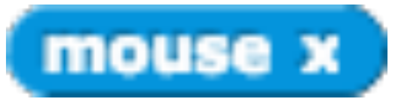
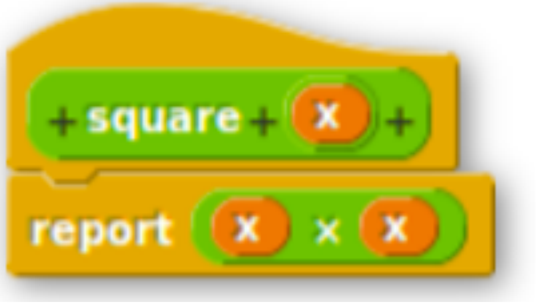
Snap – Abstraction & Testing

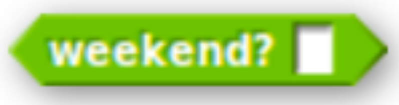
Reporters and Predicates (Recap)

- Perform a function and give back a value
- Predicates give back true/false value
- Custom blocks can report values



Reporter

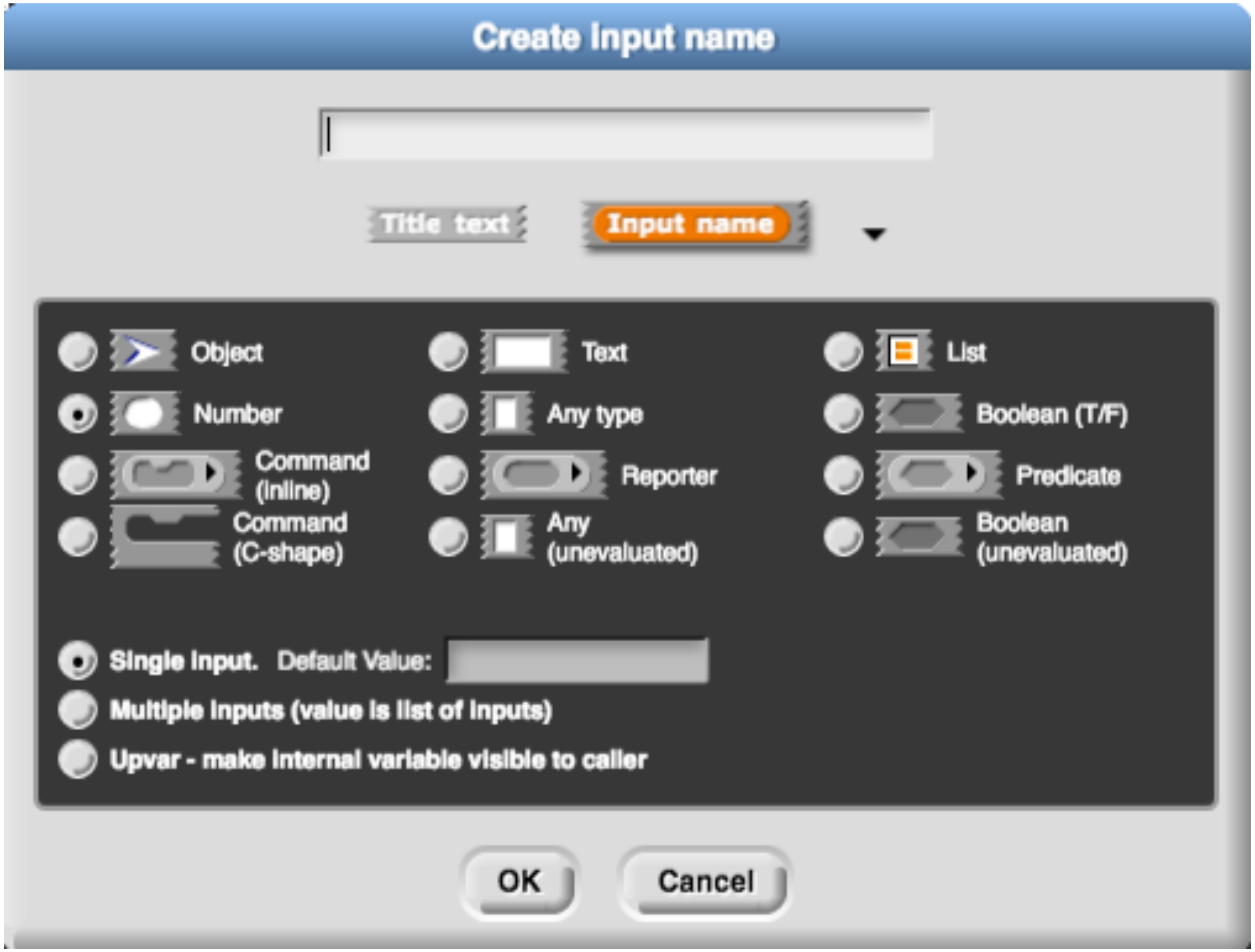




Predicate

Reporter – custom built

Blocks only accepting numbers



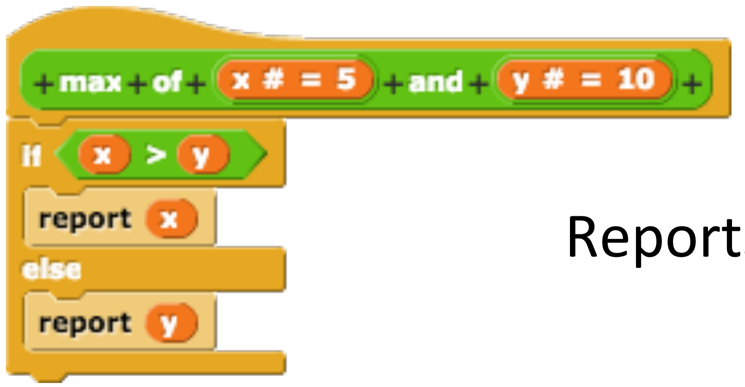
Create input name

Title text
Input name

( ) Object
( ) Text
( ) List
(•) Number
( ) Any type
( ) Boolean (T/F)
( ) Command (inline)
( ) Reporter
( ) Predicate
( ) Command (C-shape)
( ) Any (unevaluated)
( ) Boolean (unevaluated)

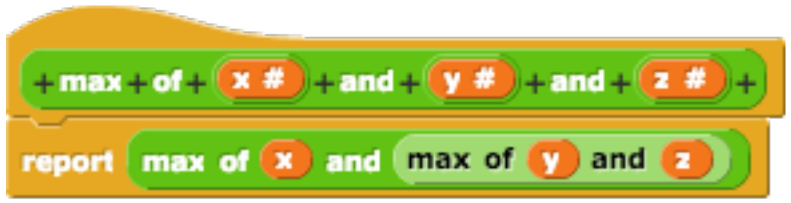
(•) Single input. Default Value:
( ) Multiple inputs (value is list of inputs)
( ) Upvar - make internal variable visible to caller

Blocks inside Blocks



```
+max+of+ (x # = 5) +and+ (y # = 10) +
if <(x) > (y)>
    report (x)
else
    report (y)
```

Reports the max of two numbers

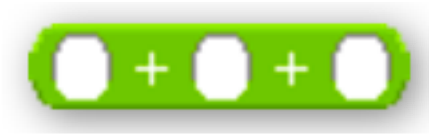


```
+max+of+ (x #) +and+ (y #) +and+ (z #) +
report (max of (x) and (max of (y) and (z)))
```

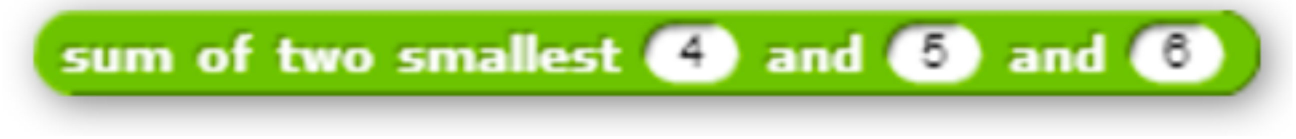
Reports the max of three numbers using the above max block

First 3 blocks

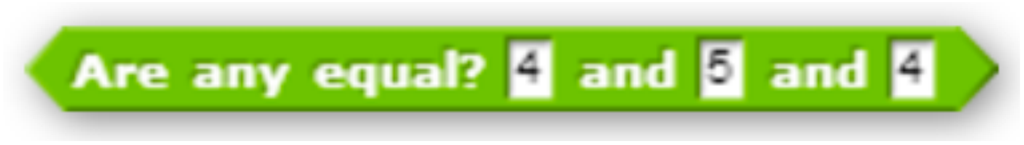
- A three-input addition operator that accepts only numbers.



- A reporter block named "sum of two smallest”



- A predicate block named "Are any equal?"

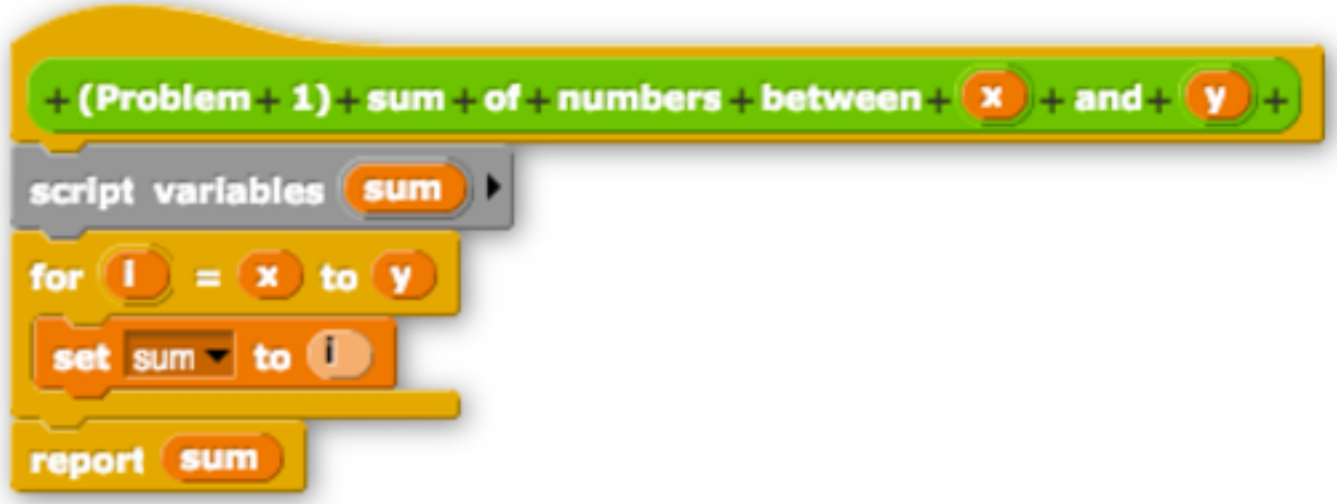


Debugging

- At some point in your programming career, you will discover errors
  - Syntax errors - code will never even run
    - something is written wrong
    - Not so common in Snap!
  - Runtime errors – code will run but crash
    - An impossible calculation is attempted
    - Ex: Division by zero; accessing a file that doesn't exist;
  - Logic errors – code runs without problem
    - You are getting incorrect results back
    - Usually a miscalculation

Debugging

- In the “Sum Things up” section, you will debug 3 blocks

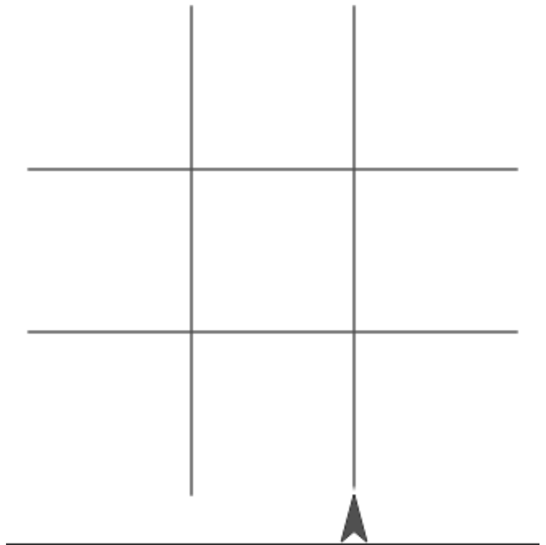


```
+ (Problem + 1) + sum + of + numbers + between + (x) + and + (y) +
script variables (sum)
for (i) = (x) to (y)
    set sum to (i)
report (sum)
```

…

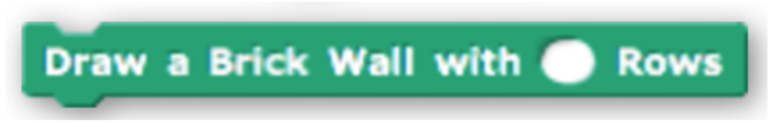
Tic Tac Toe

- Sometimes we need to break down complicated code into a more readable state
  - Use custom blocks
  - Use loops to repeat duplicate code
- You'll need to convert the provided code to produce a tic-tac-toe board



Abstraction

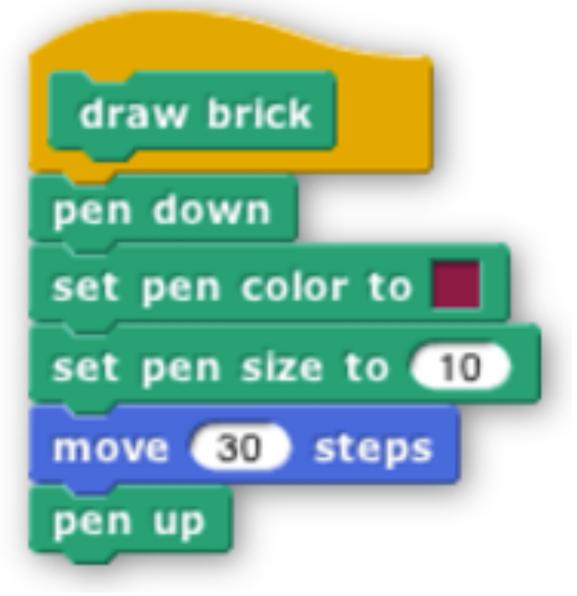
- Performing many subtasks that contribute to a greater task
- Example: Draw a brick wall with Z rows



A possible approach for brick wall

Create blocks that:

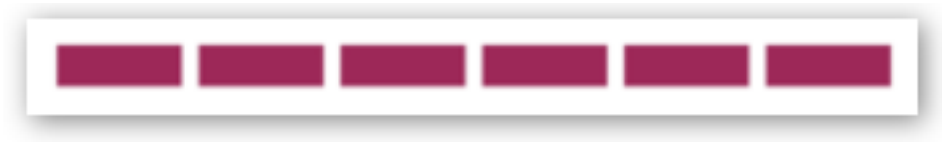
1. Draw one brick



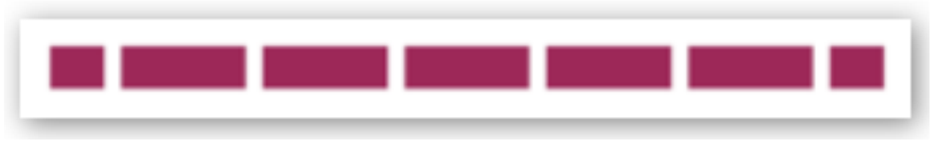
```
draw brick
pen down
set pen color to ■
set pen size to (10)
move (30) steps
pen up
```

A possible approach for brick wall

2. Draw one row `rowA`

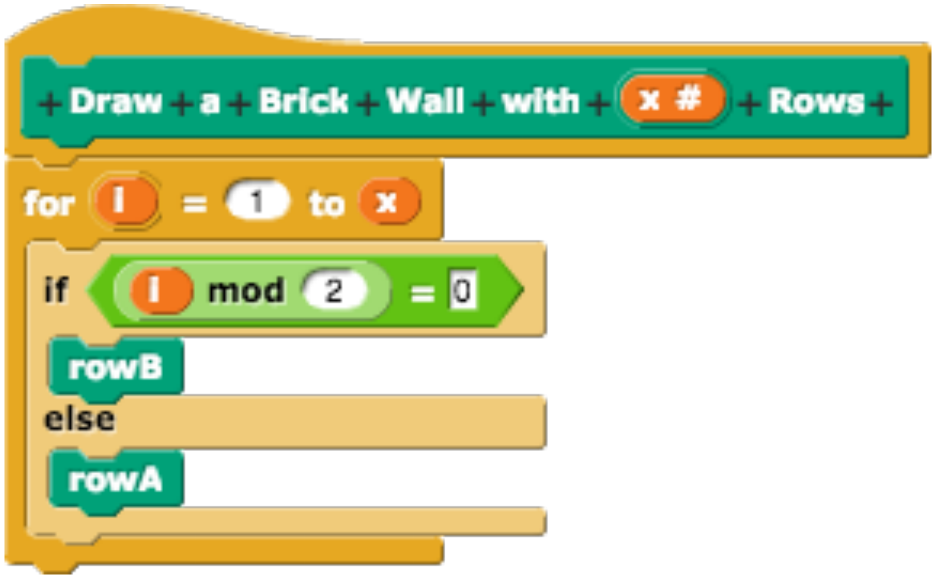


3. Draw another row (offset the first and last bricks) `rowB`



A possible approach for brick wall

4. Draw the entire wall, alternating rows



```
+ Draw + a + Brick + Wall + with + (x #) + Rows +
for (i) = (1) to (x)
    if < ((i) mod (2)) = [0] >
        rowB
    else
        rowA
```