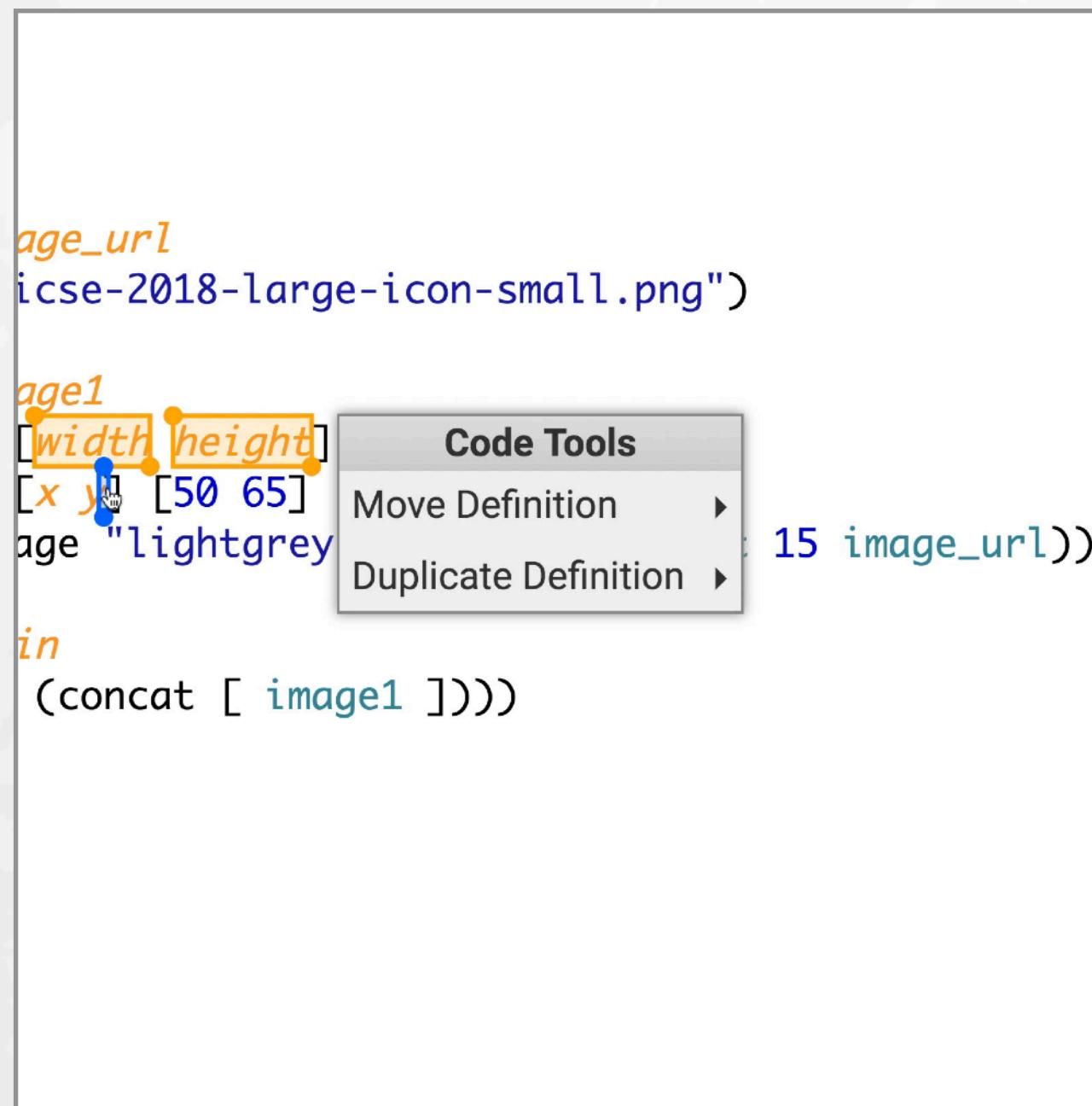
# Deuce

## Structure Select

```
(def image1
  (let [width height]
    (let [x y] [50 65]
      (image "lightgrey"))

  (def main
    (draw (concat [ image1]))
```

## Short Menu



## Configure

X Dialogs

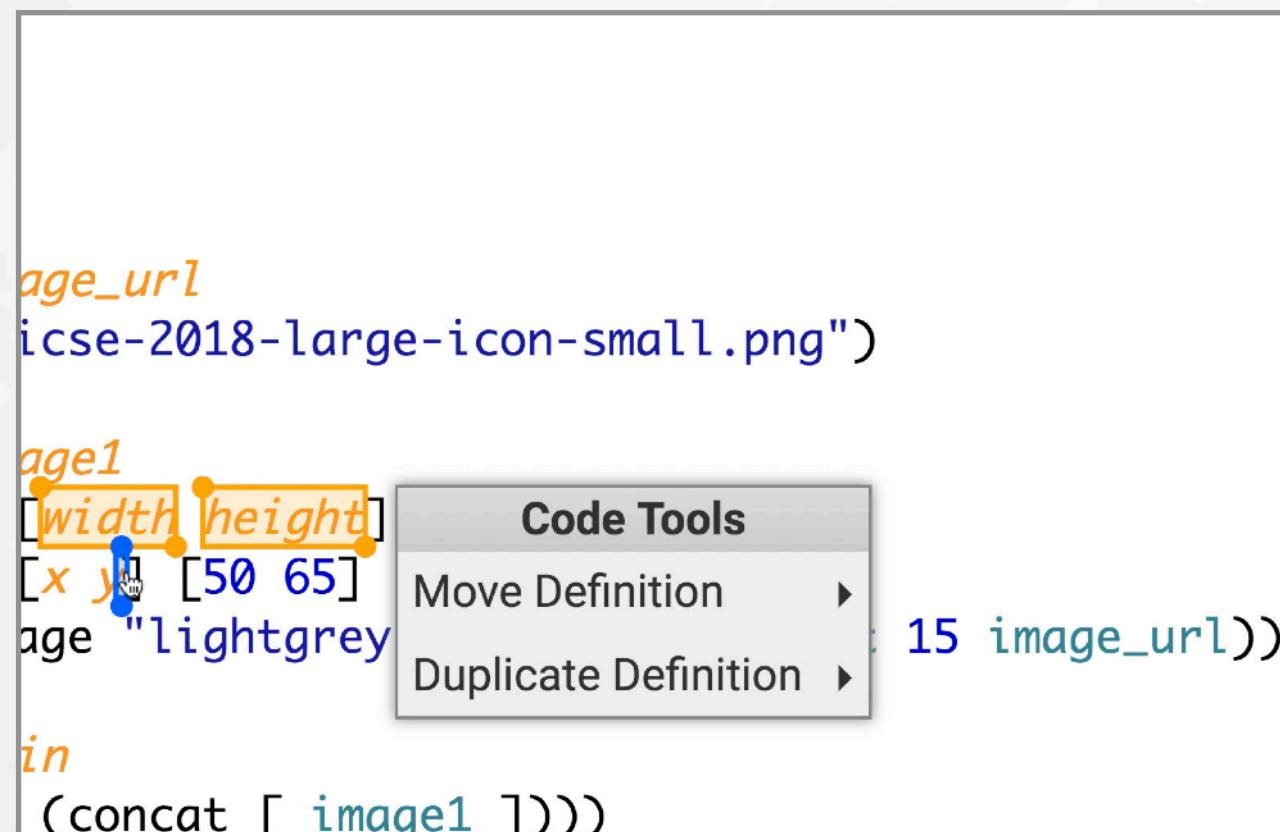
# Deuce

## Structure Select

```
(def image1
  (let [width height]
    (let [x y] [50 65]
      (image "lightgrey"))

  (def main
    (draw (concat [ image1]))
```

## Short Menu



## Defaults

- ▶ Abstract image1 over its constants
- ▶ Abstract image1 over its named constants

# Demo



Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```





Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```





Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```





Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```





Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```





Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```



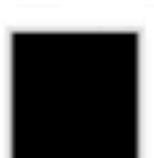


Current file: Untitled \*

 Undo     Redo    Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

 Undo     Redo    Clean Up**Run ▶**

```
1 (def image_url
2   "img/icse-2018-large-icon-")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```

**Code Tools**

- Create Function from Arguments ▶
- Introduce Variable ▶





Current file: Untitled \*

 Undo     Redo    Clean Up**Run ▶**

```
1 (def image_url
2   "img/icse-2018-large-icon")
3
4 (def image1
5   (image "lightgrey" 50 65 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ image1 ])))
```

**Code Tools**

Create Function from Arguments ▶

Introduce Variable ▶

Introduce Variable





Current file: Untitled \*

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon")
3
4 (def image1
5   (let width 283
6     (image "lightgrey" 50 65 width 254 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```

## Code Tools

Create Function from Arguments ▶

Introduce Variable ▶

Introduce Variable

Raw  
Stretchy  
Sticky



Current file: Untitled \*

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (image "lightgrey" 50 65 width 254 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```





Current file: Untitled \*

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (image "lightgrey" 50 65 width 254 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```



Raw  
Stretchy  
Sticky



Current file: Untitled \*

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (image "lightgrey" 50 65 width 254 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

Undo

Redo

Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small."
3
4 (def image1
5   (let width 283
6     (image "lightgrey" 50 65 width 254 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```

- Code Tools
  - Create Function from Arguments ▶
  - Introduce Variable ▶





Current file: Untitled \*

Undo

Redo

Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small."
3
4 (def image1
5   (let width 283
6     (image "lightgrey" 50 65 width 254 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```

Code Tools

- Create Function from Arguments ▶
- Introduce Variable





Current file: Untitled \*

Undo

Redo

Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small")
3
4 (def image1
5   (let width 283)
6   (let height 254)
7   (image "lightgrey" 50 65 width height 15 image_url))
8
9 (def main
10  (draw (concat [ image1 ])))
```

Code Tools

- Create Function from Arguments ➔
- Introduce Variable ➔

▶ Introduce Variable





Current file: Untitled \*

Undo

Redo

Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (let height 254
7       (image "lightgrey" 50 65 width height 15 image_url)))
8
9 (def main
10  (draw (concat [ image1 ])))
```



Raw  
Stretchy  
Sticky



Current file: Untitled \*

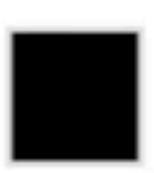
Undo

Redo

Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (let height 254)
7       (image "lightgrey" 50 65 width height 15 image_url)))
8
9 (def main
10  (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

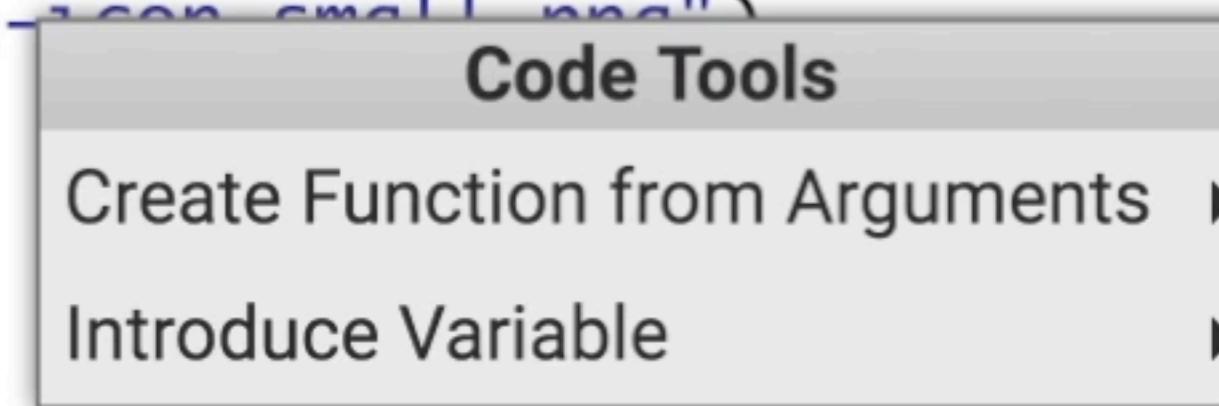
Undo

Redo

Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (let height 254
7       (image "lightgrey" 5 65 width height 15 image_url)))
8
9 (def main
10  (draw (concat [ image1 ])))
```





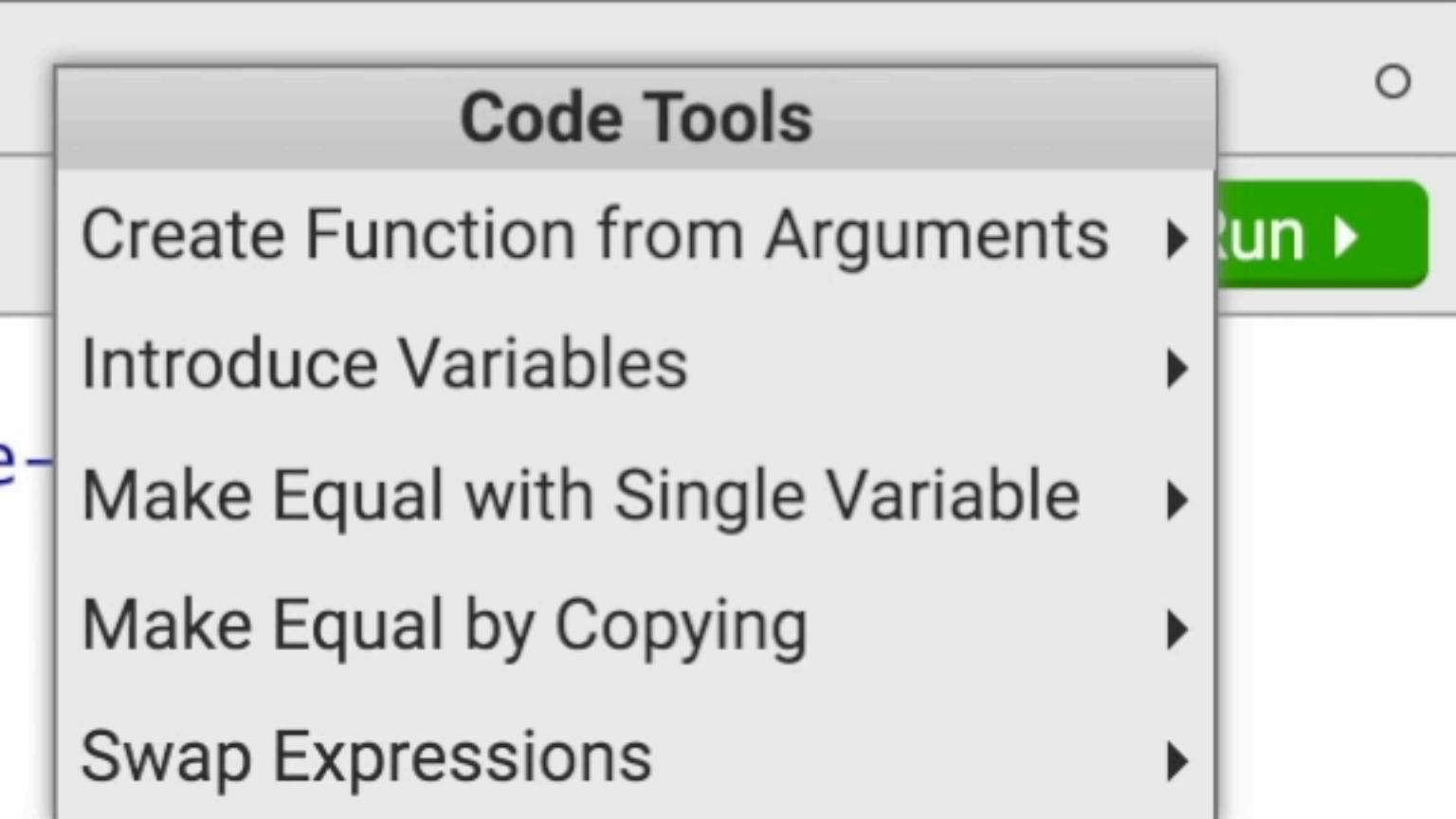
Current file: Untitled \*

Undo

Redo

Clean Up

```
1 (def image_url
2   "img/icse-2018-large")
3
4 (def image1
5   (let width 283
6     (let height 254
7       (image "lightgrey" 50 65 width height 15 image_url)))
8
9 (def main
10  (draw (concat [ image1 ])))
```



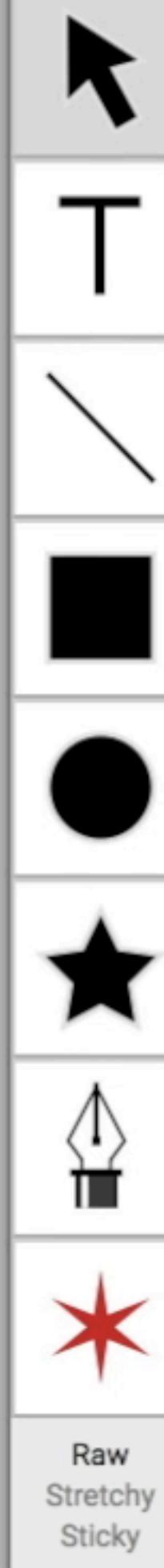
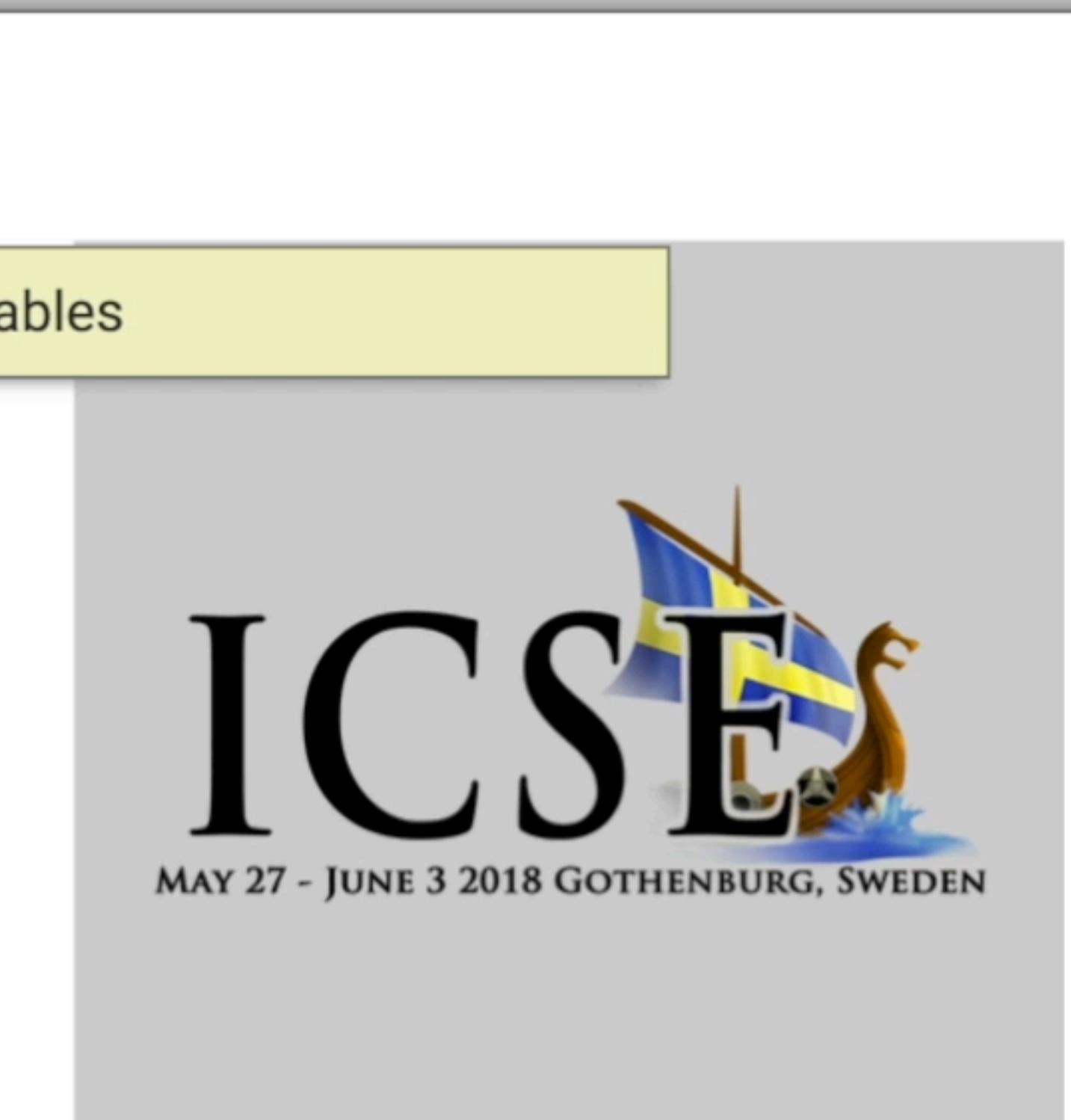
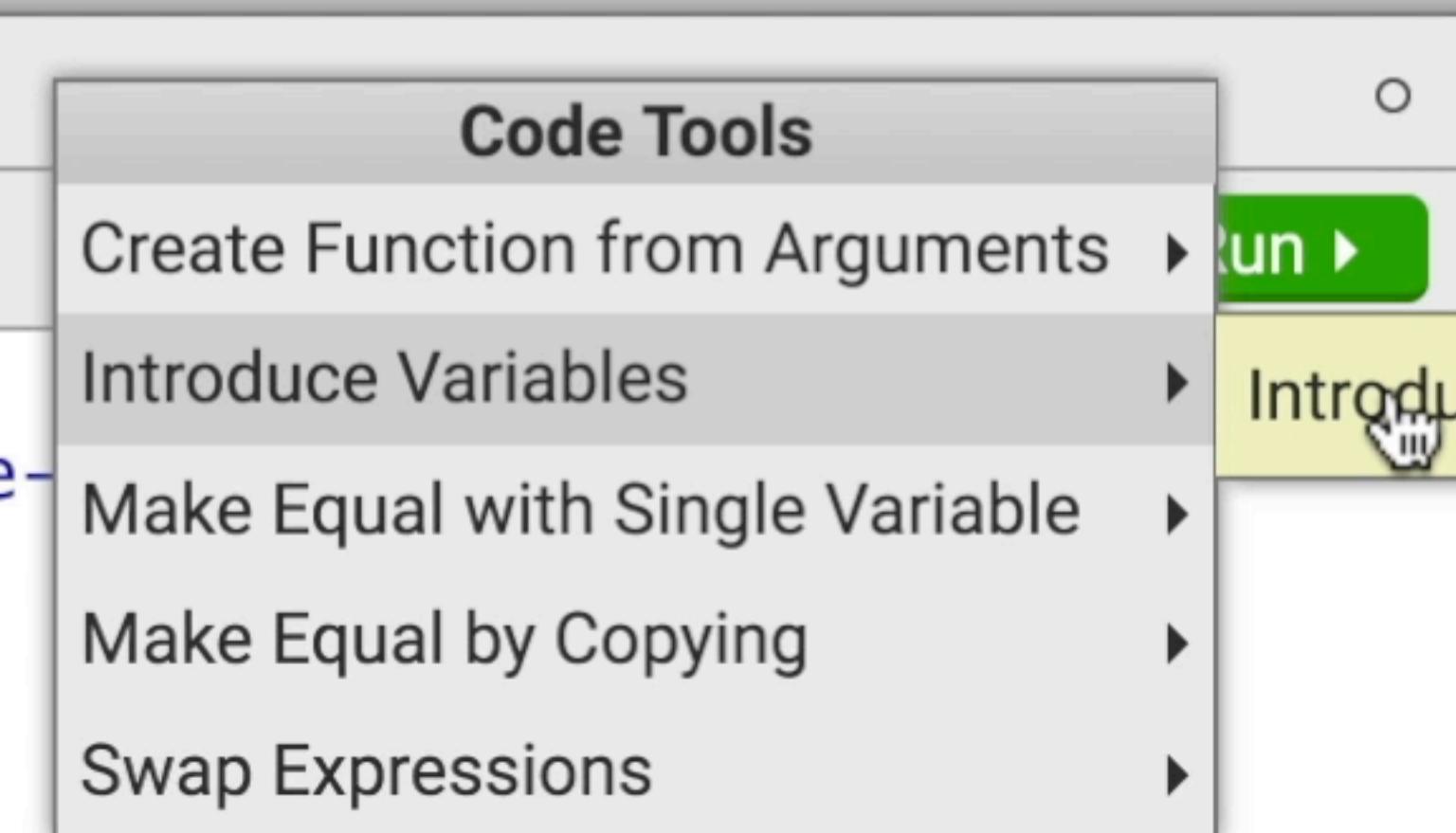


Current file: Untitled \*

Undo Redo

Clean Up

```
1 (def image_url
2   "img/icse-2018-large")
3
4 (def image1
5   (let width 283
6     (let height 254
7       (let [x y] [50 65]
8         (image "lightgrey" x y width height 15 image_url))))
9
10 (def main
11   (draw (concat [ image1 ])))
```





Current file: Untitled \*

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let width 283
6     (let height 254
7       (let [x y] [50 65]
8         (image "lightgrey" x y width height 15 image_url))))
9
10 (def main
11   (draw (concat [ image1 ])))
```





Current file: Untitled \*

Undo

Redo

Clean Up

Run ▶

```
1 (def imag
2   "img/ic
3     n-small.png")
4 (def imag
5   (let width 283
6     (let height 254
7       (let [x y] [50 65]
8         (image "lightgrey" x y width height 15 image_url))))
9
10 (def main
11   (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled

Undo

Redo

```
1 (def imag
2   "img/ic
3
4 (def imag
5   (let width 283
6     (let height 254
7       (let [x y] [50 65]
8         (image "lightgrey" x y width height 15 image_url))))
9
10 (def main
11   (draw (concat [ image1 ])))
```

## Code Tools

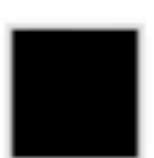
Create Function from Definition

Create Function from Arguments

Rename width

Inline Definition

Run ▶

Raw  
Stretchy  
Sticky



Current file: Untitled \*

Undo

Redo

Clean Up

Run ▶

```
1 (def imag
2   "img/ic
3     n-small.png")
4 (def image1
5   (let [width 283
6       height 254
7       [x y] [50 65]]
8     (image "lightgrey" x y width height 15 image_url)))
9
10 (def main
11   (draw (concat [ image1 ])))
```





Current file: Untitled \*

Undo

Redo

Clean Up

Run ▶

```
1 (def imag
2   "img/ic"
3
4 (def imag
5   (let [width height] [283 254]
6     (let [x y] [50 65]
7       (image "lightgrey" x y width height 15 image_url))))
8
9 (def main
10  (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

Undo

Redo

Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let [width height] [283 254]
6     (let [x y] [50 65]
7       (image "lightgrey" x y width height 15 image_url))))
8
9 (def main
10  (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

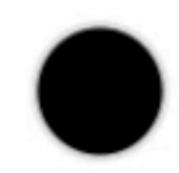
Undo

Redo

Clean Up

Run ▶

```
1 (def image1
2   "img/icse-small.png")
3
4 (def image1
5   (let [width height] [283 254])
6   (let [x y] [50 65]
7     (image "lightgrey" x y width height 15 image_url))))
8
9 (def main
10  (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

Undo

Redo

Clean Up

Run ▶

```
1 (def image1
2   "img/icse")
3
4 (def image1
5   (let [x y width height] [50 65 283 254]
6     (image "lightgrey" x y width height 15 image_url)))
7
8 (def main
9   (draw (concat [ image1 ])))
```

Move width and height

Raw  
Stretchy  
Sticky



Current file: Untitled \*

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1
5   (let [x y width height] [50 65 283 254] [
6     (image "lightgrey" x y width height 15 image_url)])
7
8 (def main
9   (draw (concat [ image1 ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

## Code Tools

Undo Create Function from Definition ▾

1 (Inline Definition ▾

2 Make Single Line ▾

4 (def image1

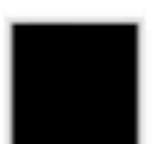
5 (let [x y width height] [50 65 283 254]

6 (image "lightgrey" x y width height 15 image\_url)))

8 (def main

9 (draw (concat [ image1 ])))

Run ▶

Raw  
Stretchy  
Sticky



Current file: Untitled \*

## Code Tools

Undo

Create Function from Definition

1 (c) Inline Definition

2 Make Single Line

4 (def image1

5 (let [x y width height] [50 65 283 254]

6 (image "lightgrey" x y width height 15 image\_url)))

8 (def main

9 (draw (concat [ image1 ])))

Abstract image1 over its constants

 Abstract image1 over its named  
 constantsRaw  
Stretchy  
Sticky



Current file: Untitled \*

## Code Tools

Undo

Create Function from Definition

Abstract image1 over its constants

1 (o) Inline Definition

Abstract image1 over its named

2 Make Single Line

constants

```
4 (def image1 (\(x y width height bgColor padding)
 5   (image bgColor x y width height padding| image_url)))
```

```
6
7 (def main
```

```
8   (draw (concat [ (image1 50 65 283 254 "lightgrey" 15) ]
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

## Code Tools

Undo Create Function from Definition

1 (c) Inline Definition

2 Make Single Line

 4 (def *image1* (\(x y width height)

5 (image "lightgrey" x y width height 15 |image\_url|))

6

 7 (def *main* 8 (draw (concat [ (*image1* 50 65 283 254) ])))

Abstract image1 over its constants

 Abstract image1 over its named  
 constants



Current file: Untitled \*

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1 (x y width height)
5   (image "lightgrey" x y width height 15 image_url))
6
7 (def main
8   (draw (concat [ (image1 50 65 283 254) ])))
```

Raw  
Stretchy  
Sticky



Sketch-n-Sketch

File

Code Tools

Output Tools

View

Options

**Code Tools**

- Remove Arguments
- Swap Variable Names and Usages
- Swap Variable Usages
- Inline Definitions

**Run ▶**

Current file: Untitled \*

Undo

Redo

Clean U

```
1 (def image_url
2   "img/icse-2018-large")
3
4 (def image1 ((x y width height)
5   (image "lightgrey" x y width height 15 image_url)))
6
7 (def main
8   (draw (concat [ (image1 50 65 283 254) ])))
```

Raw  
Stretchy  
Sticky



Sketch-n-Sketch

File

Code Tools

Output Tools

View

Options

**Code Tools**

- Remove Arguments ➔ Remove Arguments
- Swap Variable Names and Usages ➔ Run ➔
- Swap Variable Usages ➔
- Inline Definitions ➔

Current file: Untitled \*

Undo

Redo

Clean U

```
1 (def image_url
 2   "img/icse-2018-large")
3
4 (def image1 (\(x y)
5   (image "lightgrey" x y 283 254 15 image_url)))
6
7 (def main
8   (draw (concat [ (image1 50 65) ])))
```



Raw  
Stretchy  
Sticky

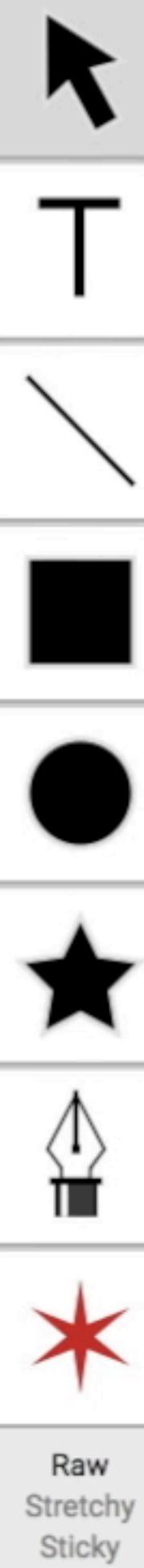


Current file: Untitled \*

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def image1 (x y)
5   (image "lightgrey" x y 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ (image1 50 65) ])))
```





Sketch-n-Sketch

File

Code Tools

Output Tools

View

Options

Current file: Untitled

Undo

```
1 (def image1
 2   "img/
 3   Inline Definition
 4 (def image1 (\(x y)
 5   (image "lightgrey" x y 283 254 15 image_url)))
 6
 7 (def main
 8   (draw (concat [ (image1 50 65) ])))
```

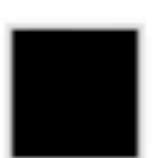
**Code Tools**

Create Function from Definition

Rename *image1*

Inline Definition

Run ▶

Raw  
Stretchy  
Sticky



Current file: Untitled

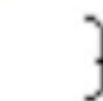
Undo

## Code Tools

Create Function from Definition ▶

Run ▶

```
1 (def image1
2   "img/"
3
4 (def image1 (\(x y)
5   (image "lightgrey" x y 283 254 15 image_url)))
6
7 (def main
8   (draw (concat [ (image1 50 65) ])))
```

Rename *image1* to...



Current file: Untitled

Undo

## Code Tools

Create Function from Definition ▶

Run ▶

1 (def *image1* "img/ Rename *image1* to icse20182 "img/  
3 Inline Definition ▶

icse2018

4 (def **image1** (\(x y)  
5 (image "lightgrey" x y 283 254 15 image\_url)))6  
7 (def **main**  
8 (draw (concat [ (image1 50 65) ])))



Current file: Untitled \*

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def icse2018 (\x y)
5   (image "lightgrey" x y 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ (icse2018 50 65) ])))
```

Raw  
Stretchy  
Sticky



Current file: Untitled \*

Undo Redo Clean Up

Run ▶

```
1 (def image_url
2   "img/icse-2018-large-icon-small.png")
3
4 (def icse2018 (\x y)
5   (image "lightgrey" x y 283 254 15 image_url))
6
7 (def main
8   (draw (concat [ (icse2018 50 65) ])))
```

Raw  
Stretchy  
Sticky

# Deuce

## Structure Select

## Short Menu

## Defaults

# Deuce

## Structure Select

```
(def image-url
  "img/icse-2018-large-icon-small.png")

(def image1
  (let [width height] [324 200]
  (let [x y] [100 100]
    (image "lightgrey" x y width height 15 image-url)))))

(def main
  (draw (concat [ image1 ])))
```

## Short Menu

```
(def image1
  (let [width height]
  (let [x y] [100 100]
    (image "lightgrey"
```

## Defaults

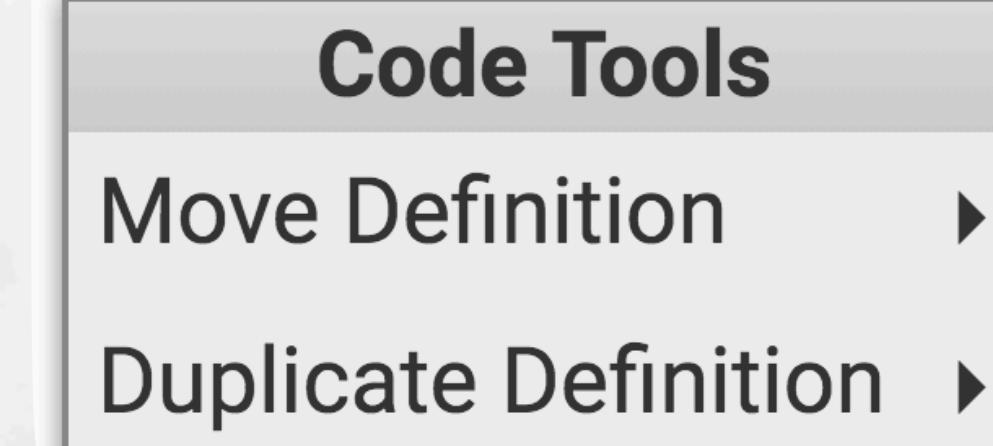
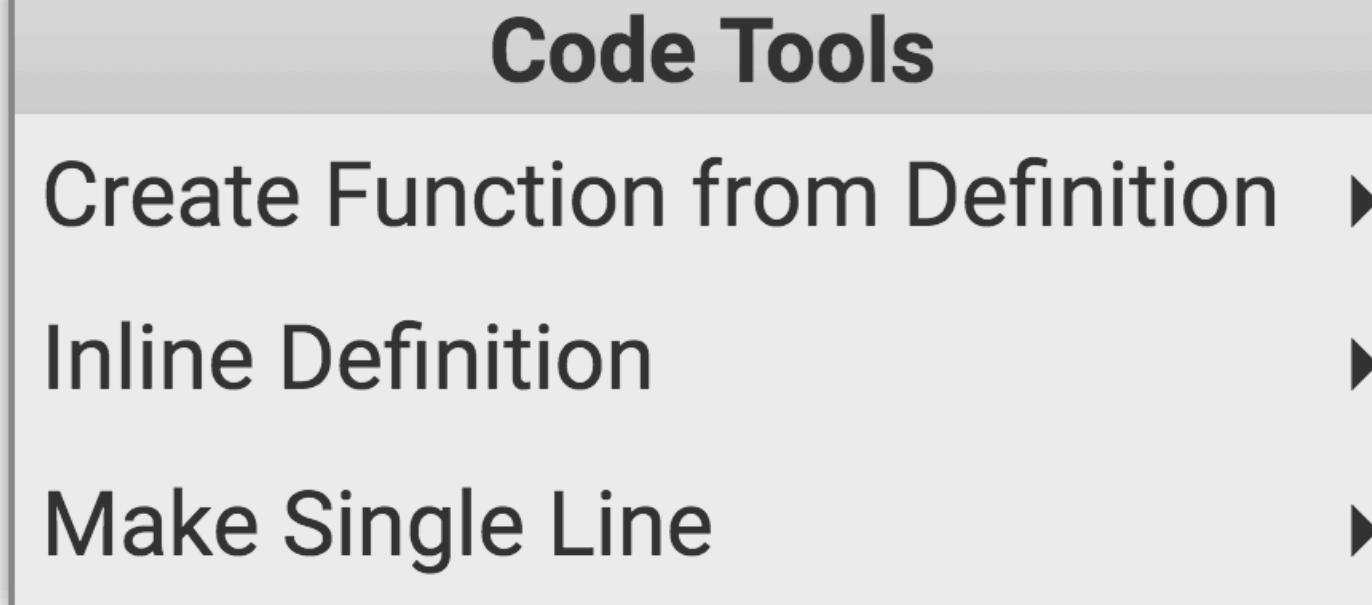
# Deuce

## Structure Select

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200])  
  (let [x y] [100 100])  
  (image "lightgrey" x y width height 15 image_url)))  
  
(def main  
  (draw (concat [ image1 ])))
```

```
(def image1  
  (let [width height]  
    (let [x y] [100 100]  
      (image "lightgrey"
```

## Short Menu



## Defaults

# Structure Select

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200])  
  (let [x y] [100 100])  
  (image "lightgrey" x y width height 15 image_url)))  
  
(def main  
  (draw (concat [ image1 ])))
```

```
(def image1  
  (let [width height]  
    (let [x y] [100 100]  
      (image "lightgrey"
```

# Deuce

## Short Menu

## Defaults

### Code Tools

Create Function from Definition

Abstract image1 over its constants

Inline Definition

Abstract image1 over its named

Make Single Line

constants

### Code Tools

Move Definition

Move width and height

Duplicate Definition

**Deuce more effective  
than Traditional?**

**Deuce more effective  
than Traditional?**

**Deuce preferred  
to Traditional?**

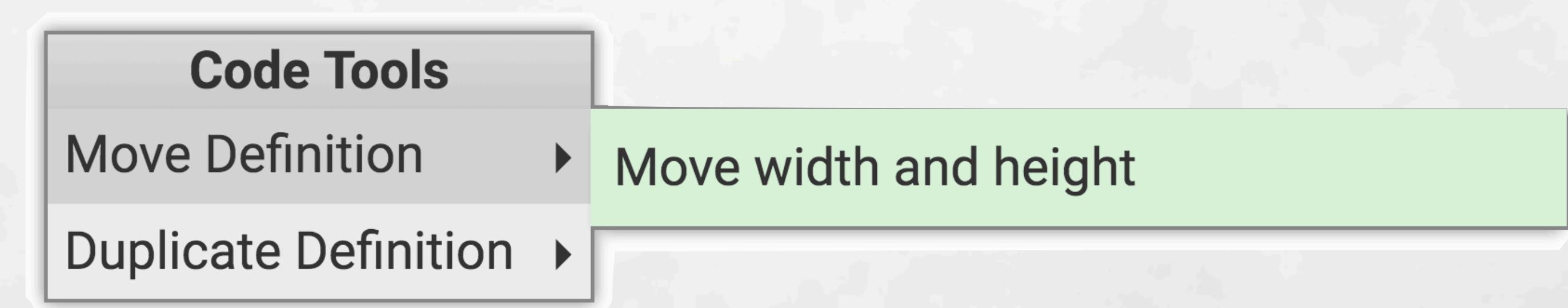
# Deuce

# Deuce

## Structure Select

```
(def image1
  (let [width height]
    (let [x y] [100 100]
      (image "lightgrey"))))
```

## Short Menu



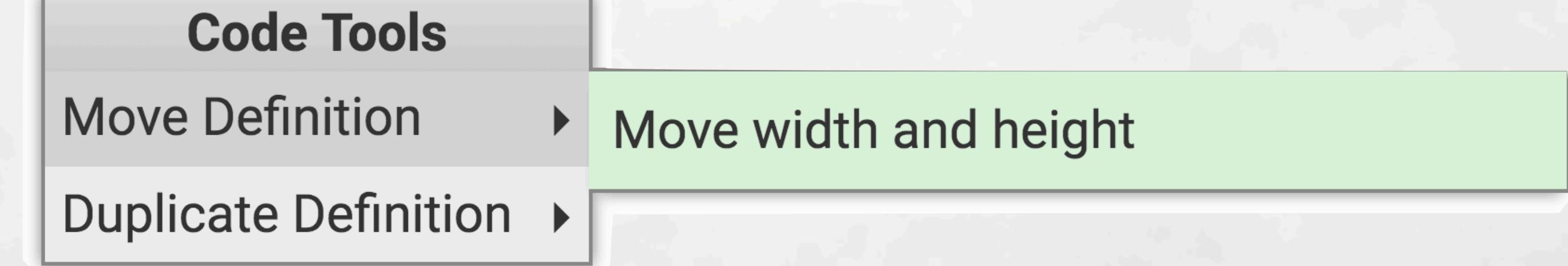
## Defaults

# Deuce “Box-Select Mode”

## Structure Select

```
(def image1
  (let [width height]
    (let [x y] [100 100]
      (image "lightgrey"))))
```

## Short Menu



## Defaults

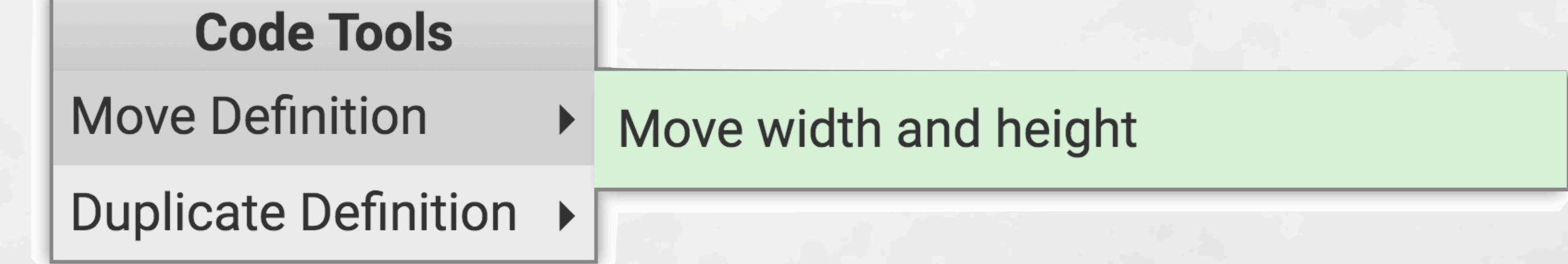
# Traditional

## Deuce “Box-Select Mode”

### Structure Select

```
(def image1
  (let [width height]
    (let [x y] [100 100]
      (image "lightgrey"))))
```

### Short Menu



### Defaults

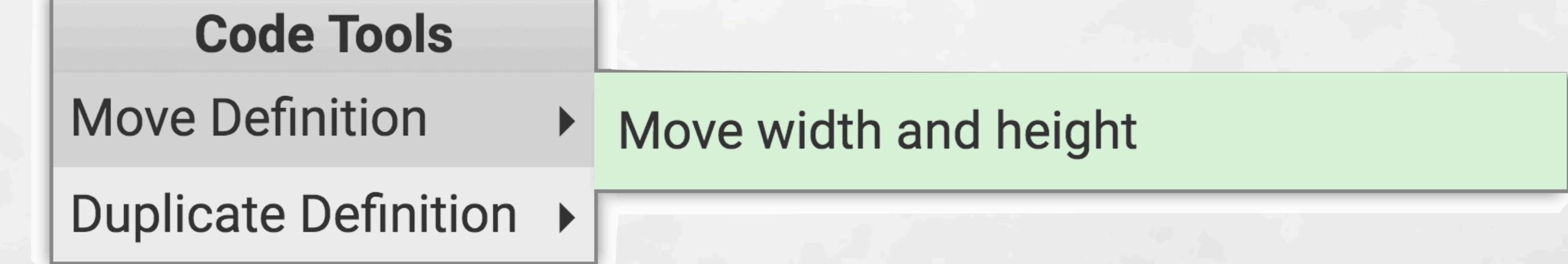
# Traditional “Text-Select Mode”

## Deuce “Box-Select Mode”

### Structure Select

```
(def image1
  (let [width height]
    (let [x y] [100 100]
      (image "lightgrey"))))
```

### Short Menu



### Defaults

# Traditional “Text-Select Mode”

## Text Select

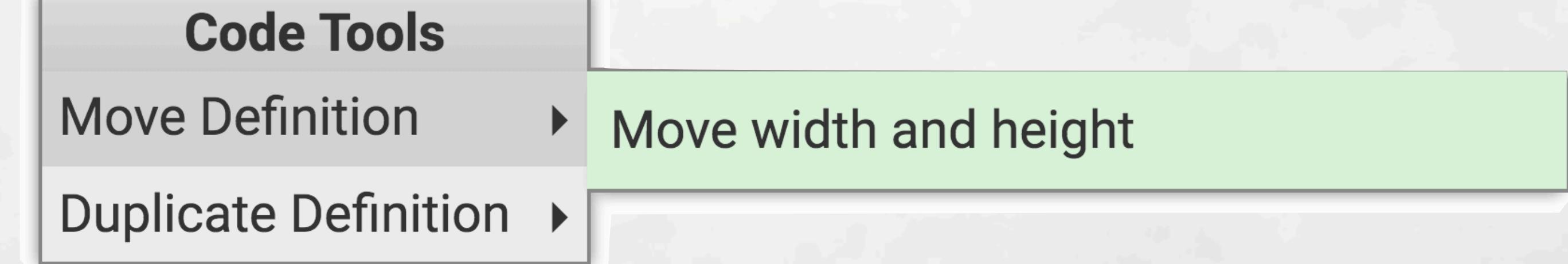
```
(def image1
  (let [width height]
    (let [x y] [100 100]
      (image "lightgrey"))))
```

# Deuce “Box-Select Mode”

## Structure Select

```
(def image1
  (let [width height]
    (let [x y] [100 100]
      (image "lightgrey"))))
```

## Short Menu



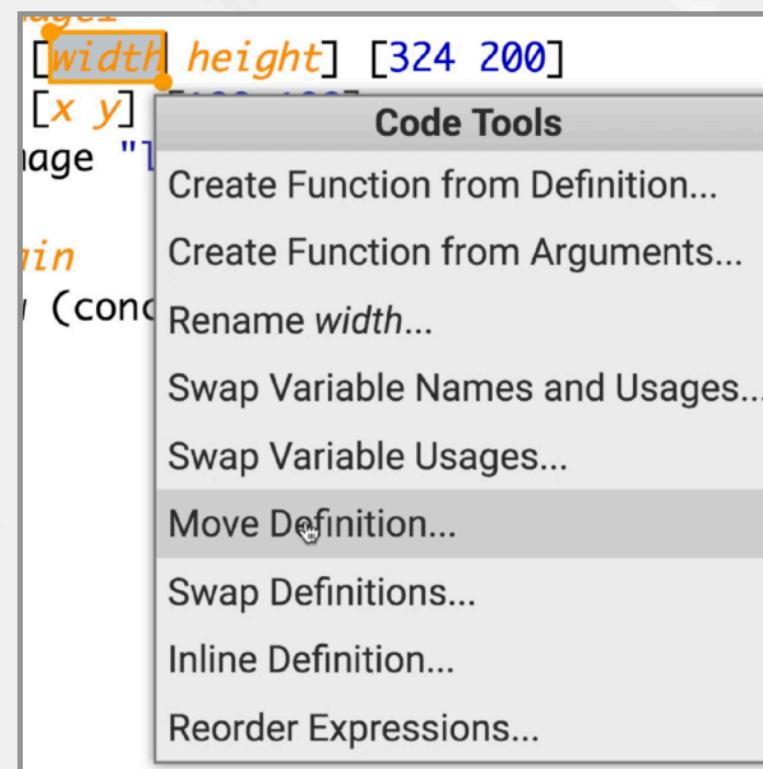
## Defaults

# Traditional “Text-Select Mode”

Text Select

```
(def image1
  (let [width height]
    (let [x y] [100 100]
      (image "lightgrey"))))
```

Right-Click Menu

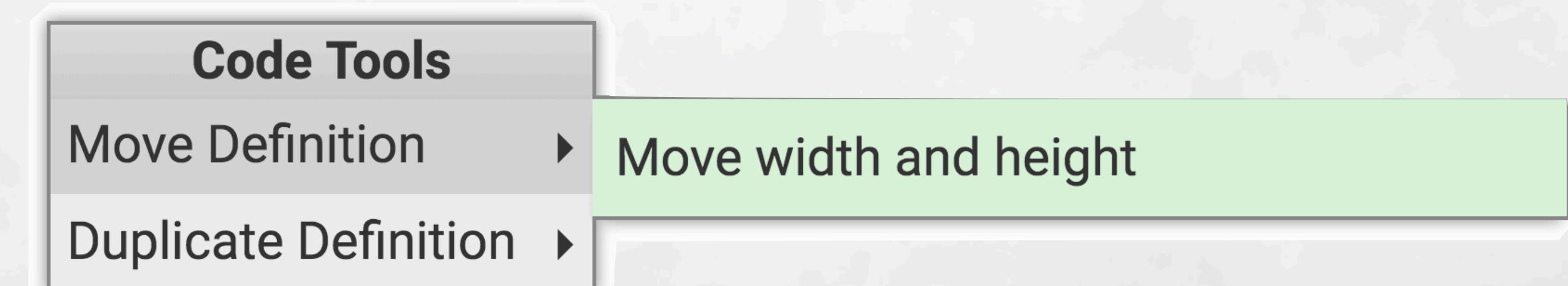


# Deuce “Box-Select Mode”

Structure Select

```
(def image1
  (let [width height]
    (let [x y] [100 100]
      (image "lightgrey"))))
```

Short Menu



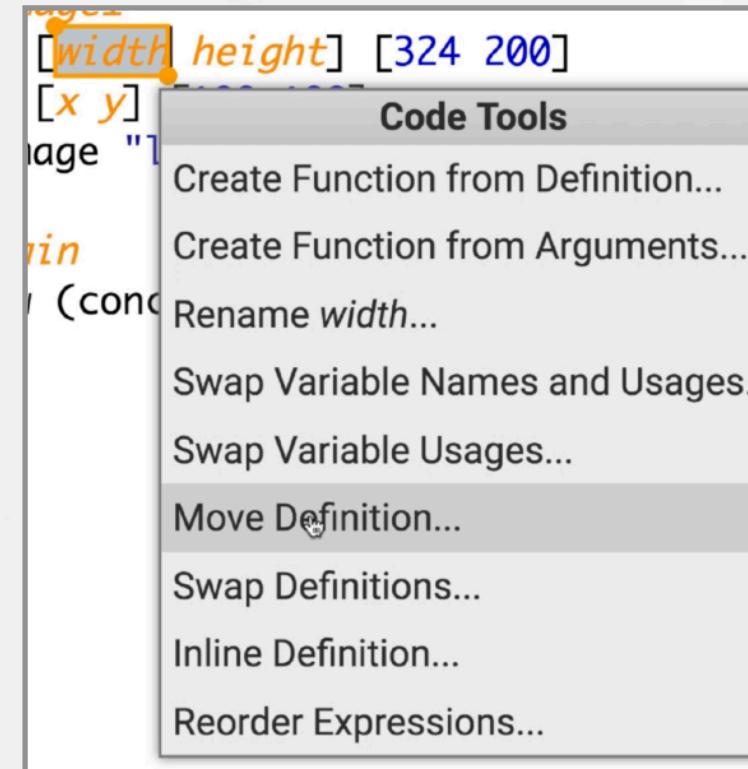
Defaults

# Traditional “Text-Select Mode”

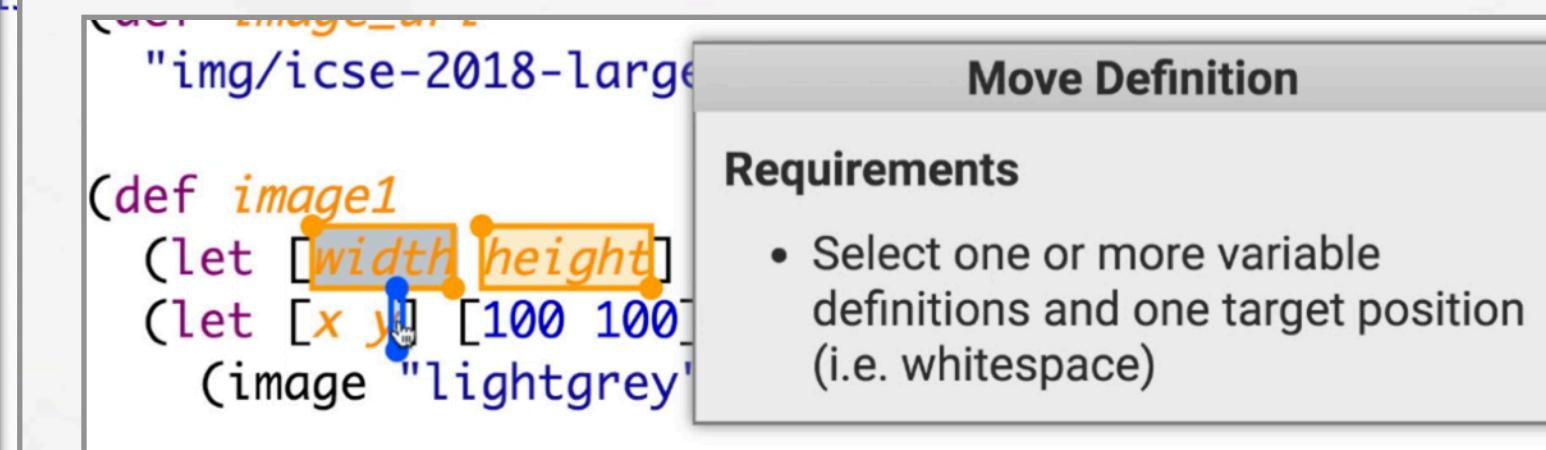
Text Select

```
(def image1
  (let [width height]
    (let [x y] [100 100]
      (image "lightgrey"))))
```

Right-Click Menu



Select Arguments

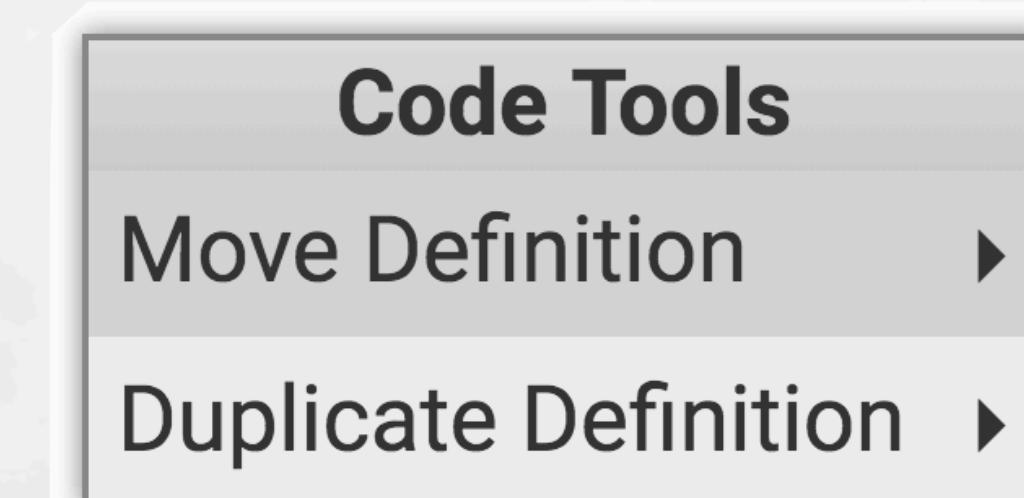


# Deuce “Box-Select Mode”

Structure Select

```
(def image1
  (let [width height]
    (let [x y] [100 100]
      (image "lightgrey"))))
```

Short Menu



Defaults

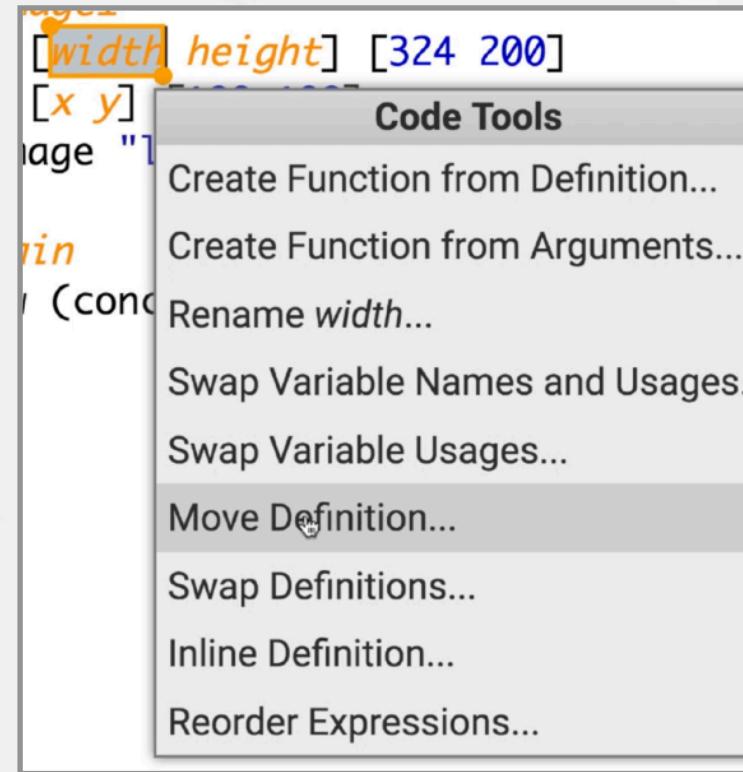
Move width and height

# Traditional “Text-Select Mode”

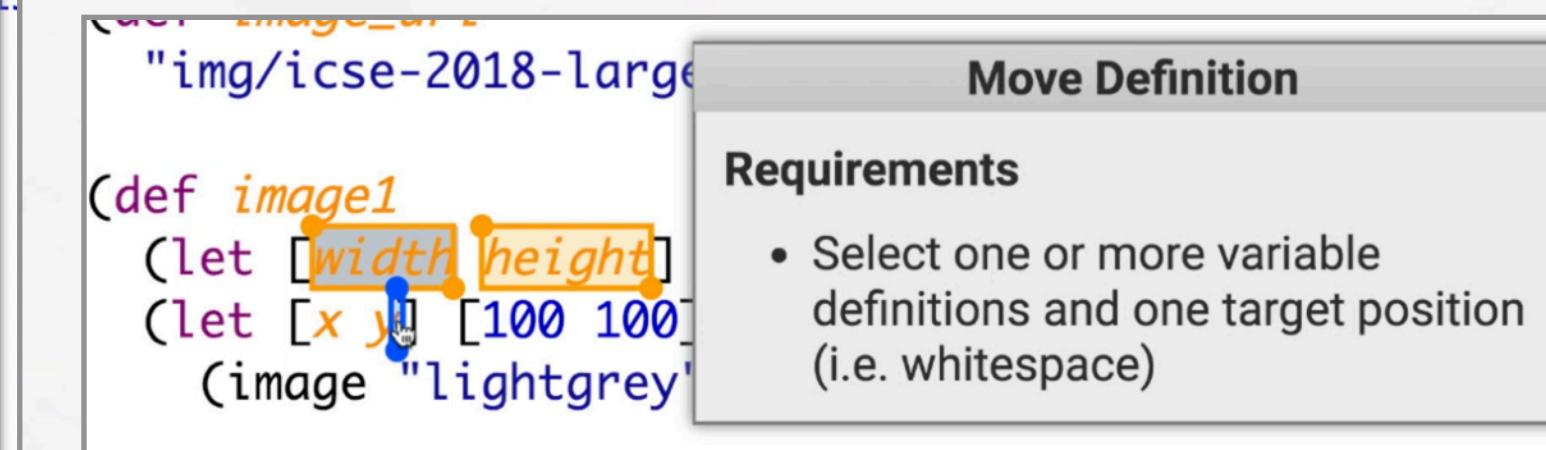
## Text Select

```
(def image1
  (let [width height]
    (let [x y] [100 100]
      (image "lightgrey"))))
```

## Right-Click Menu



## Select Arguments



## Defaults

**Move Definition**

**Requirements**

- Select one or more variable definitions and one target position (i.e. whitespace) (Satisfied)

**Code Updates**

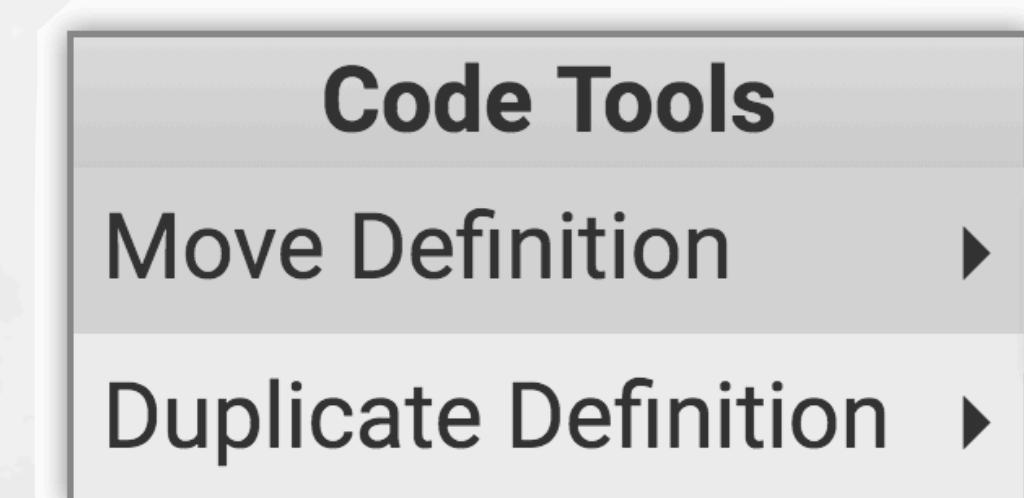
Move width and height

# Deuce “Box-Select Mode”

## Structure Select

```
(def image1
  (let [width height]
    (let [x y] [100 100]
      (image "lightgrey"))))
```

## Short Menu



## Defaults

Move width and height

# Tutorial

# Tutorial

## Head-to-Head Tasks (2x; once per mode)

The figure displays four separate windows, each representing a task in a head-to-head competition. Each window contains a code editor on the left and a canvas on the right.

- Task 1 (Left):** The code defines a rectangle with width 20 and height 250. The goal is to draw a black rectangle of width 20 and height 250 pixels.
- Task 2 (Middle Left):** The code defines a function `connectedCircles` that takes `startX`, `endX`, `startY`, and `endY`. The goal is to draw two gray circles connected by a line.
- Task 3 (Middle Right):** The code defines rectangles `rect1` and `rect2` at different coordinates. The goal is to draw three rectangles: a red one at (30, 30, 50, 70), a green one at (109, 53, 50, 70), and a blue one at (192, 35, 50, 70).
- Task 4 (Right):** The code defines a ring function. The goal is to draw a target consisting of three concentric rings: a red outer ring, a green middle ring, and a blue inner ring.

Each task includes specific instructions and goals for the user to follow.

# Tutorial

## Head-to-Head Tasks (2x; once per mode)

The screenshots show four separate tasks, each with its own code editor and target image.

- Task 1:** A black rectangle. The code defines a rectangle with width 20 and height 250, filled with black.
- Task 2:** Two connected circles. The code defines a function `connectedCircles` that creates two gray circles connected by a line. It then calls `draw` with the result.
- Task 3:** Three colored rectangles (red, green, blue). The code defines three rectangles of different colors and widths, then draws them.
- Task 4:** A target icon. The code defines a ring with an outer radius of 150 and an inner radius of 100, filled with redbrick and lightgray.

## Mix & Match Tasks (free to use both modes)

The screenshots show two separate tasks, each with its own code editor and target image.

- Task 1:** A 2x2 grid of colored squares (black, yellowgreen, gold, royalblue, salmon). The code defines a helper function `oneCorner` and uses it to create four squares in a 2x2 grid.
- Task 2:** A logo icon consisting of a black rectangle with white diagonal lines forming an 'X' pattern. The code defines a rectangle and two lines to create this shape.

# Tutorial

## Head-to-Head Tasks (2x; once per mode)

The figure displays four separate windows of a drawing application, each containing a code editor and a preview window showing a target image. The tasks are as follows:

- Task 1 (Left):** A black rectangle. The code defines a rectangle with width 20 and height 250, filled with black.
- Task 2 (Second from Left):** Two connected circles. The code defines a function `connectedCircles` that creates two circles connected by a line. It uses `cx`, `cy`, `radius`, and `strokeWidth` arguments.
- Task 3 (Second from Right):** Three colored rectangles (red, green, blue) arranged in a triangle. The code defines a function `rects` that creates three rectangles at specific coordinates with different colors and widths.
- Task 4 (Right):** A target icon consisting of concentric circles. The code defines a function `ring` that generates concentric rings of a specified color and width.

## Mix & Match Tasks (free to use both modes)

The figure displays two separate windows of a drawing application, each containing a code editor and a preview window showing a target image. The tasks are as follows:

- Task 1 (Left):** A 2x2 grid of colored squares (yellowgreen, gold, royalblue, salmon). The code defines a function `fourSquares` that creates a 2x2 grid of squares with specific colors and widths.
- Task 2 (Right):** A logo icon resembling a stylized 'X' or a square with diagonal lines. The code defines a function `logo` that creates a logo icon using lines and rectangles.

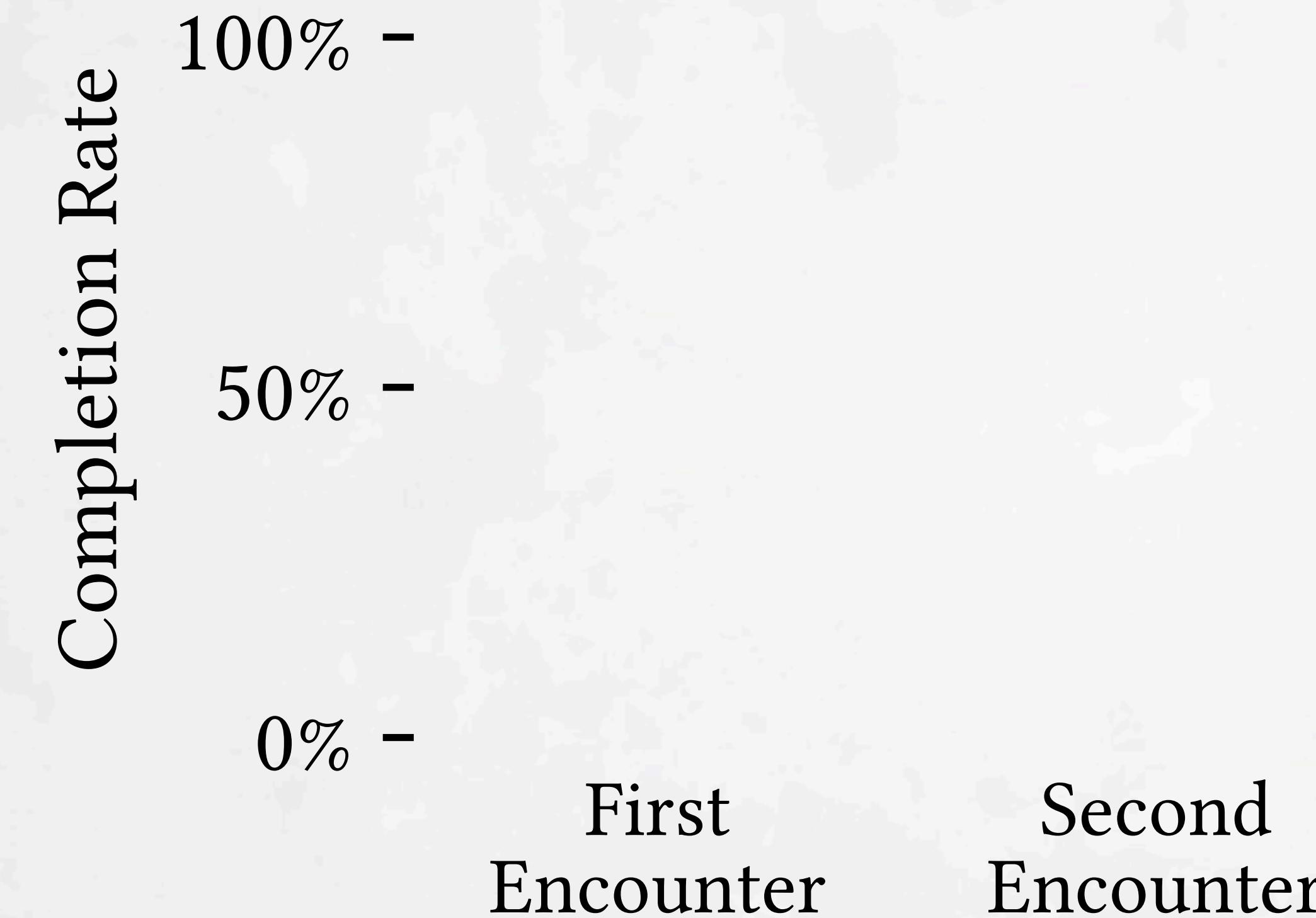
## Exit Survey

**Deuce** more effective than **Traditional**?

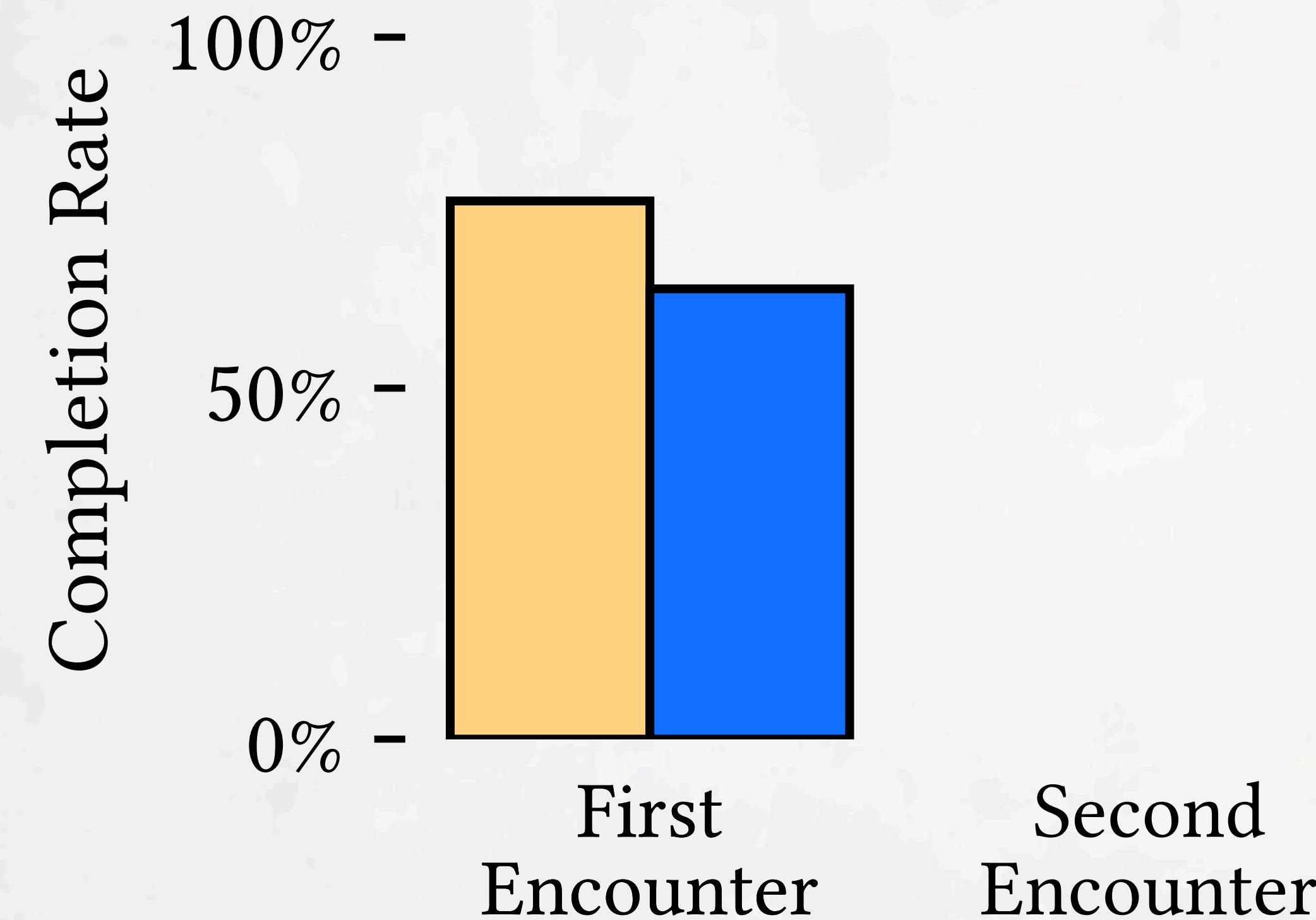
# Deuce more effective than Traditional?



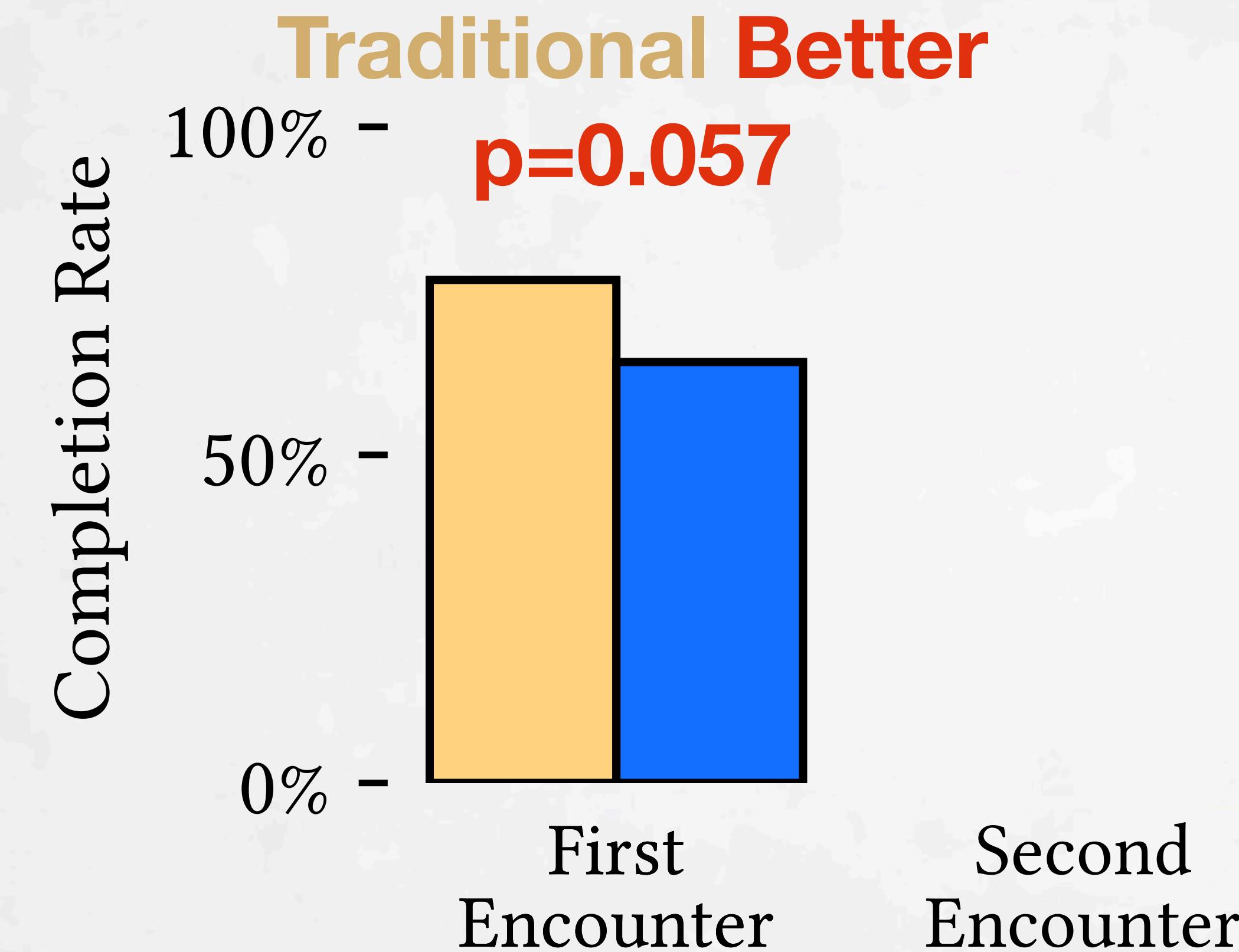
# Deuce more effective than Traditional?



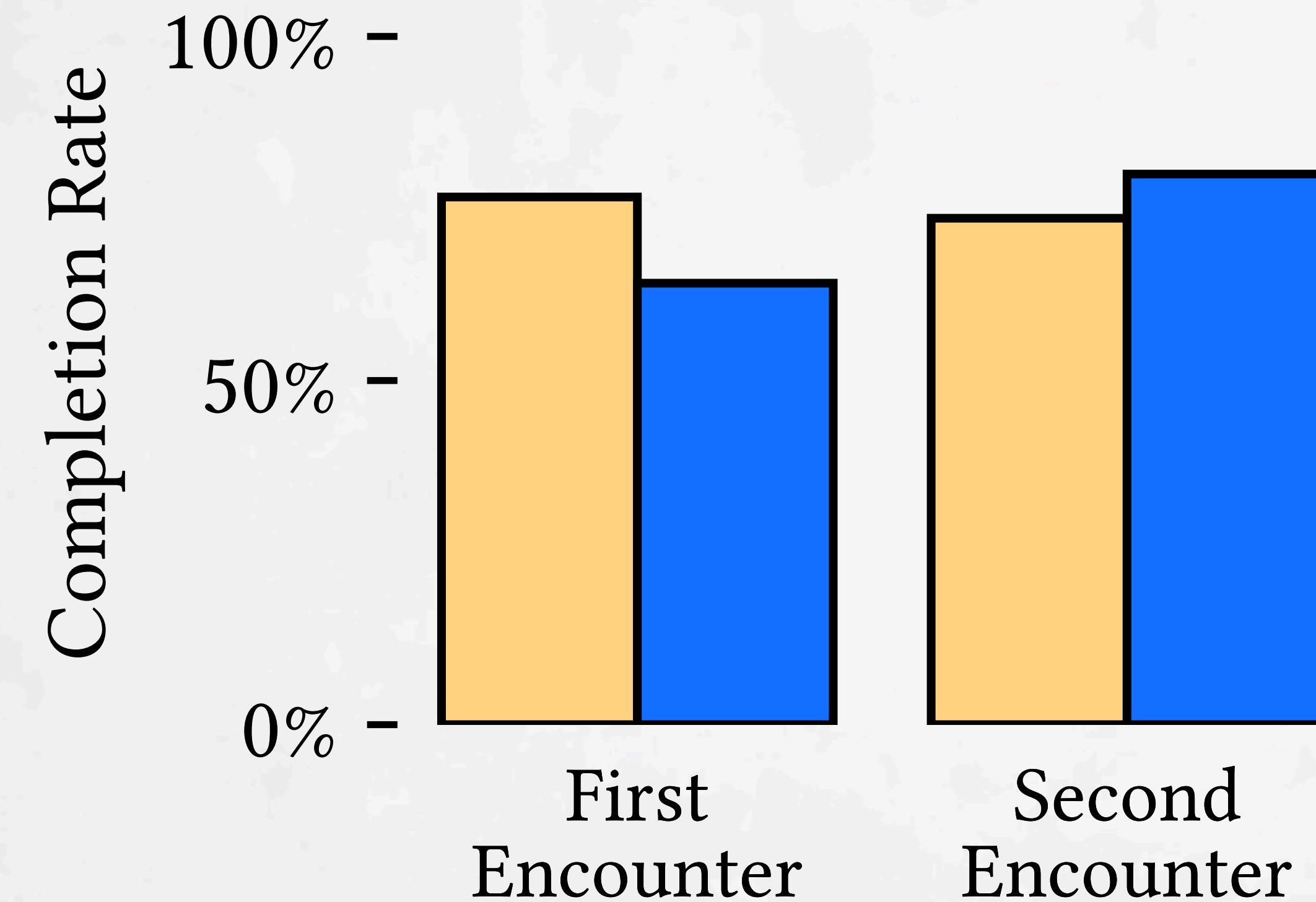
# Deuce more effective than Traditional?



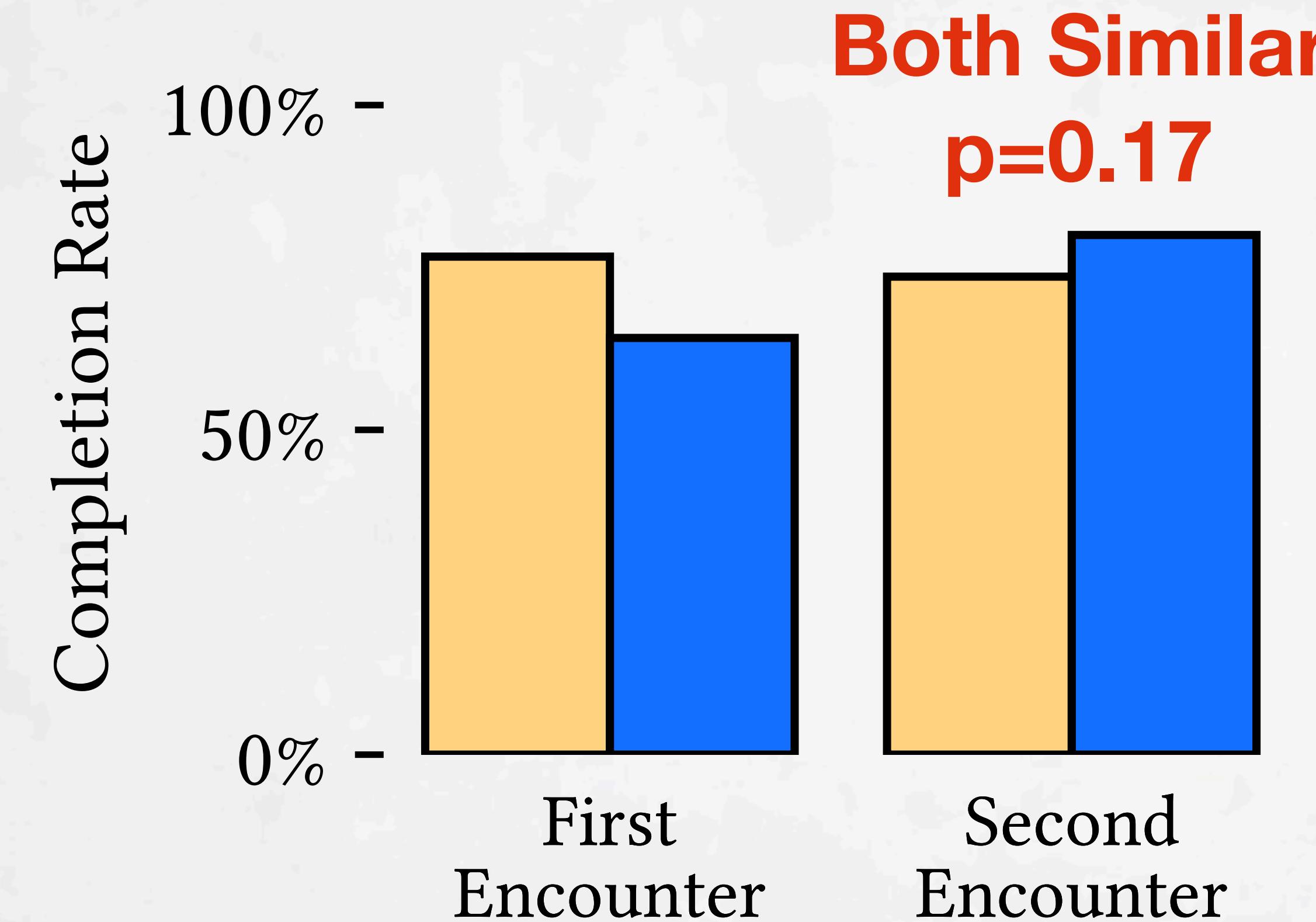
# Deuce more effective than Traditional?



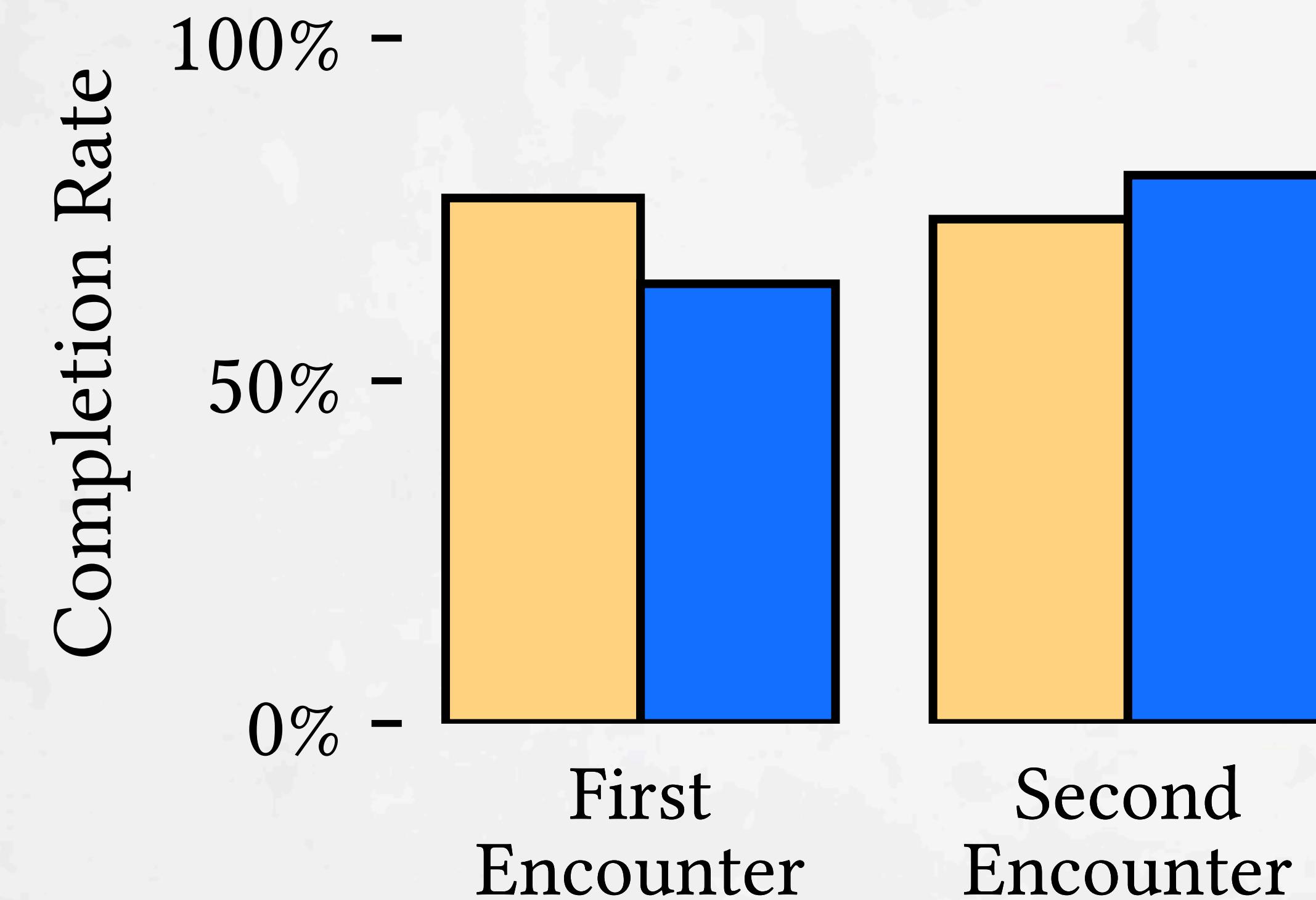
# Deuce more effective than Traditional?



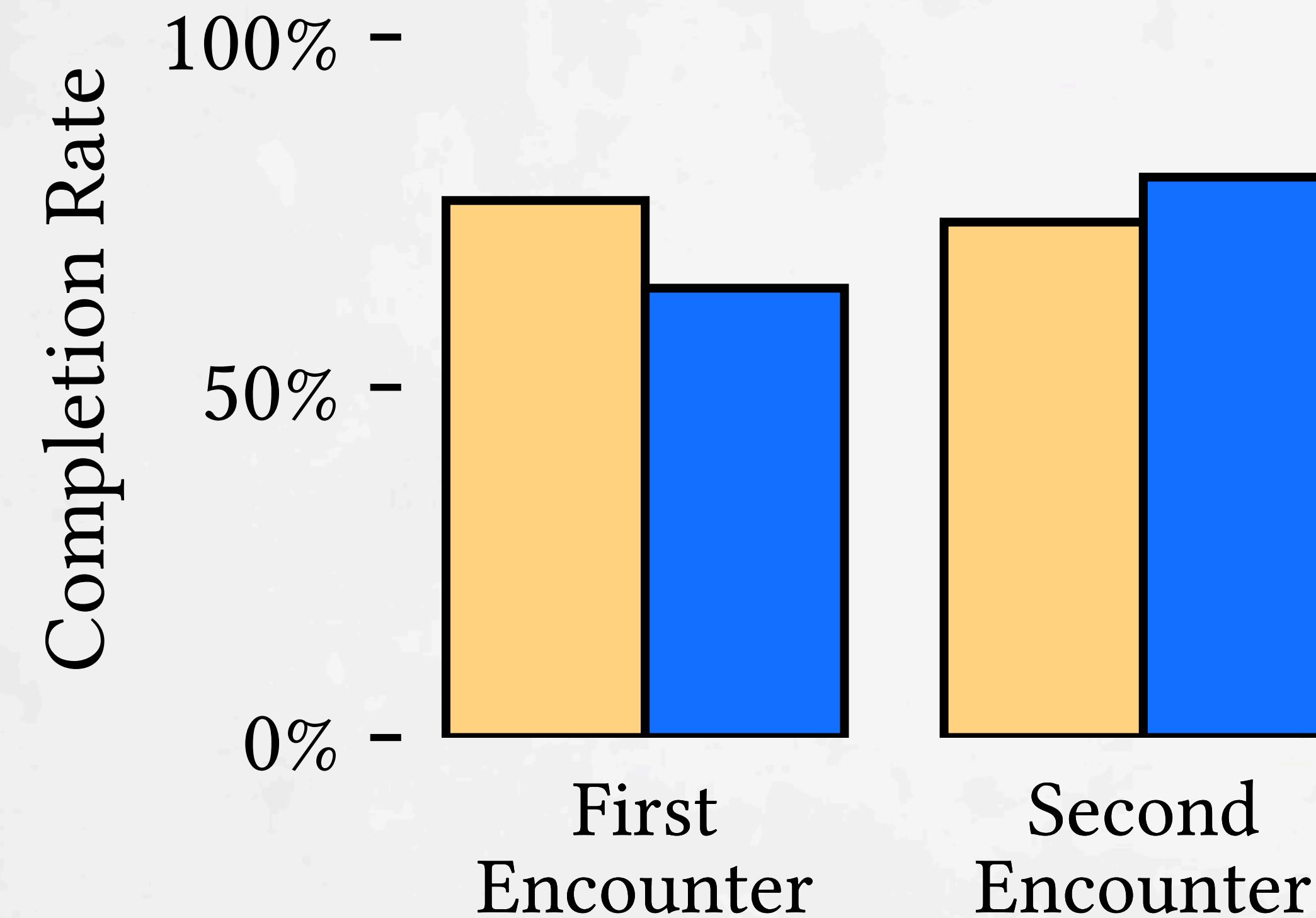
# Deuce more effective than Traditional?



# Deuce more effective than Traditional?



# Deuce more effective than Traditional?

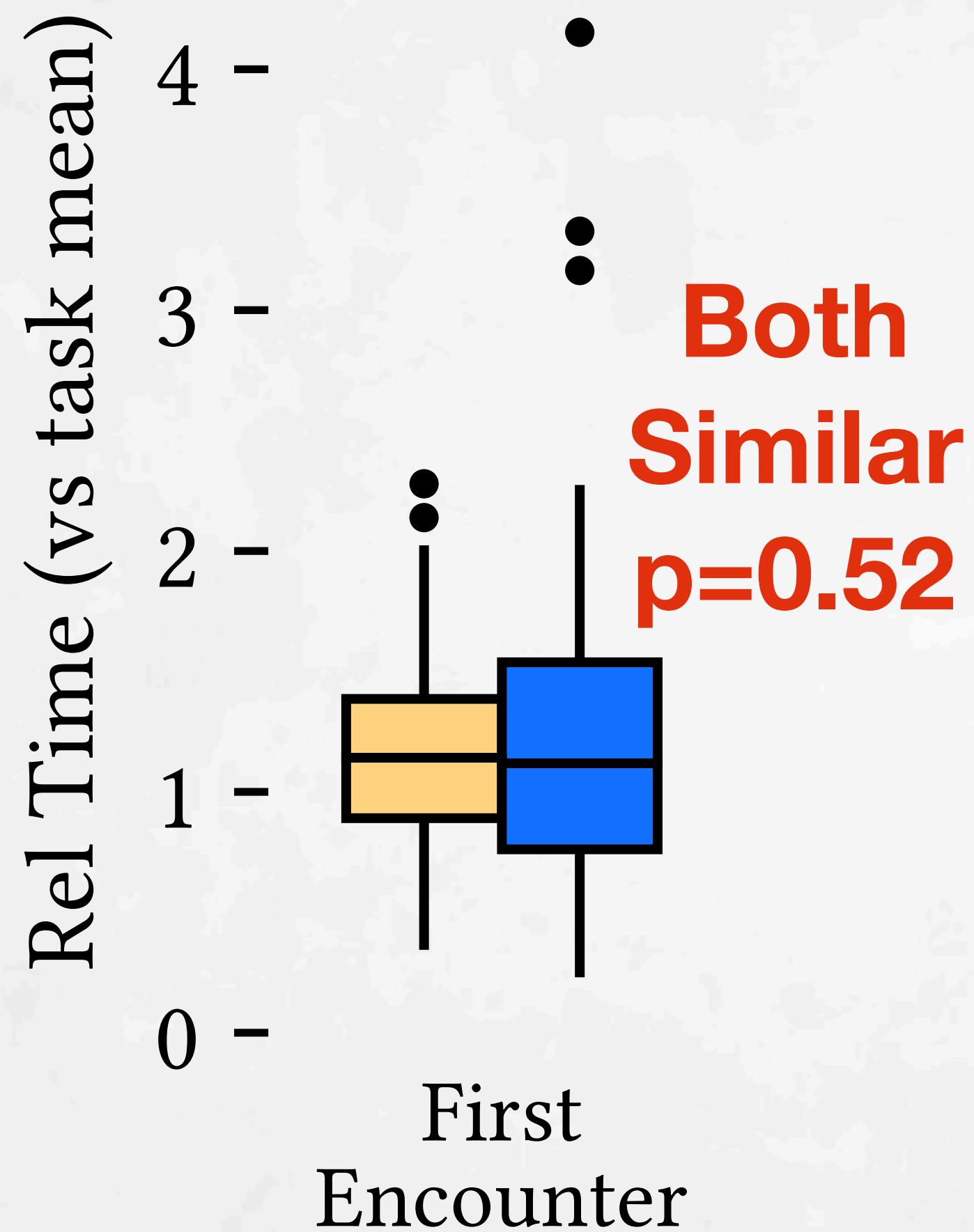


**Deuce** doesn't help discoverability

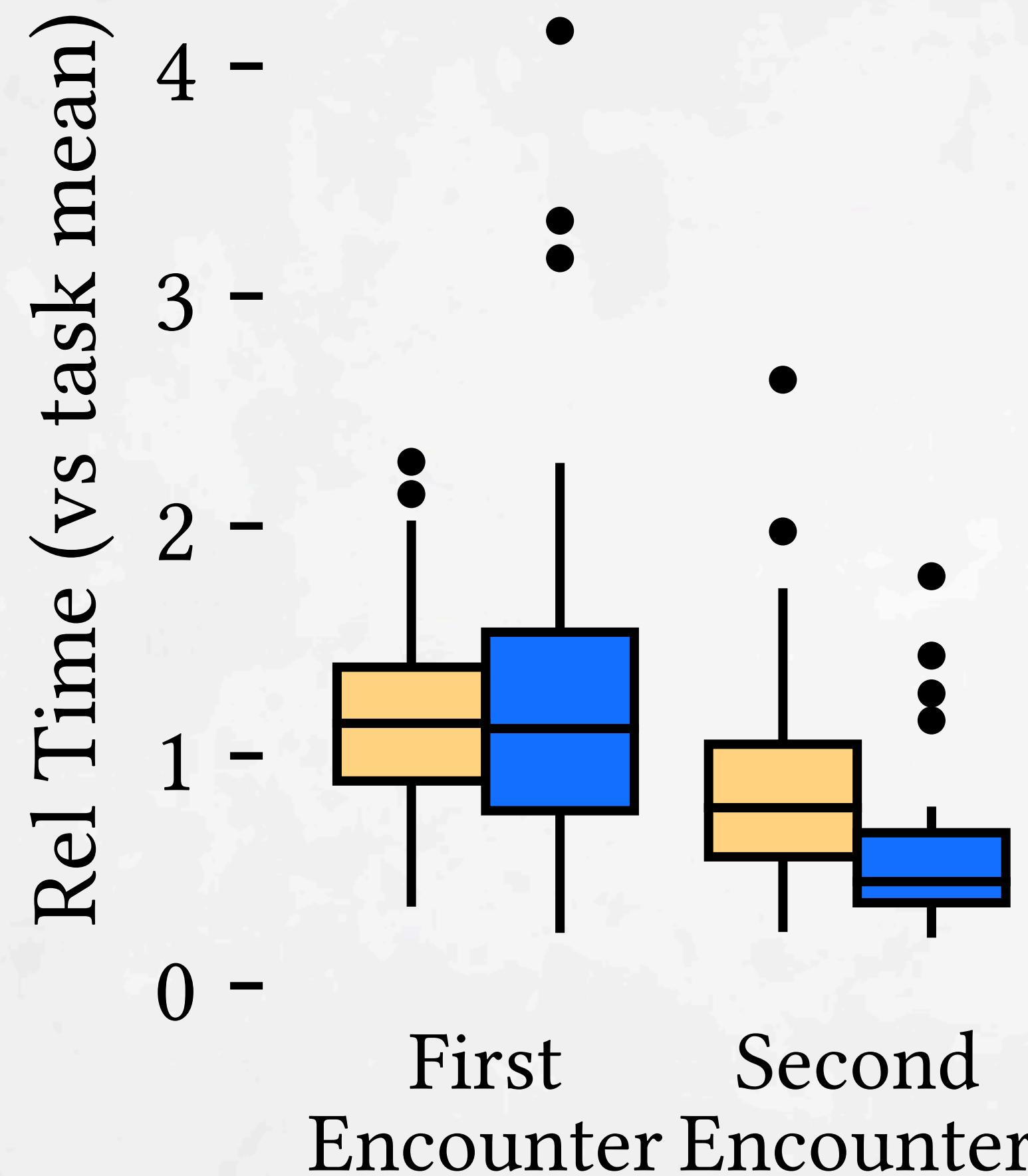
# Deuce more effective than Traditional?



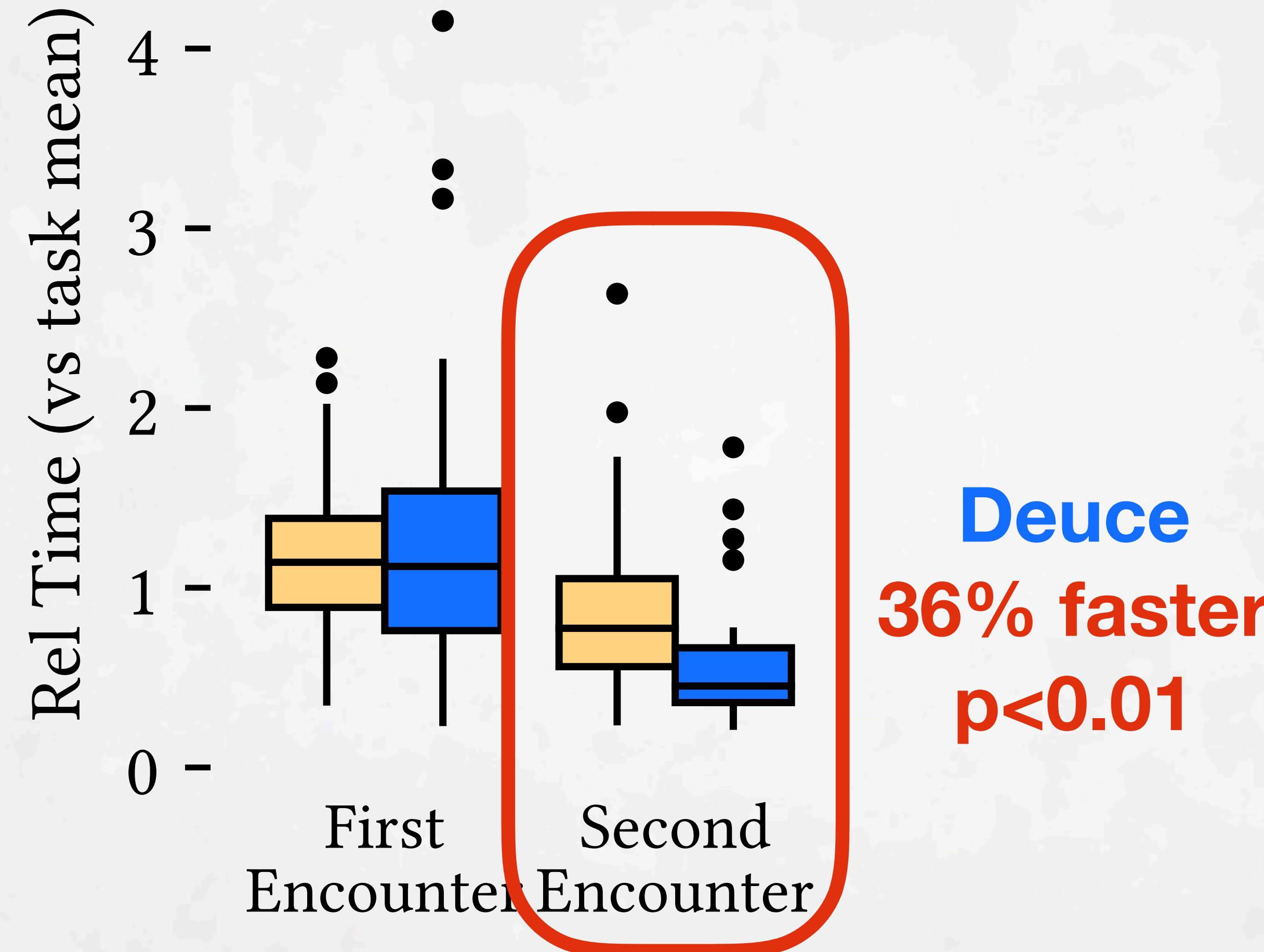
# Deuce more effective than Traditional?



# Deuce more effective than Traditional?

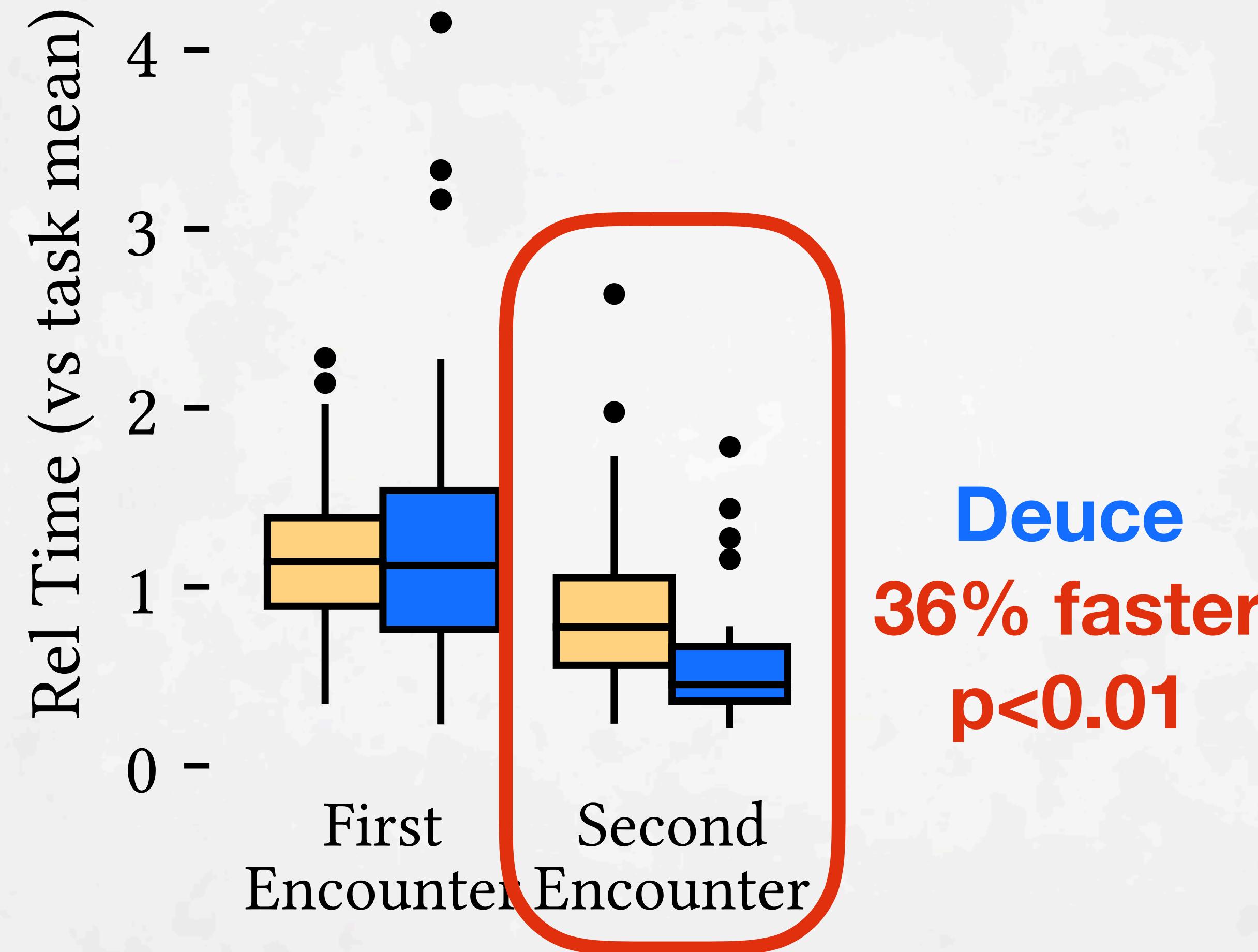


# Deuce more effective than Traditional?



Deuce  
36% faster  
 $p < 0.01$

# Deuce more effective than Traditional?



Deuce may be faster once learned

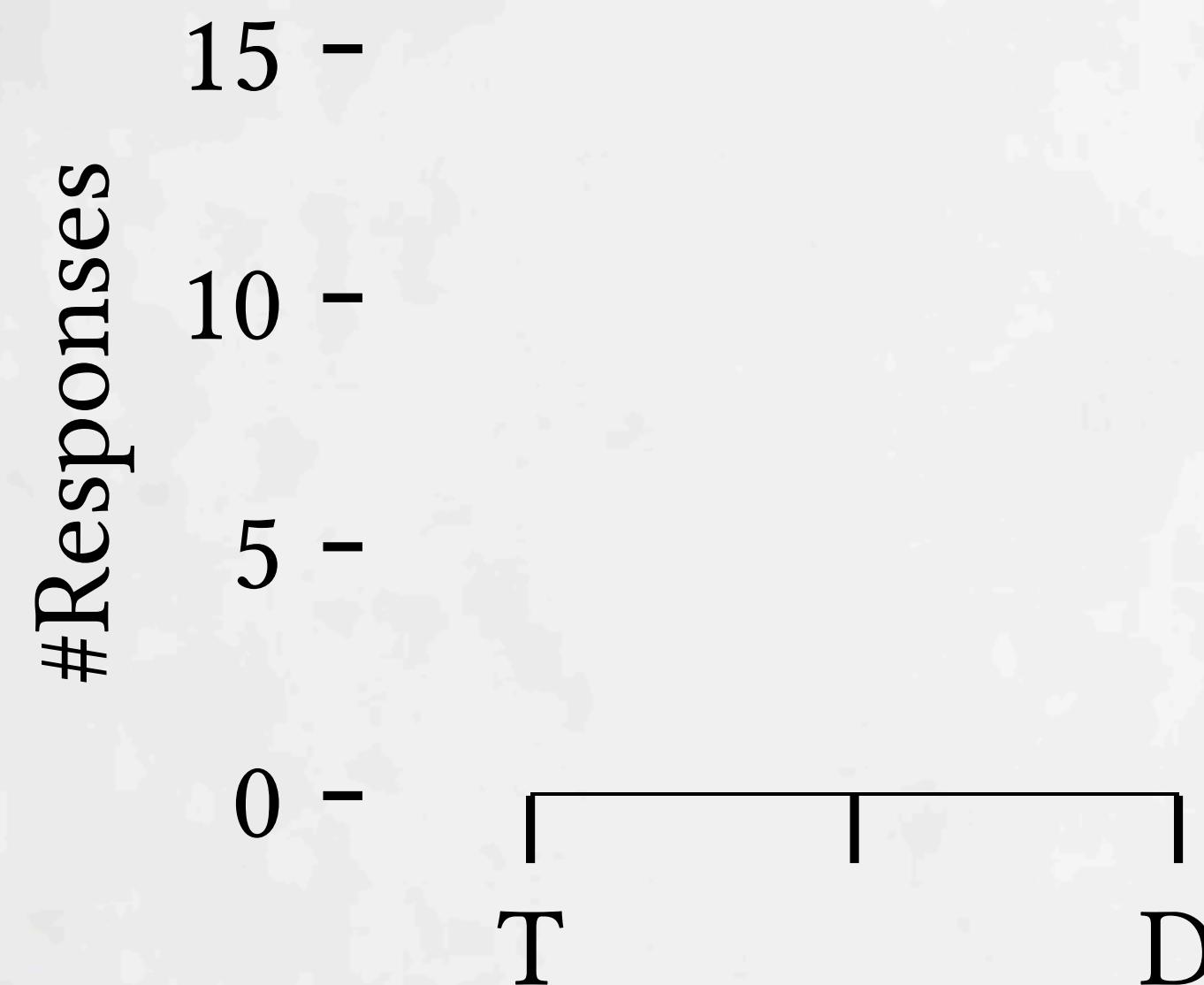
# Deuce preferred to Traditional?

# Deuce preferred to Traditional?

Survey

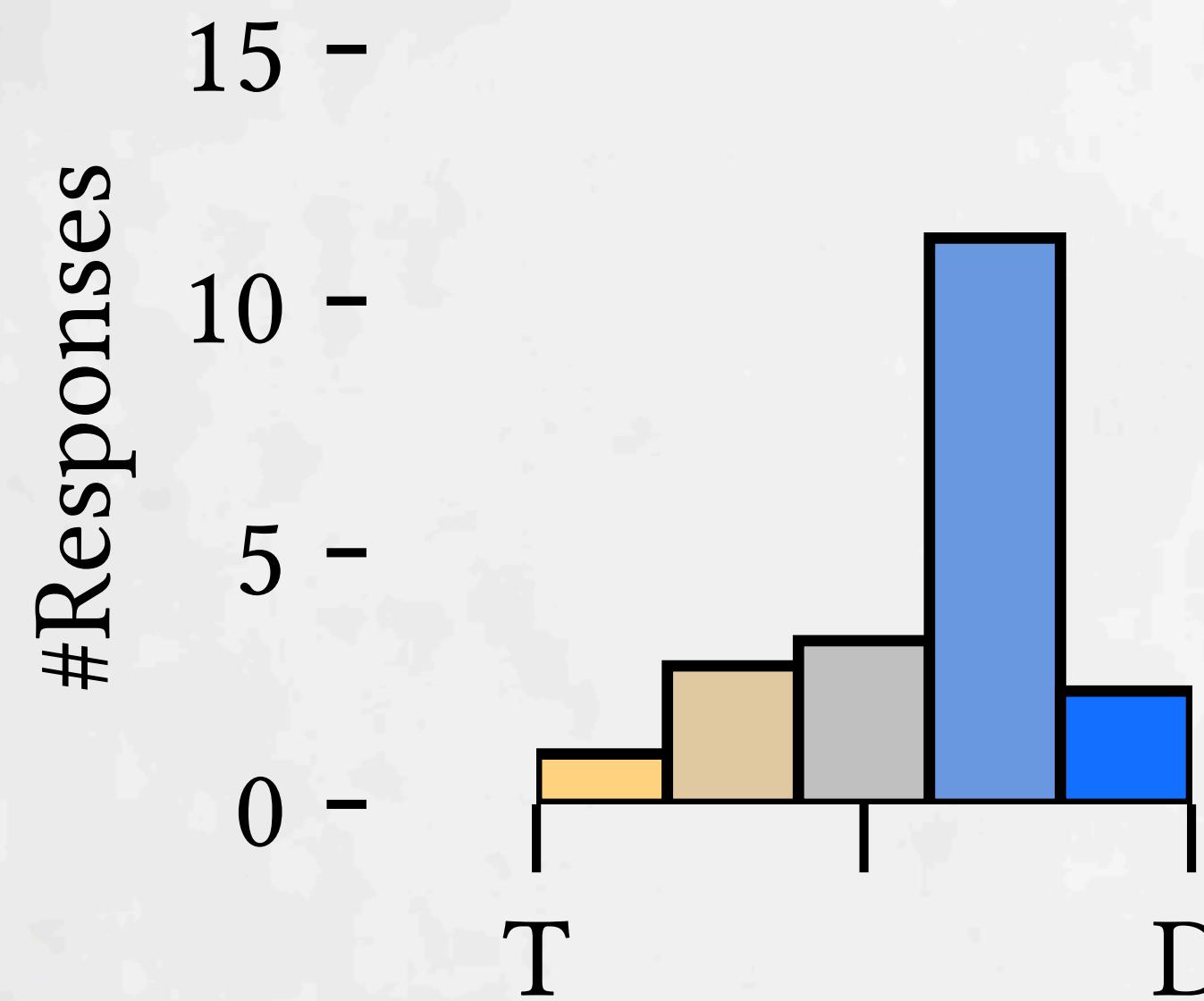
# Deuce preferred to Traditional?

Survey



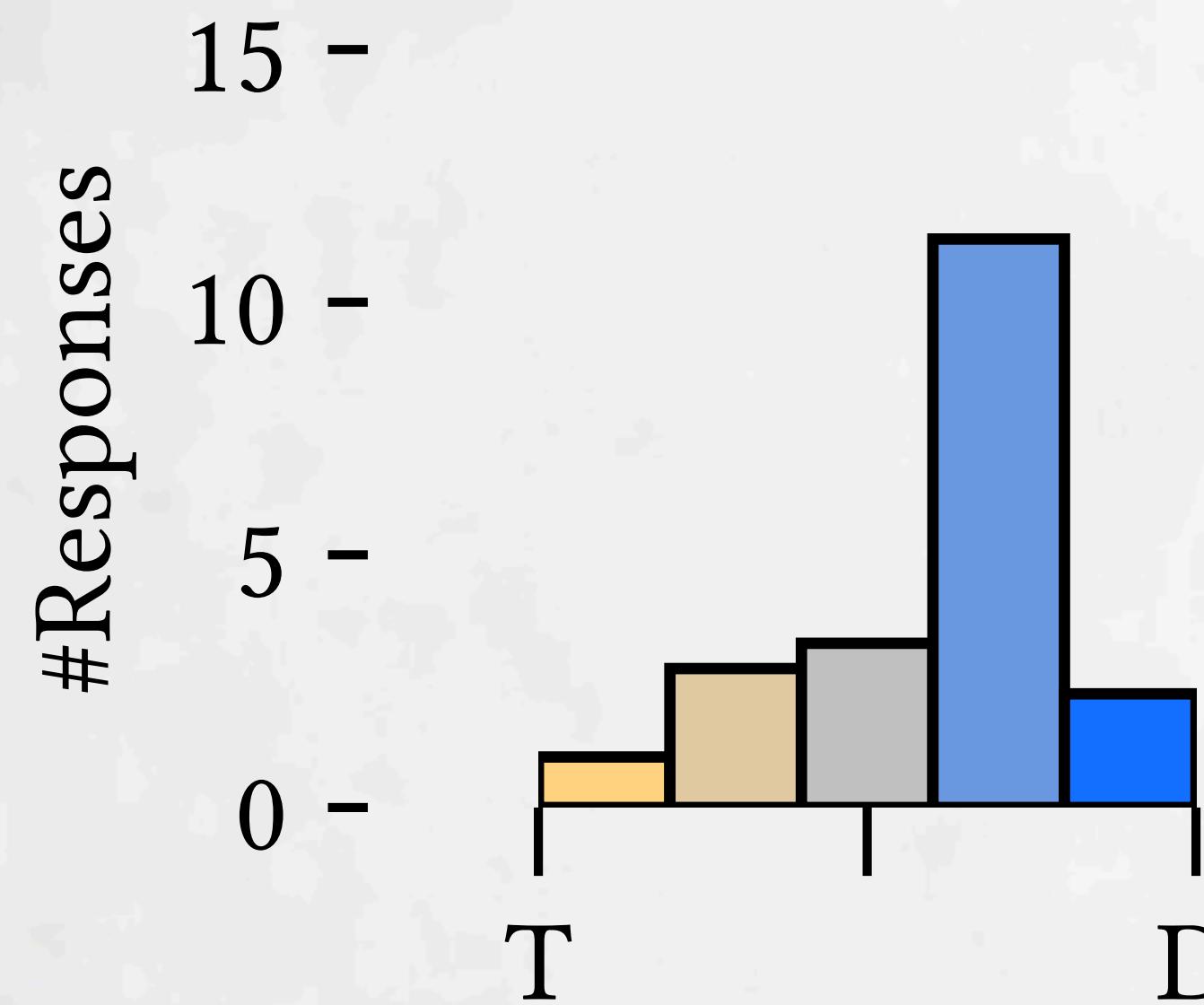
# Deuce preferred to Traditional?

Survey



# Deuce preferred to Traditional?

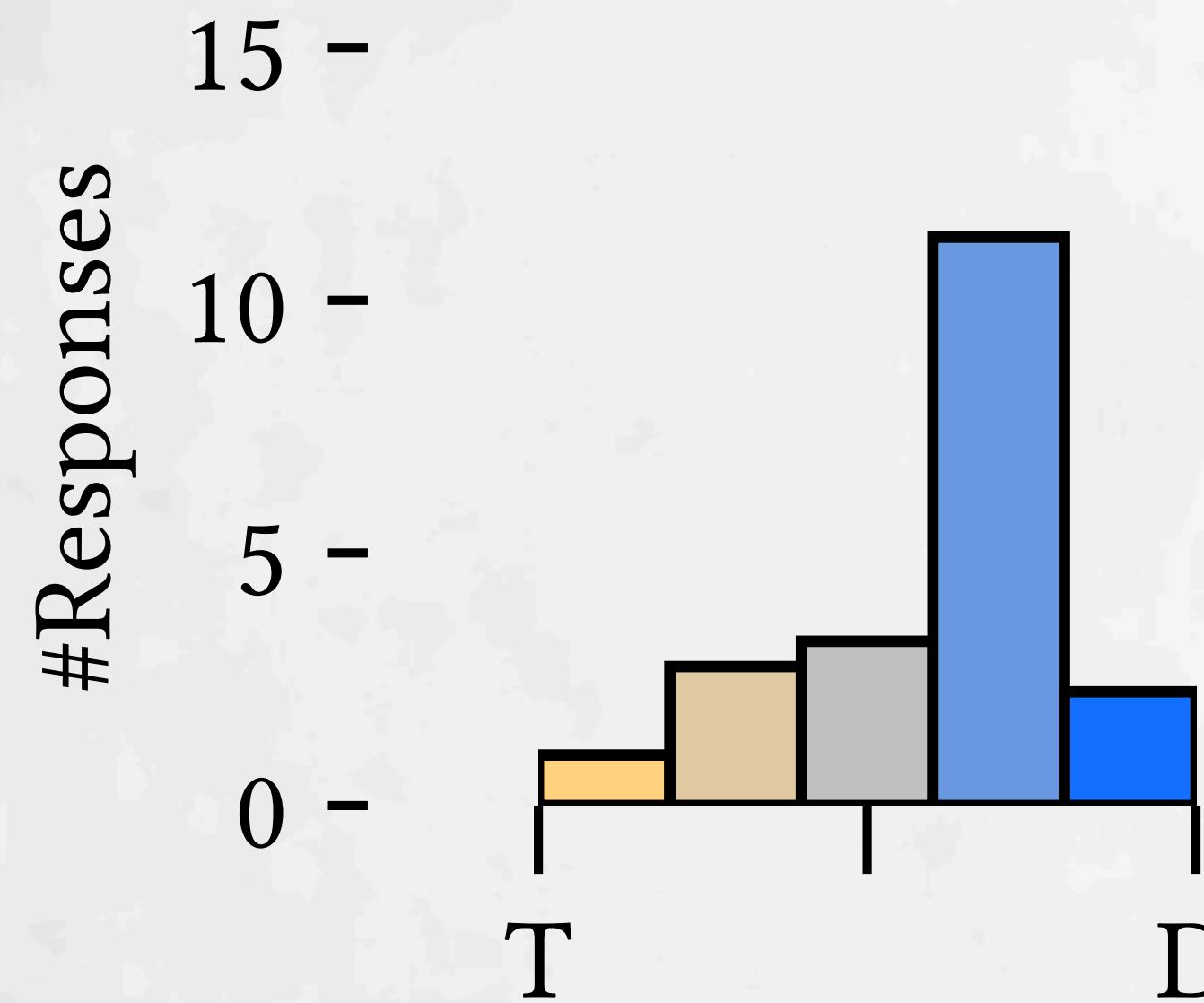
Survey



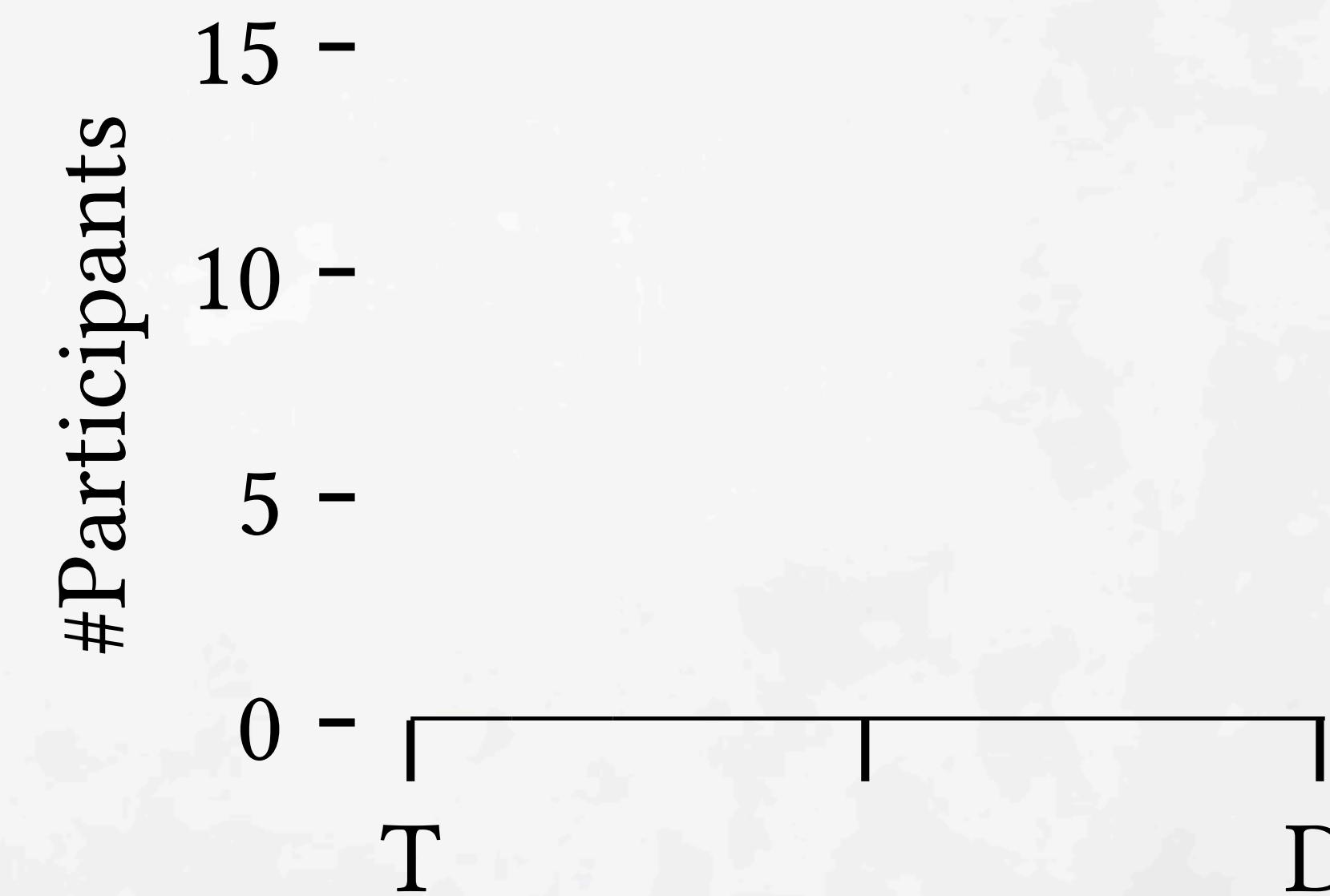
Modest subjective  
preference for Deuce

# Deuce preferred to Traditional?

Survey



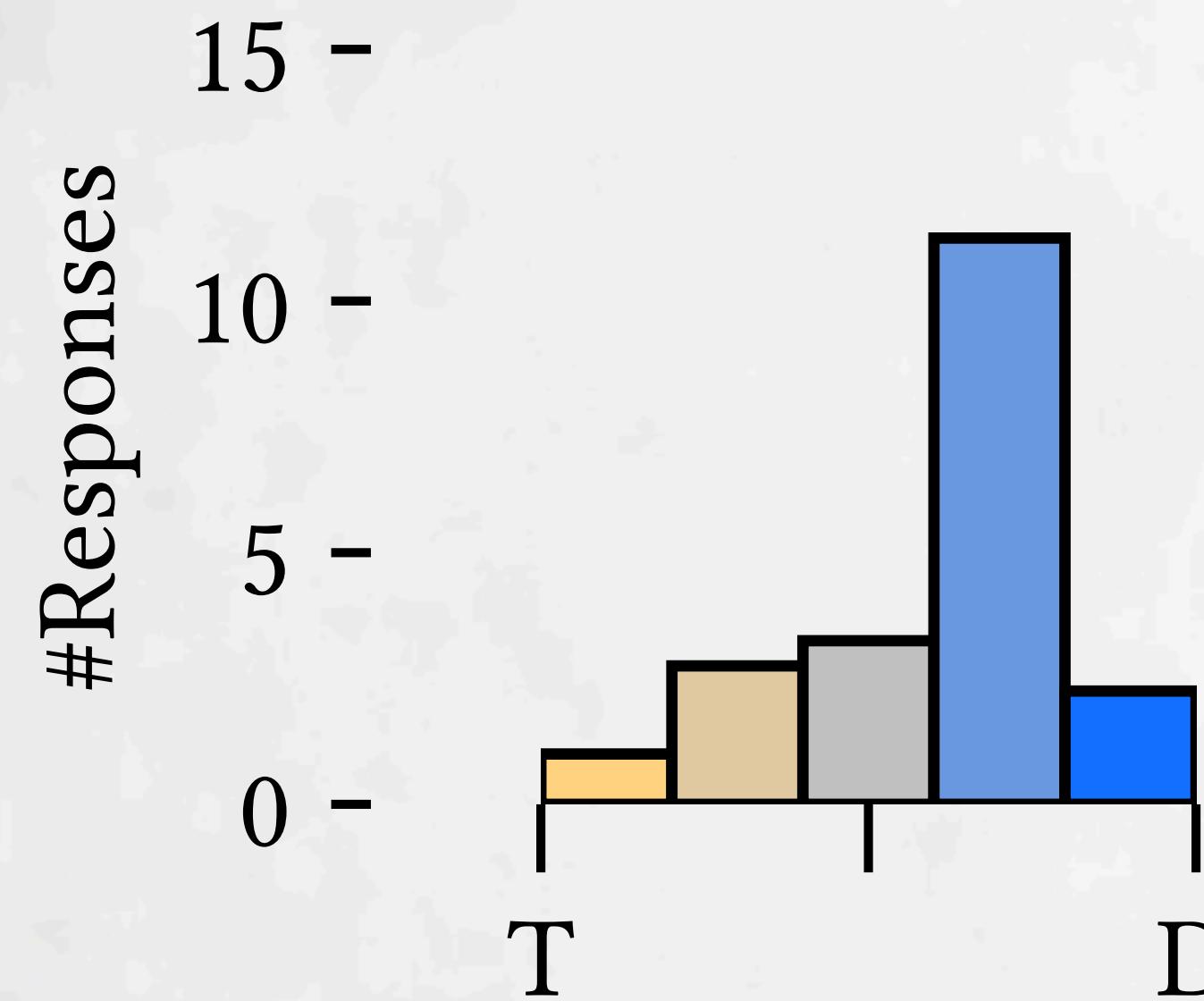
Observed



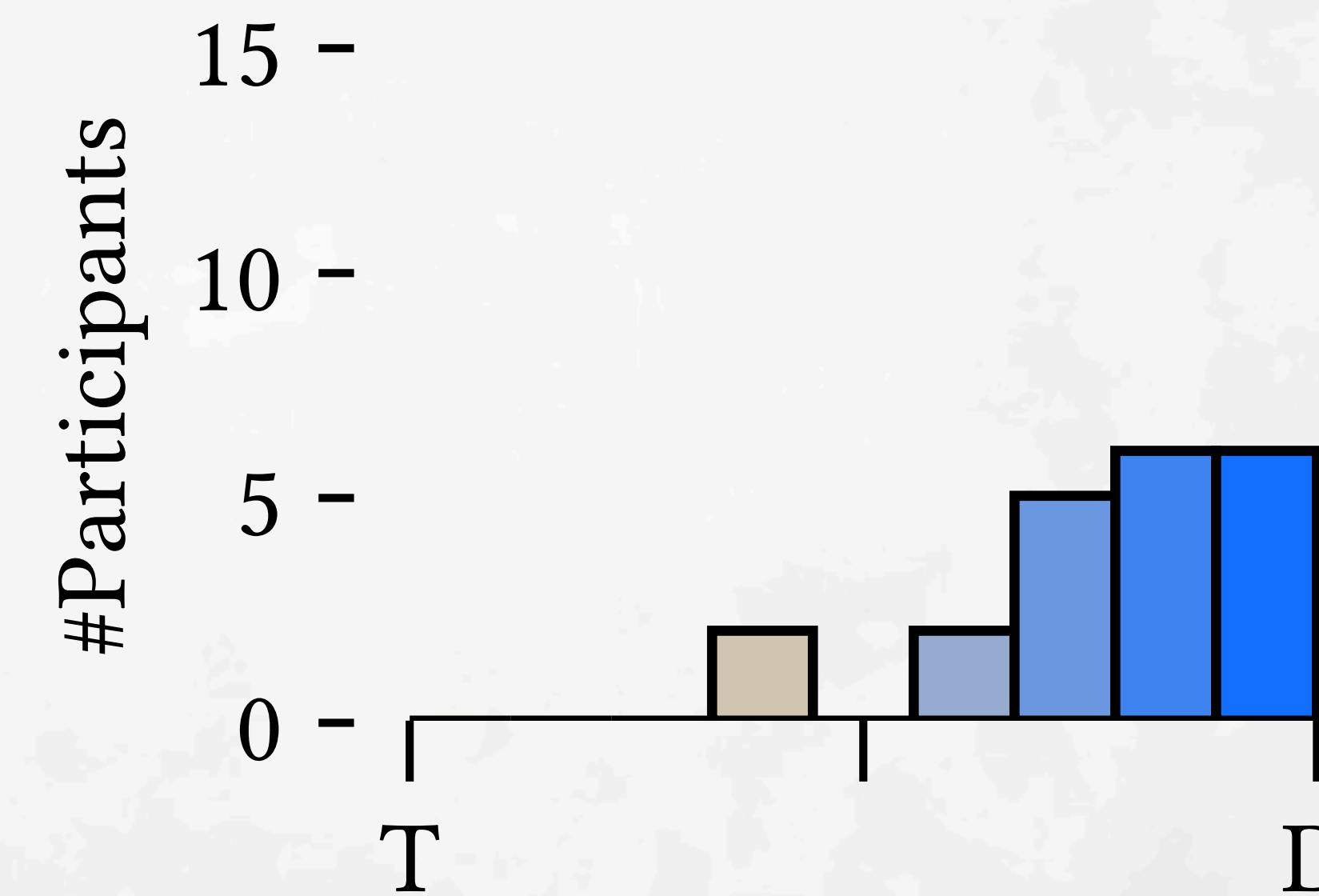
Modest subjective  
preference for Deuce

# Deuce preferred to Traditional?

Survey



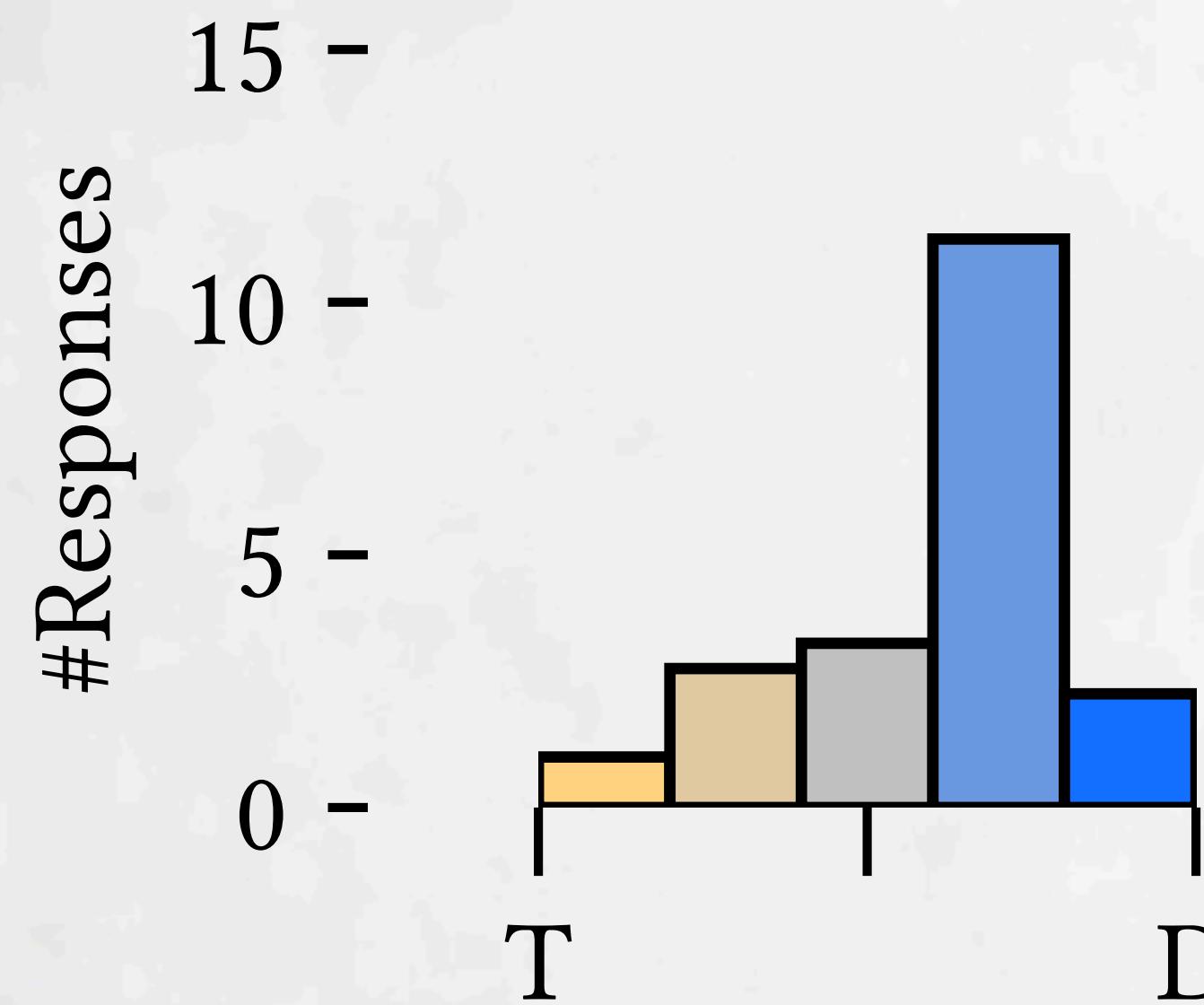
Observed



Modest subjective  
preference for Deuce

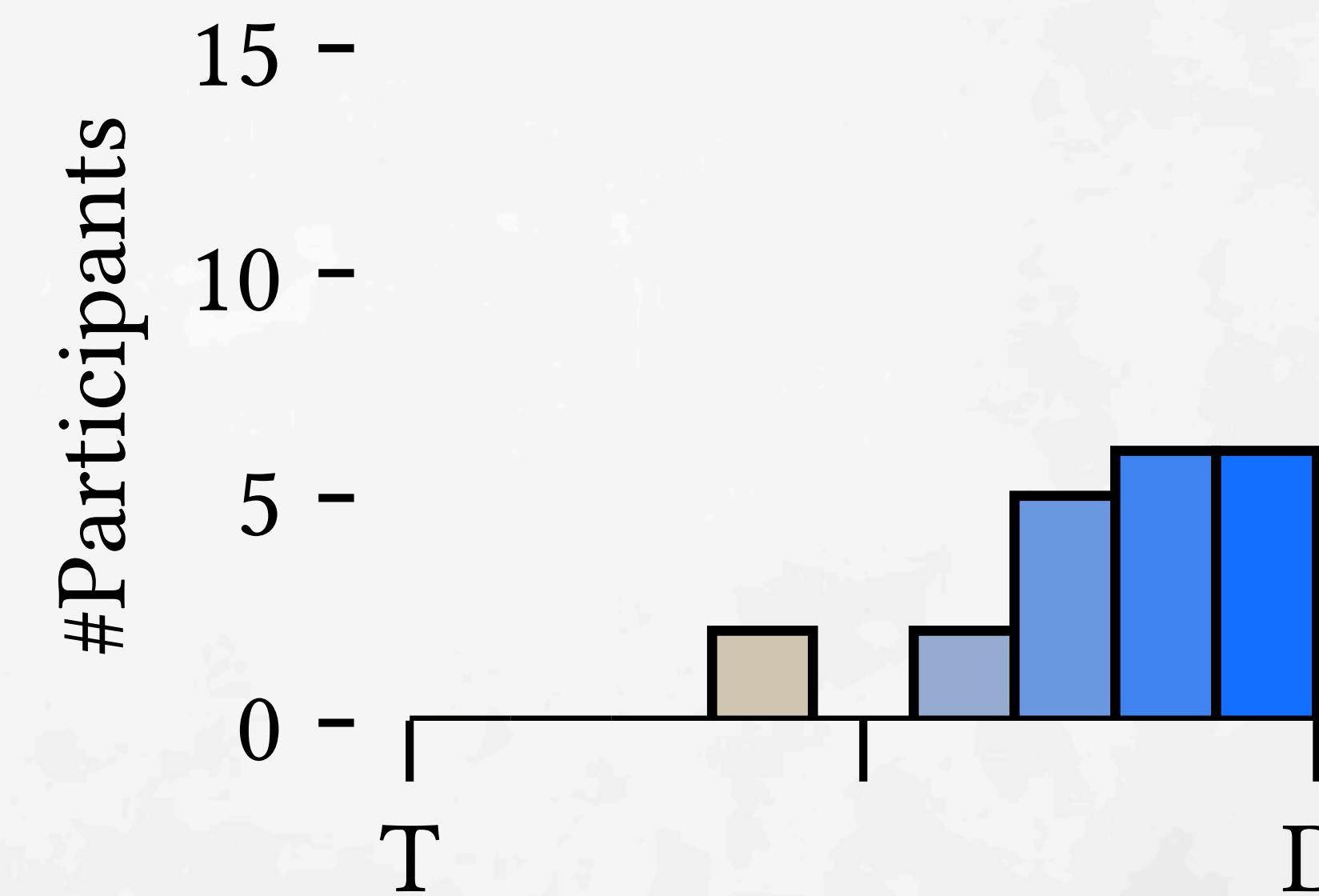
# Deuce preferred to Traditional?

Survey



Modest subjective  
preference for Deuce

Observed



Almost everyone  
used Deuce more

# Mix & Match Tool Usage

# Mix & Match Tool Usage

Rename

Make Equal with Single Variable

Introduce Variable(s)

Add Argument(s)

Create Function from Arguments

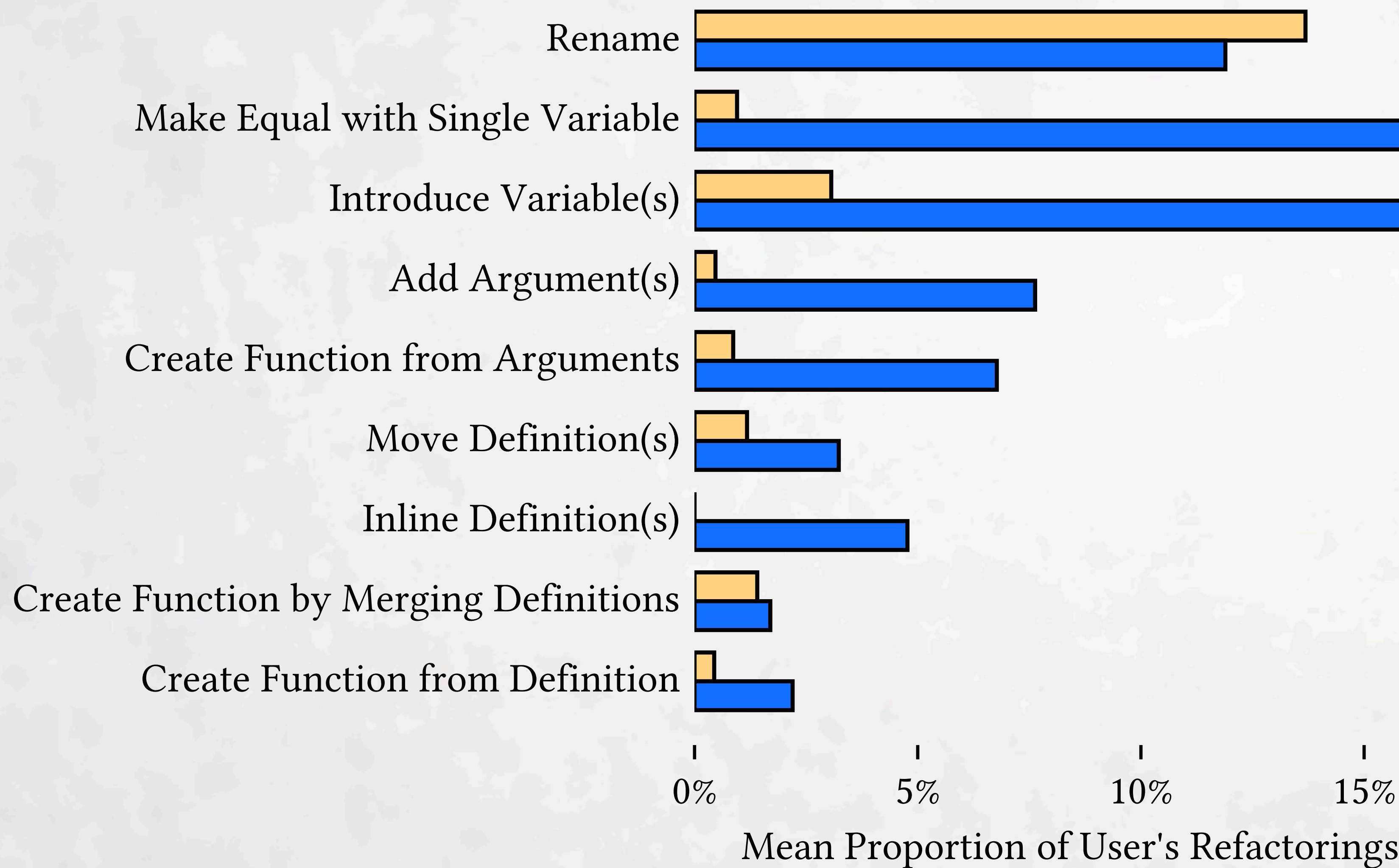
Move Definition(s)

Inline Definition(s)

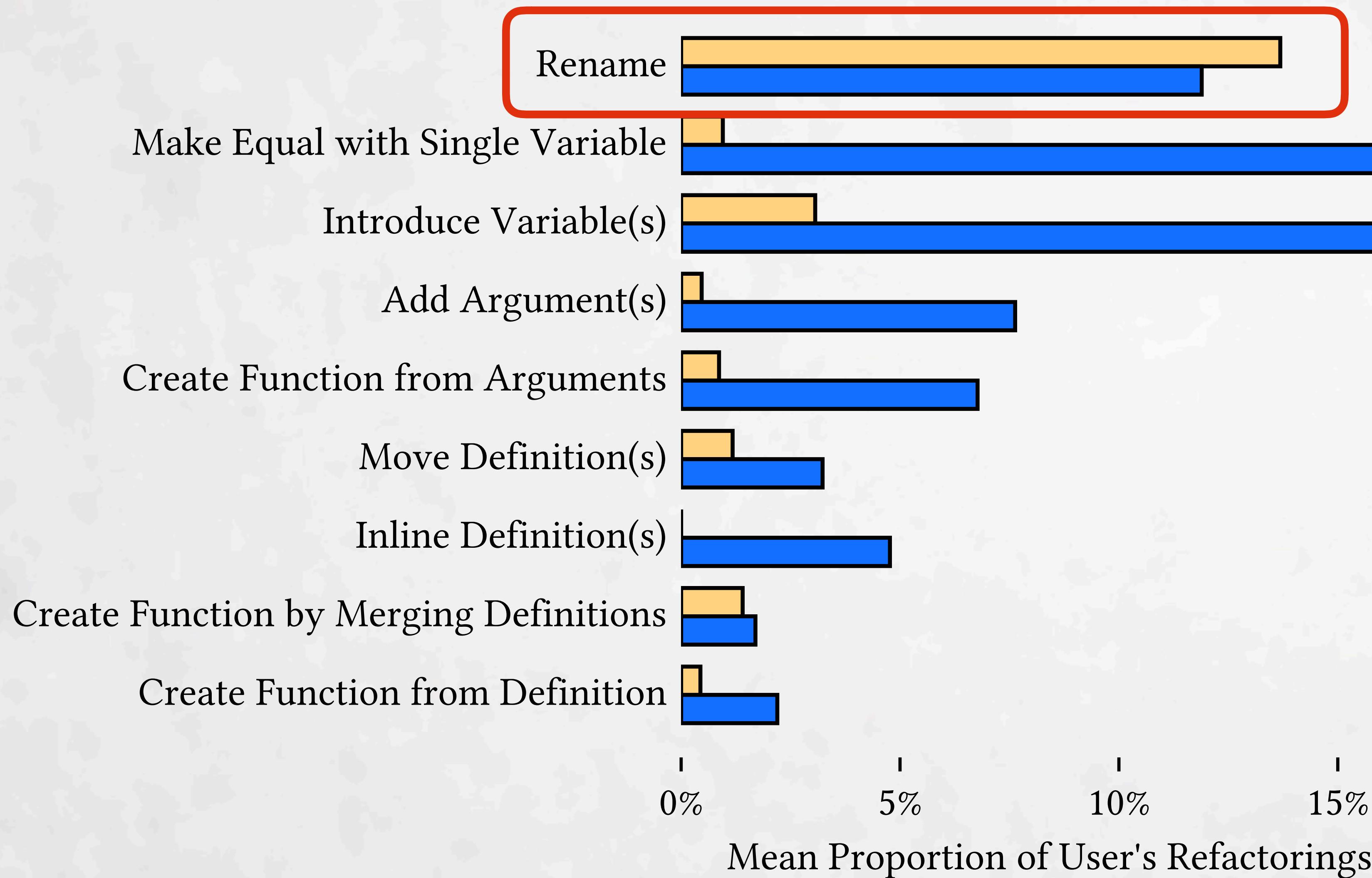
Create Function by Merging Definitions

Create Function from Definition

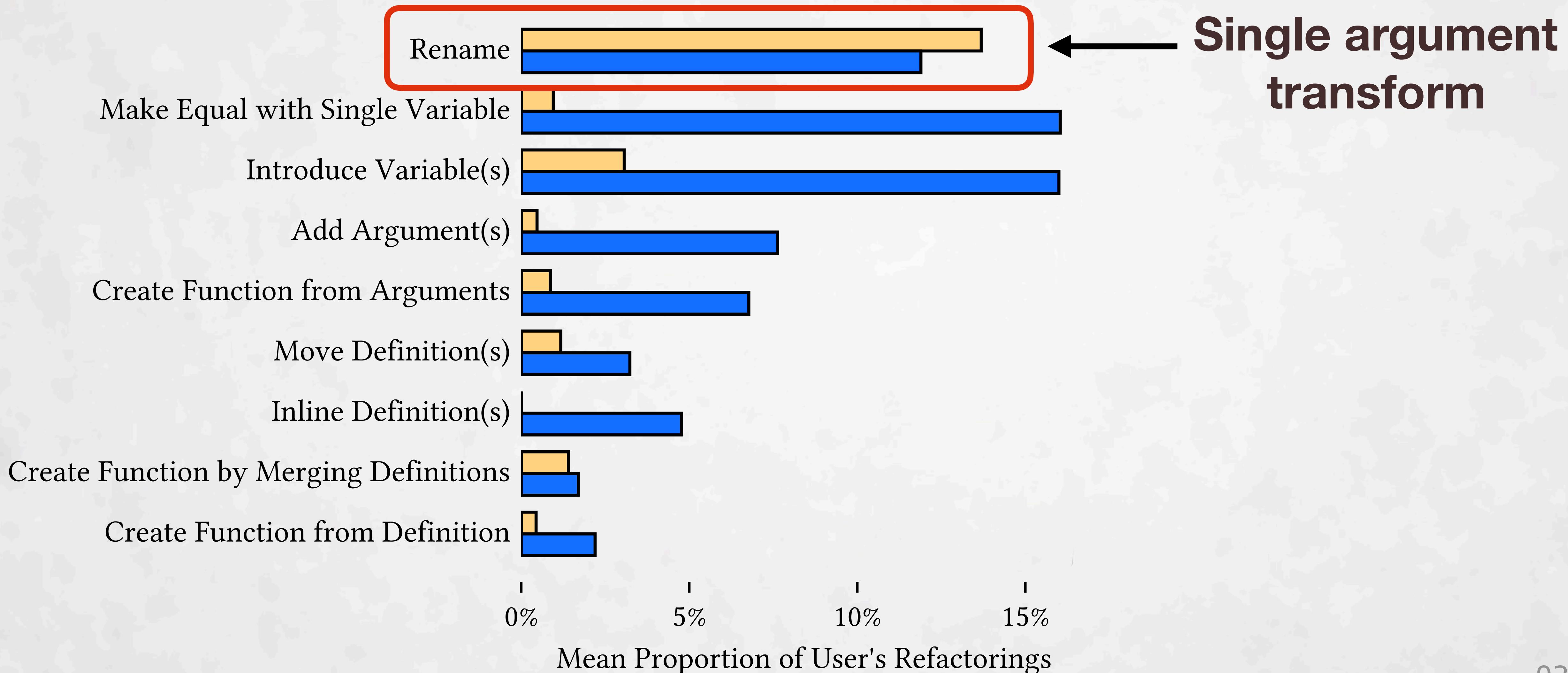
# Mix & Match Tool Usage



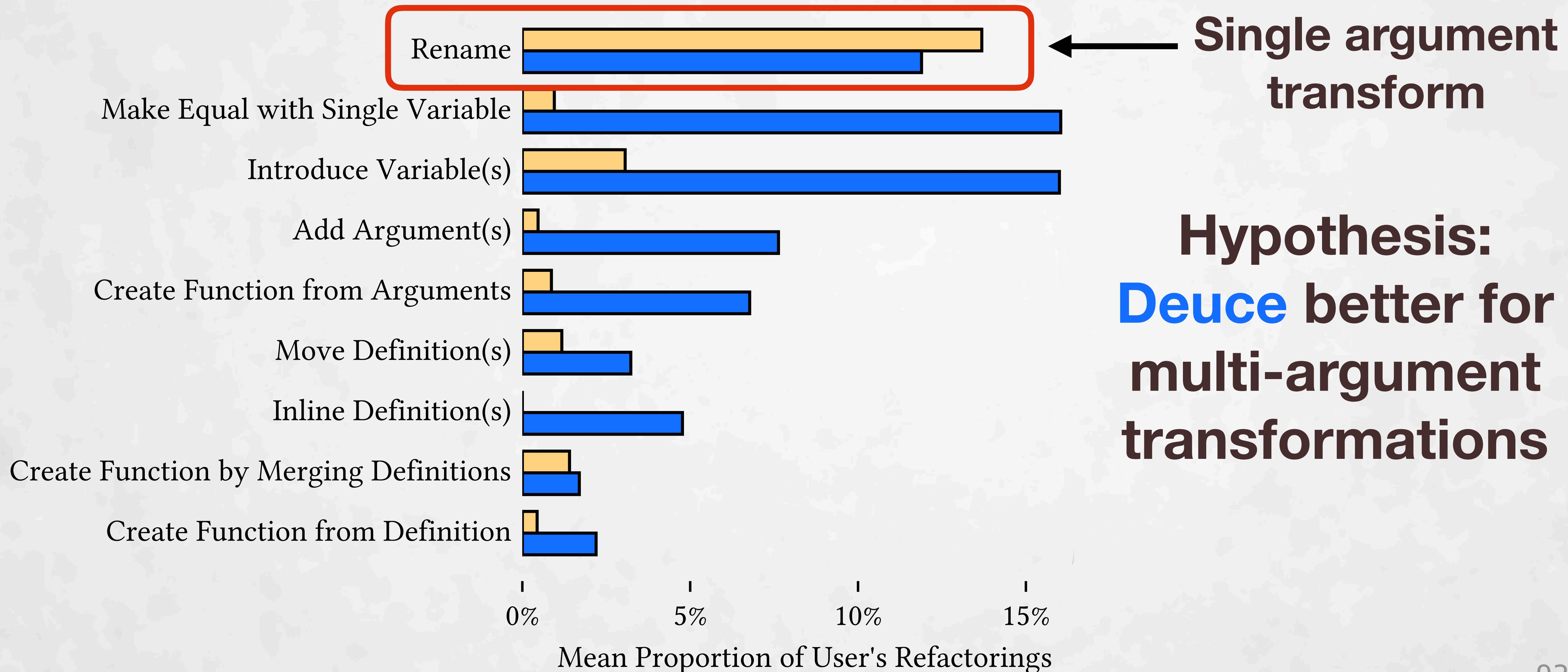
# Mix & Match Tool Usage



# Mix & Match Tool Usage



# Mix & Match Tool Usage



# Deuce vs Traditional

# Deuce vs Traditional

Traditional may be better for learning

# Deuce vs Traditional

Traditional may be better for learning

Deuce may be faster once learned

# Deuce vs Traditional

Traditional may be better for learning

Deuce may be faster once learned

Deuce strongly preferred

# Future Work

# Future Work

UI concerns for larger programs

# Future Work

UI concerns for larger programs

How to encourage refactoring?

# Future Work

UI concerns for larger programs

How to encourage refactoring?

DSL for defining new transformations

# Future Work

UI concerns for larger programs

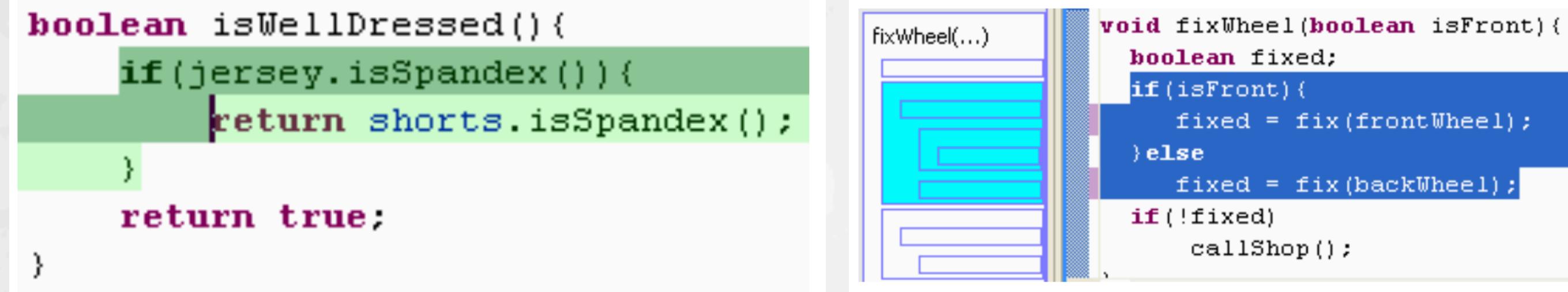
How to encourage refactoring?

DSL for defining new transformations

Real languages in existing editors

# Related Work

# Related Work



The image shows a Java code editor interface. On the left, a method named `isWellDressed()` is defined:

```
boolean isWellDressed() {
    if(jersey.isSpandex()) {
        return shorts.isSpandex();
    }
    return true;
}
```

A cursor is positioned at the end of the first line of the `if` block. A green rectangular box highlights the entire body of the `if` block, indicating a selected range. On the right, another method named `fixWheel(boolean isFront)` is shown with its implementation:

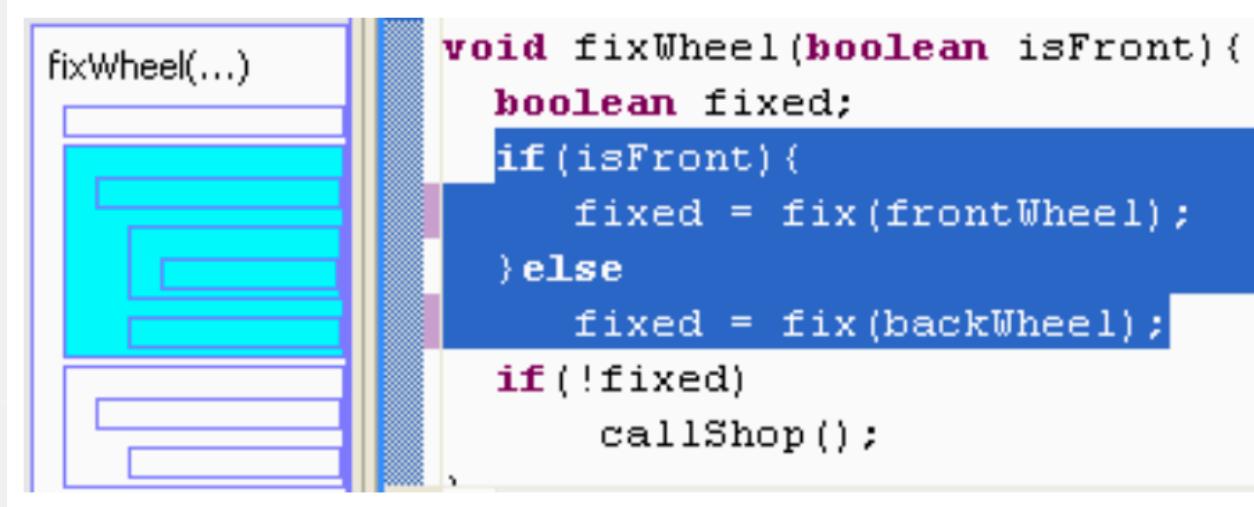
```
void fixWheel(boolean isFront) {
    boolean fixed;
    if(isFront) {
        fixed = fix(frontWheel);
    } else
        fixed = fix(backWheel);
    if(!fixed)
        callShop();
}
```

A blue rectangular box highlights the entire body of the `if` statement within the `fixWheel` method, indicating another selected range.

**Selection Assist + Box View**  
(Murphy-Hill and Black 2008)

# Related Work

```
boolean isWellDressed() {
    if(jersey.isSpandex()) {
        return shorts.isSpandex();
    }
    return true;
}
```



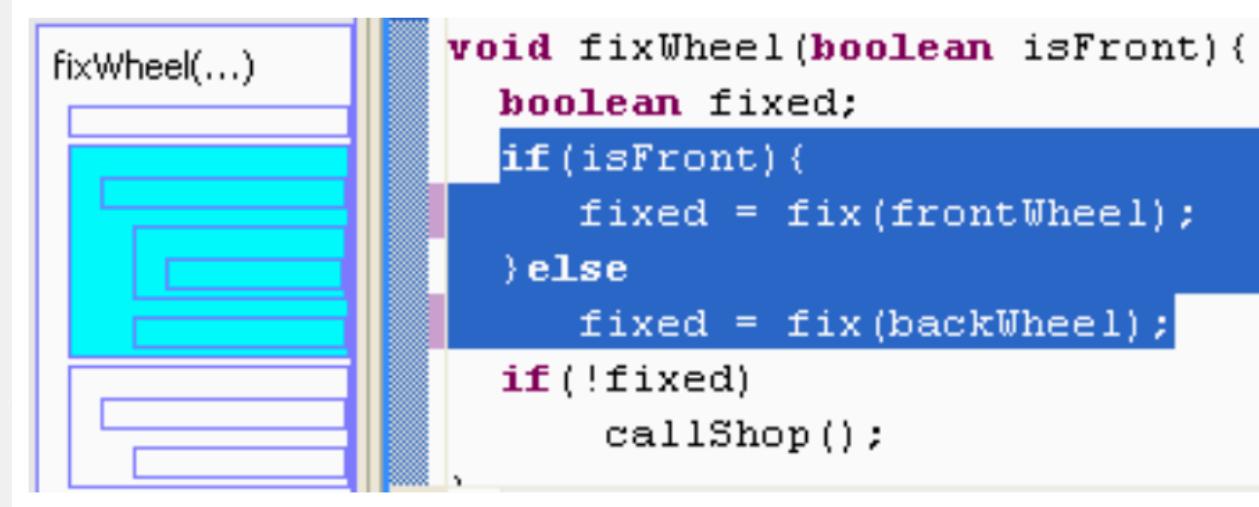
**Selection Assist + Box View**  
(Murphy-Hill and Black 2008)

```
public class Foo {
    public void bar() {
        ...
        System.out.println("hello world!");
        System.out.println("=====");
        ...
    }
}
```

**DNDRefactoring**  
(Lee et al. 2013)

# Related Work

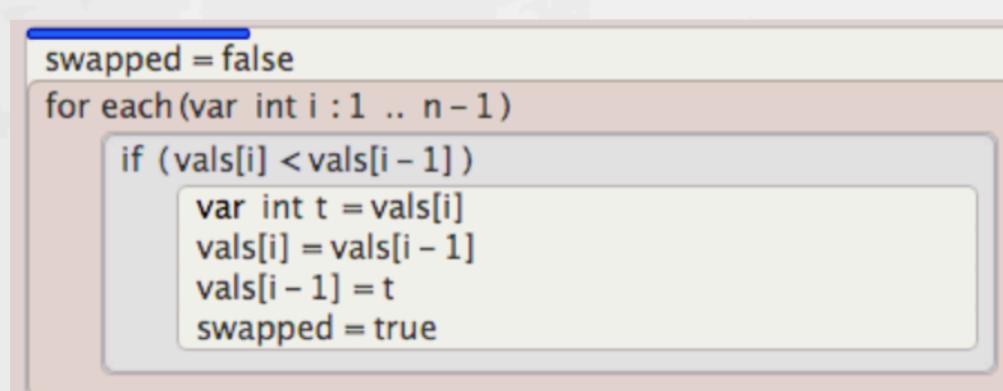
```
boolean isWellDressed() {
    if(jersey.isSpandex()) {
        return shorts.isSpandex();
    }
    return true;
}
```



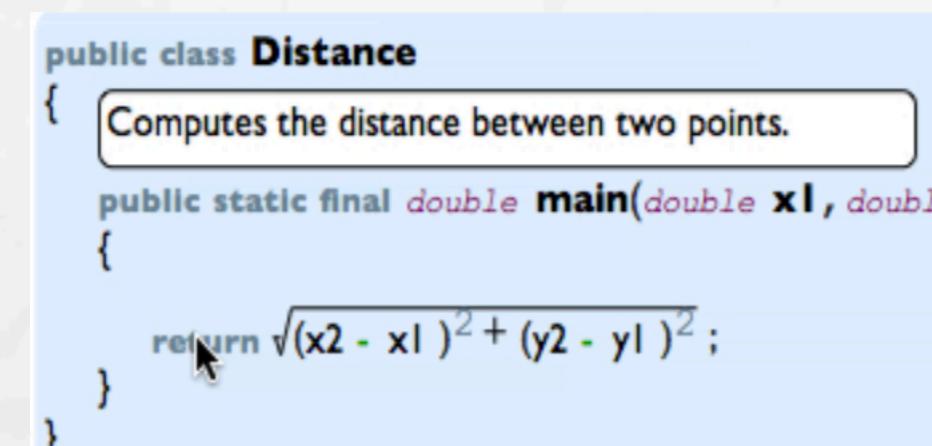
```
public class Foo {
    public void bar() {
        ...
        System.out.println("hello world!");
        System.out.println("=====");
        ...
    }
}
```

**Selection Assist + Box View**  
(Murphy-Hill and Black 2008)

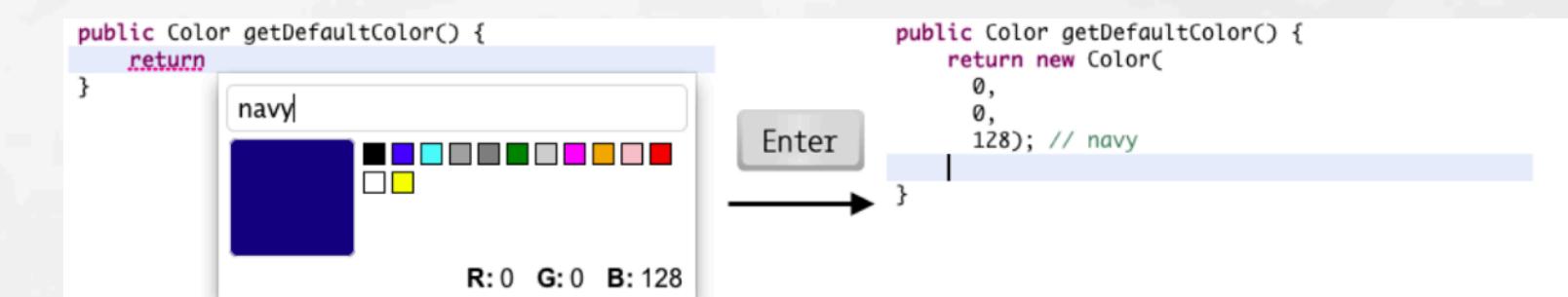
**DNDRefactoring**  
(Lee et al. 2013)



**Greenfoot**  
(Brown et al. 2016)



**Barista**  
(Ko and Myers 2006)



**Graphite**  
(Omar et al. 2012)



# Structure Select

```
(def image_url
  "img/icse-2018-large-icon-small.png")

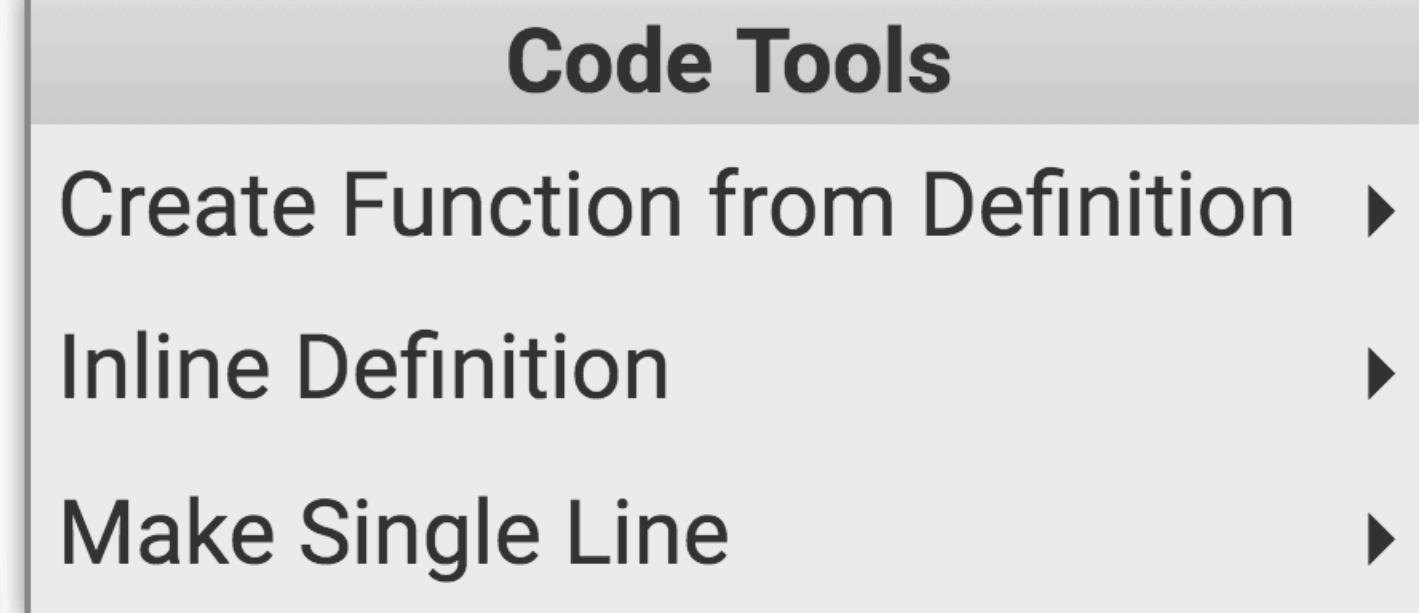
(def image1
  (let [width height] [324 200]
    (let [x y] [100 100]
      (image "lightgrey" x y width height 15 image_url)))))

(def main
  (draw (concat [ image1 ])))
```

# Structure Select

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200]  
    (let [x y] [100 100]  
      (image "lightgrey" x y width height 15 image_url))))  
  
(def main  
  (draw (concat [ image1 ])))|
```

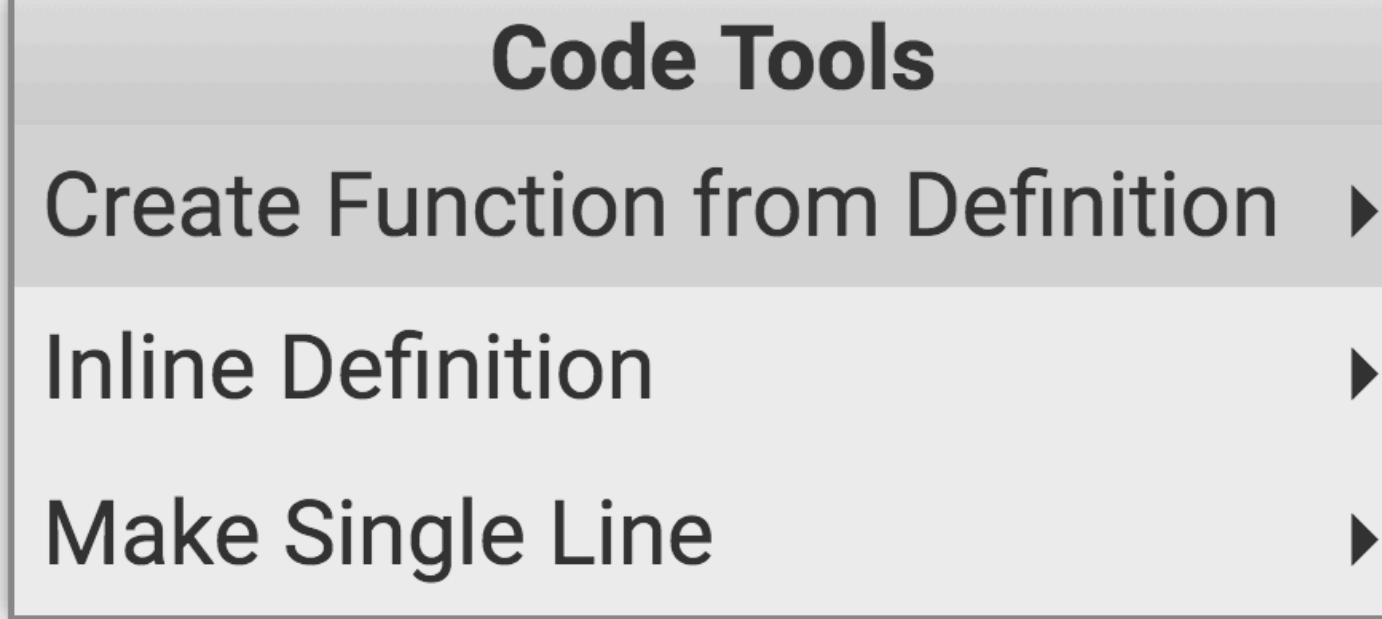
# Short Menu



# Structure Select

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200]  
    (let [x y] [100 100]  
      (image "lightgrey" x y width height 15 image_url))))  
  
(def main  
  (draw (concat [ image1 ])))
```

# Short Menu



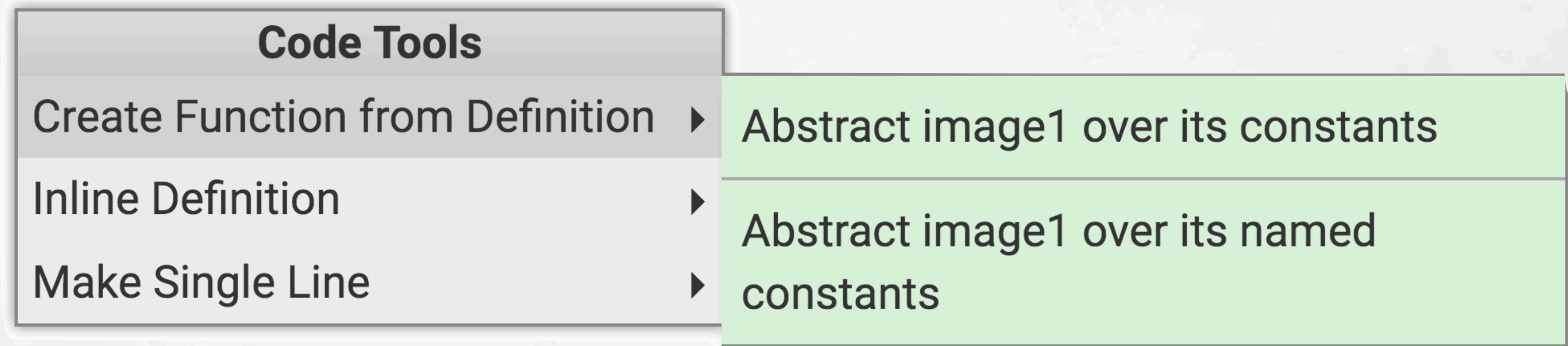
# Defaults

- Abstract image1 over its constants
- Abstract image1 over its named constants

# Structure Select

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200]  
    (let [x y] [100 100]  
      (image "lightgrey" x y width height 15 image_url))))  
  
(def main  
  (draw (concat [ image1 ])))
```

# Short Menu

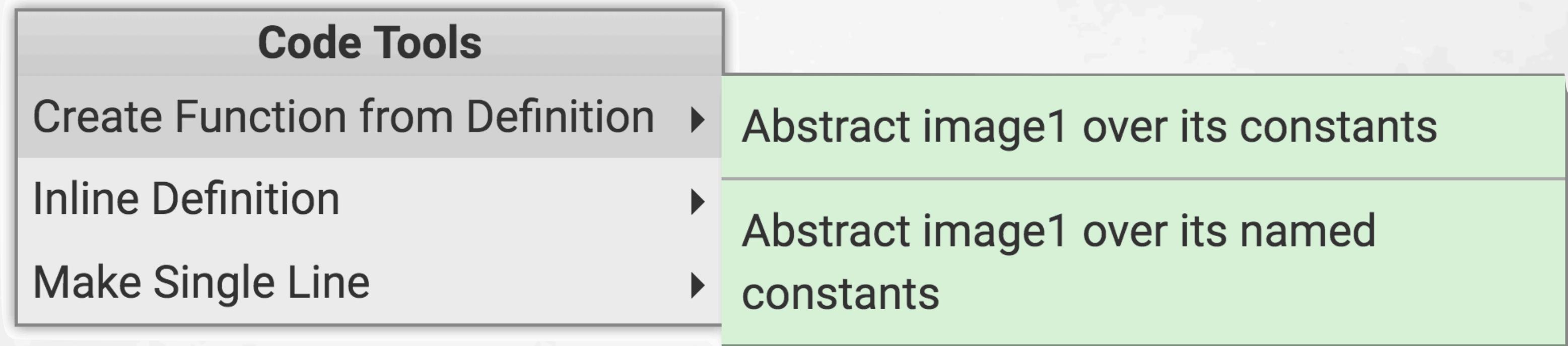


**Deuce** provides *streamlined structural editing* in a *familiar text-based environment*.

# Structure Select

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200]  
    (let [x y] [100 100]  
      (image "lightgrey" x y width height 15 image_url))))  
  
(def main  
  (draw (concat [ image1 ])))
```

# Short Menu



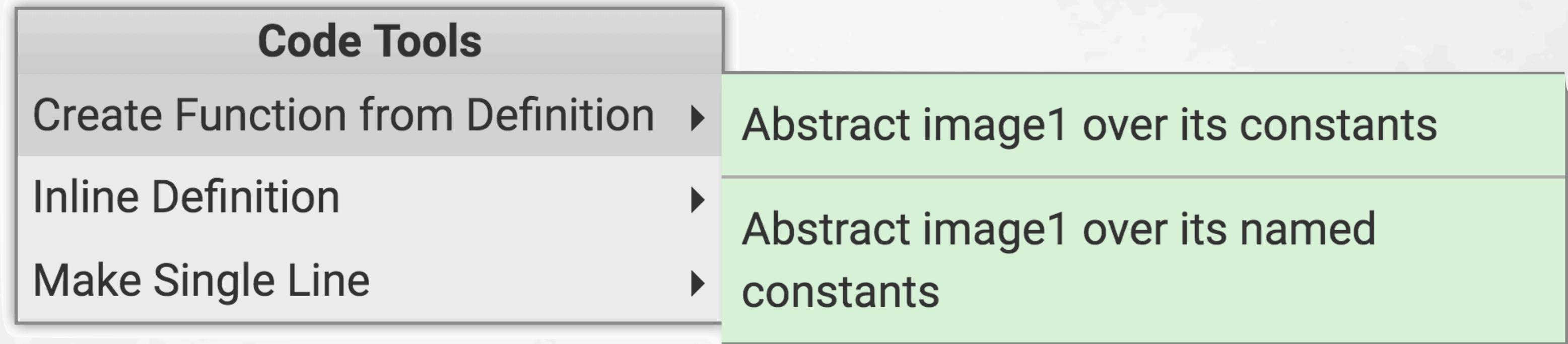
Search “sketch n sketch” to play with Deuce

# Thank you!

## Structure Select

```
(def image_url  
  "img/icse-2018-large-icon-small.png")  
  
(def image1  
  (let [width height] [324 200]  
    (let [x y] [100 100]  
      (image "lightgrey" x y width height 15 image_url))))  
  
(def main  
  (draw (concat [ image1 ])))
```

## Short Menu



## Search “sketch n sketch” to play with Deuce



# Extra Slides

<i>program</i>	$\bullet$ (def $x_0$ $e_0$ ) $\bullet \dots \bullet$ (def main $e$ )	
$e$	::=	$c$   $x$   $(\lambda p e)$   $(e_1 e_2)$   $[e_1   e_2]$   (let $p$ $e_1$ $e_2$ )   (case $e$ $(p_1 e_1) \dots$ )
$p$	::=	$c$   $x$   $[]$   $[p_1   p_2]$
Expressions $e$ ::=		$\bullet$ $e$ $\bullet$
Patterns $p$ ::=		$\bullet$ $p$ $\bullet$

Figure 1: Syntax of LITTLE. The orange boxes and blue dots identify features for structural selection.

```
EditorState = { code: Program, selections: Set Selection }
ActiveState = Active | NotYetActive | Inactive
Options     = NoOptions | StringOption String
Result       = { description: String, code: Program }
```

```
CodeTool =
{ name : String
, requirements : String
, active : EditorState -> ActiveState
, run : (EditorState, Options) -> List Result }
```

**Figure 2: Code tool interface.**

```
1
2 (def redSquare
3   (let [x y] [50 70]
4     (rect "salmon" x y 120 80)))
```

```
1
2 (def redSquare
3   (let [x y] [50 70]
4     (rect "salmon" x y 120 80)))
```

```
1
2 (def redSquare
3   (let [x y] [50 70]
4     (rect "salmon" x y 120 80)))
```

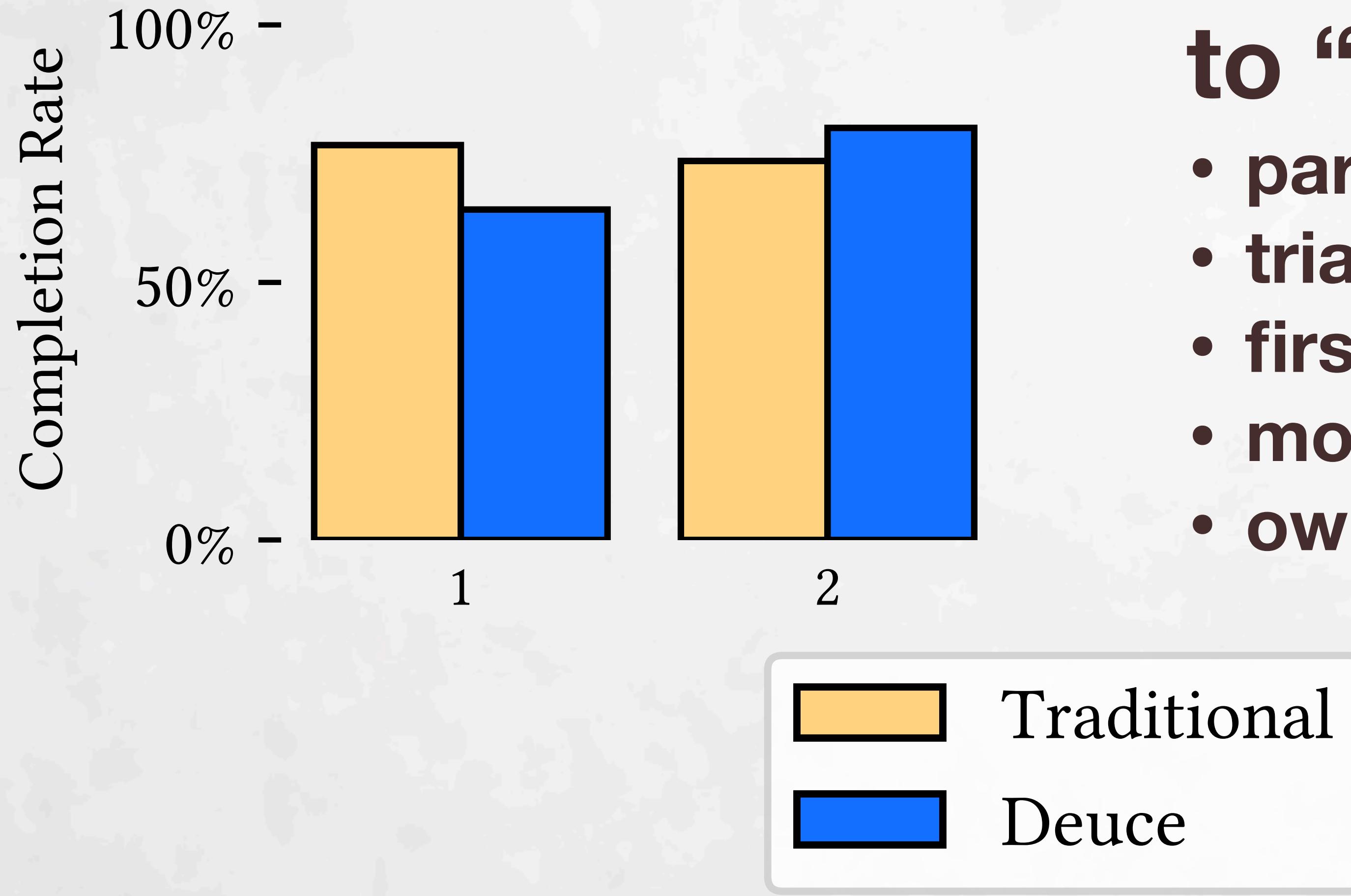
Figure 3: Example target positions.

## Code Tools

- Create Function from Definition...
- Create Function from Arguments...
- Create Function by Merging Definitions...
- Add Argument...
- Remove Argument...
- Reorder Arguments...
- Rename Variable...
- Introduce Local Variable...
- Swap Variable Names and Usages...
- Swap Variable Usages...

- Make Equal with Single Variable...
- Make Equal by Copying...
- Move Definition...
- Swap Definitions...
- Inline Definition...
- Duplicate Definition...
- Reorder Expressions...
- Swap Expressions...
- Make Single Line...
- Make Multi-line...
- Align Expressions...

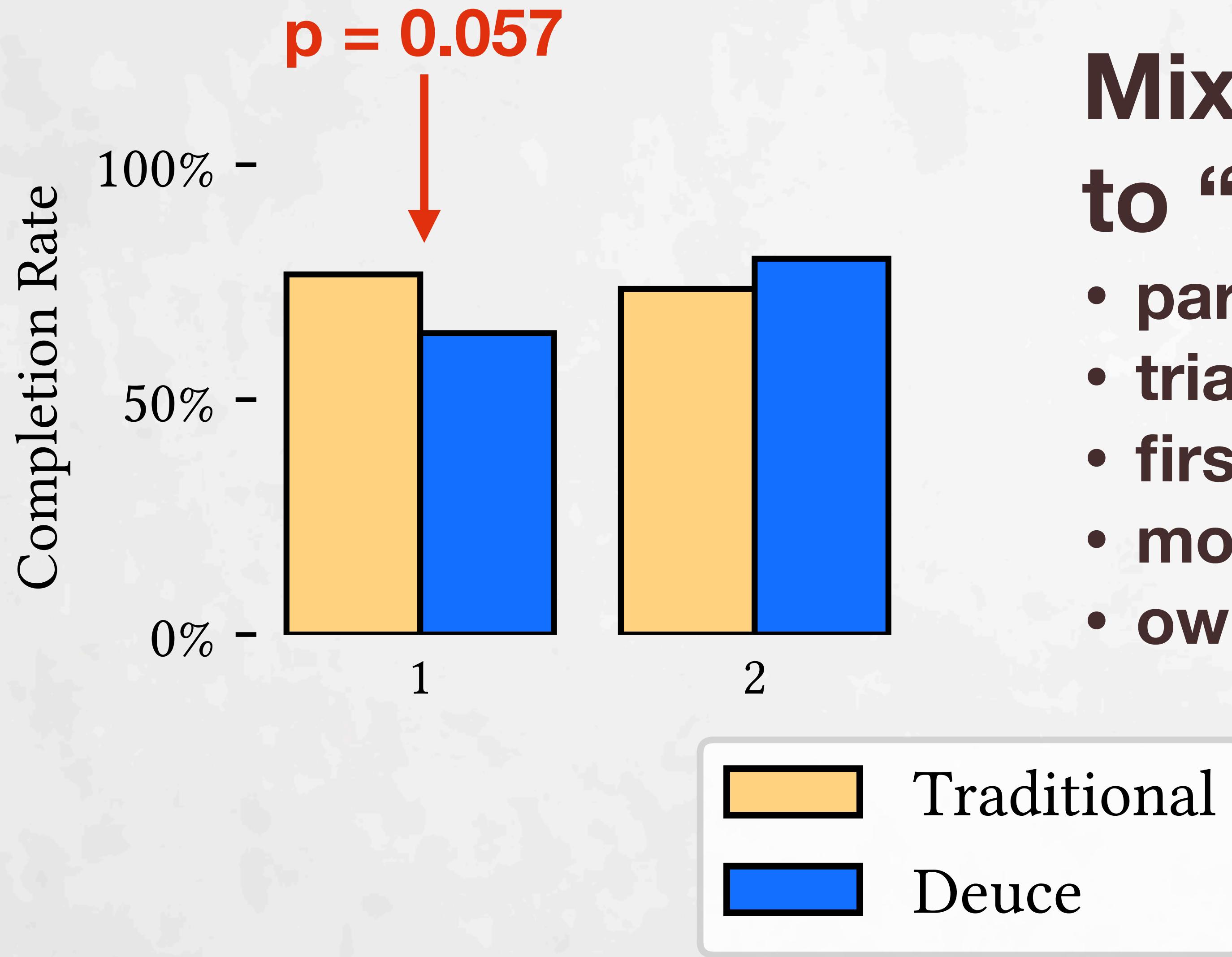
# Head-to-Head Tasks



**Mixed effects model  
to “control” for:**

- participant skill
- trial number
- first/second encounter
- mouse/trackpad
- own/our computer

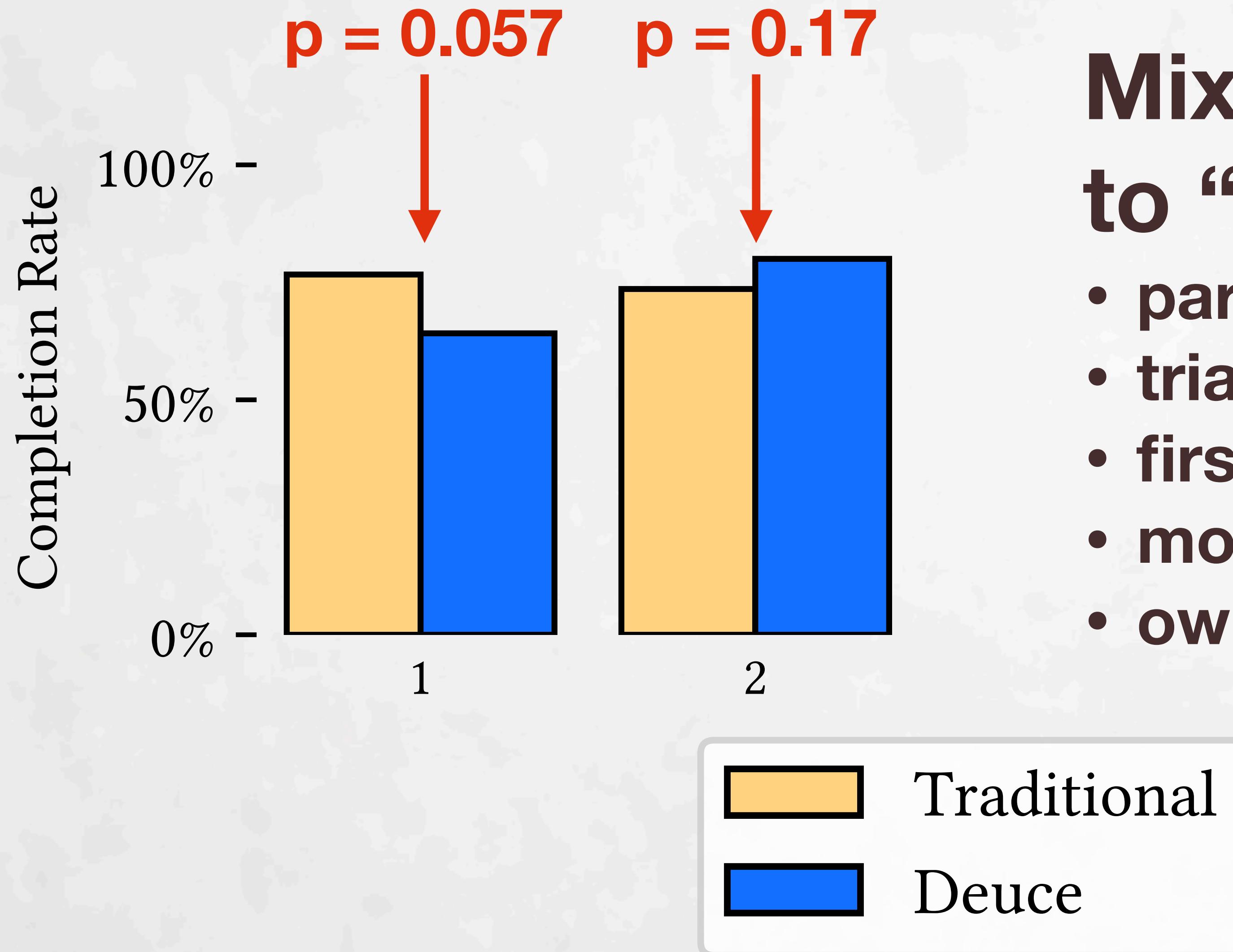
# Head-to-Head Tasks



Mixed effects model  
to “control” for:

- participant skill
- trial number
- first/second encounter
- mouse/trackpad
- own/our computer

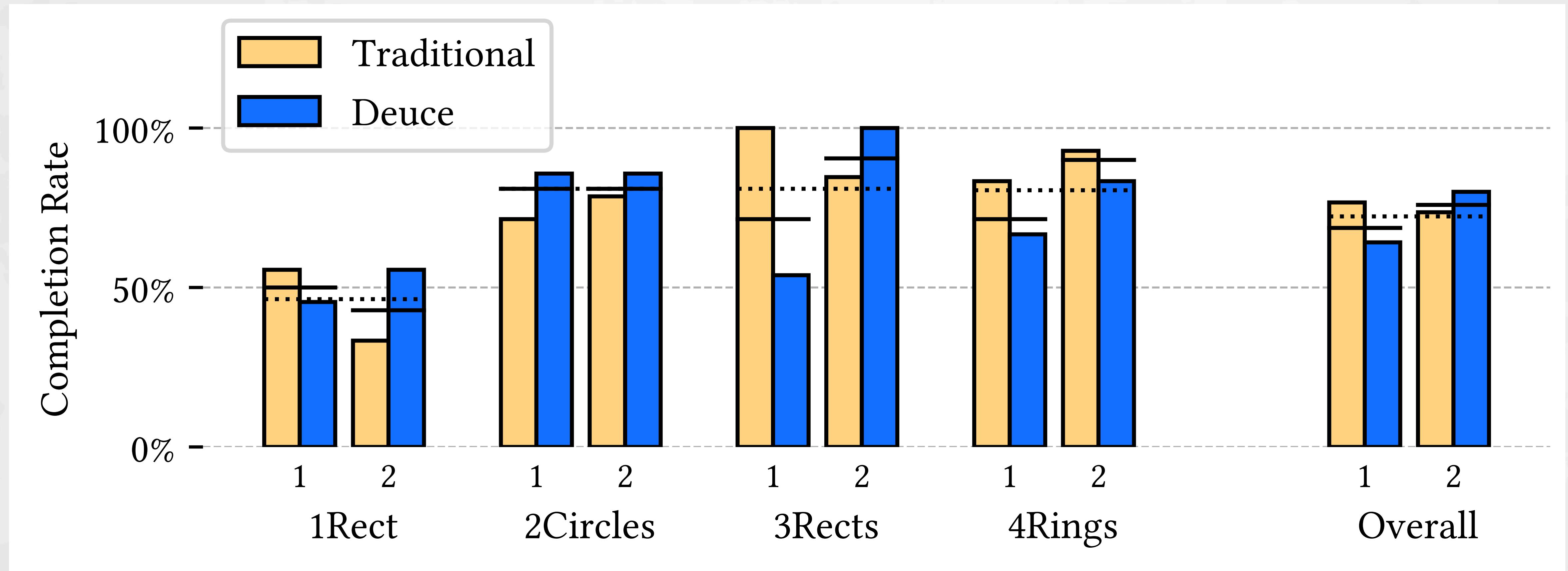
# Head-to-Head Tasks



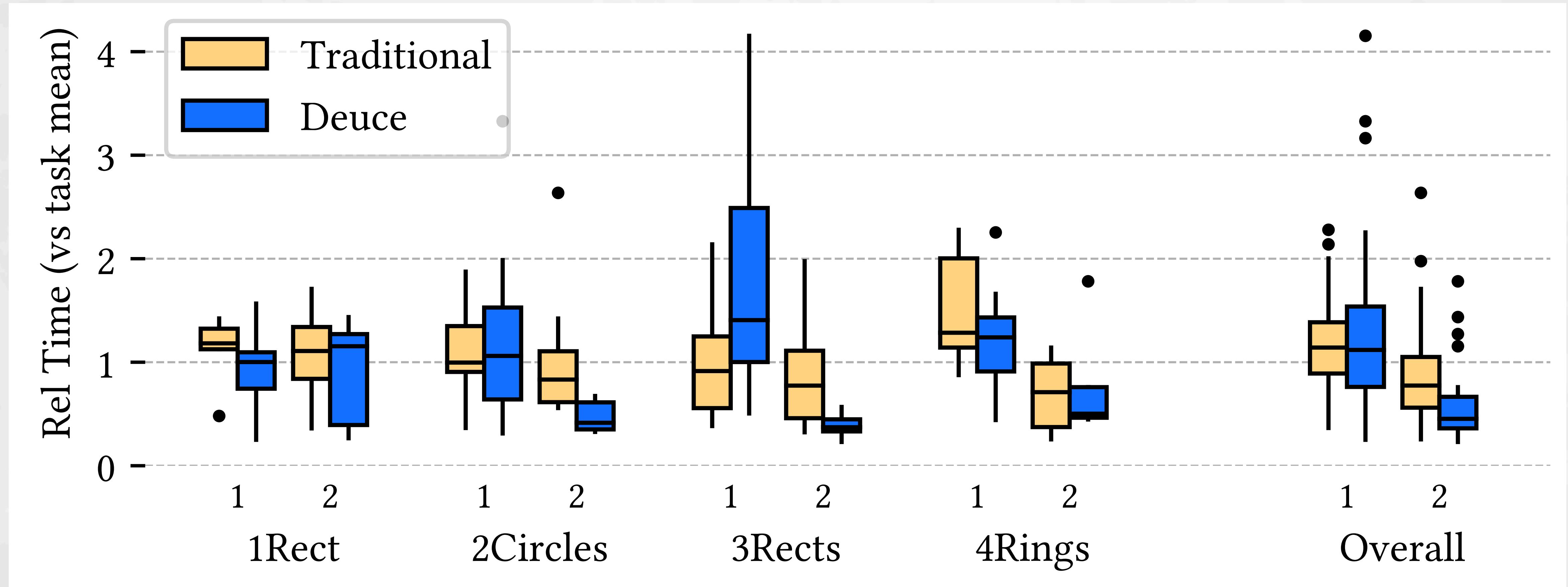
Mixed effects model  
to “control” for:

- participant skill
- trial number
- first/second encounter
- mouse/trackpad
- own/our computer

# Head-to-Head Tasks

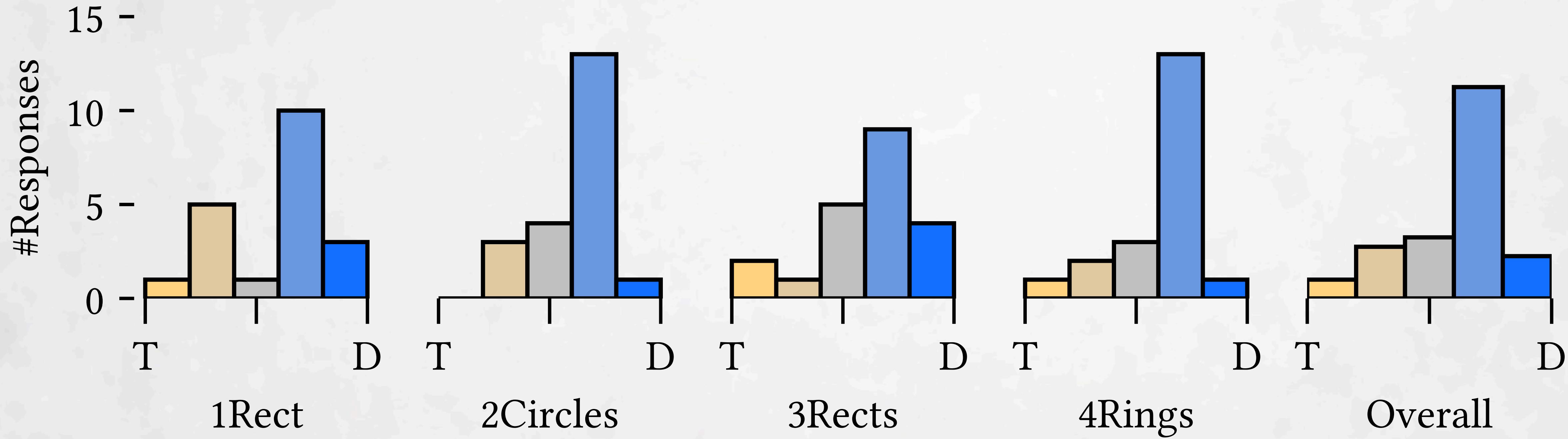


# Head-to-Head Tasks



# Head-to-Head Tasks

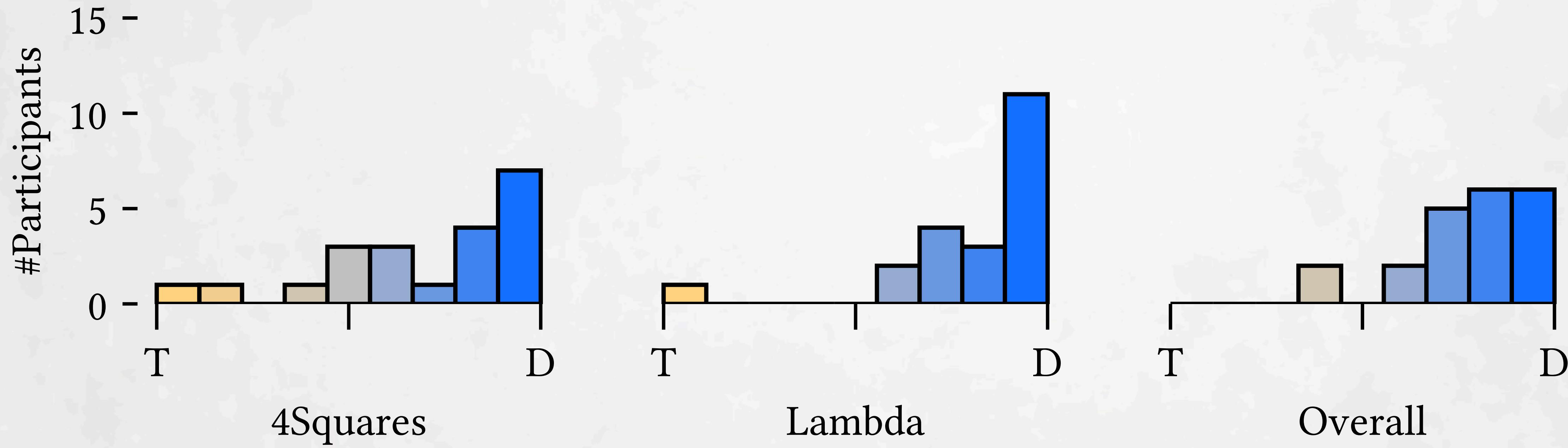
*“Which interaction worked better for the ... task?”*



Modest subjective preference for Deuce

# Mix & Match Tasks

What did participants actually use?



**Deuce preferred by almost all users.**