Slides for Pre-reading

CPSC 100

Computational Thinking

Intro to Programming

Instructor: Firas Moosvi
Department of Computer Science
University of British Columbia

Agenda

- Learning Goals
- Course Admin
- Intro to Programming [Continued]
  - Boolean Functions/Expressions
  - Repeat Blocks → For Loops

Course Admin

Content for Test X

- In general, if a test starts on the Saturday of Week 5, then test content will specifically include stuff from Friday (Wk 4), Monday (Wk 5) and Wednesday (Wk 5)

- Because in computational thinking, everything builds on itself, knowledge from previous topics is fair game!

Learning Goals

After this week's lecture, you should be able to:

- Define **boolean expressions** and their role in Snap! Programming
- **Apply** boolean operations to control flow in programming
- Describe the concept of **loops** and **iteration** in programming
  - Differentiate between **finite loops** (repeat N times) and **condition-based loops** (repeat until)
- Use logical reasoning to **determine the output of given code**
  - Apply CT to **trace and evaluate** code snippets

Q: What is the value of gift after the block is run, assume user input is 100?

iClicker

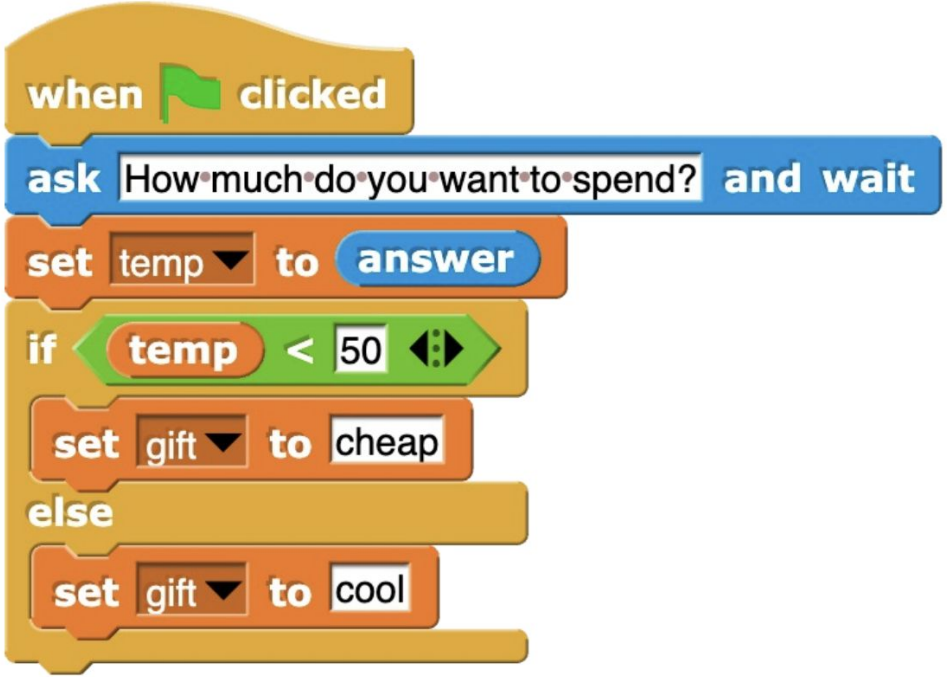
A. 50

B. 100

C. 150

D. cheap

E. cool



Q: What is the value of gift after the block is run, assume user input is 50?

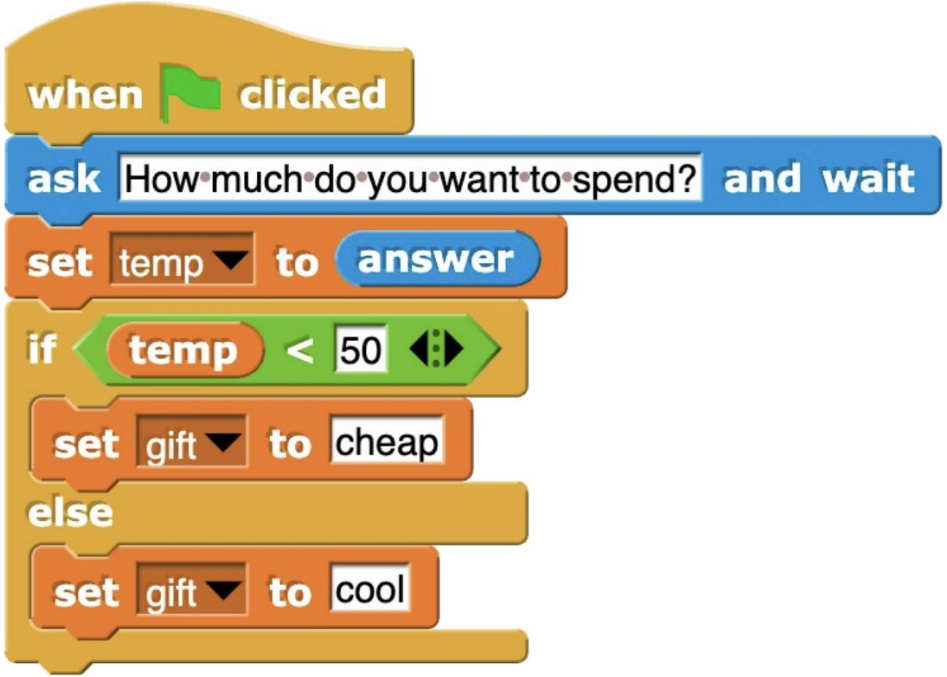
A. 50

B. 100

C. 150

D. cheap

E. cool



Programming Basics

Recall

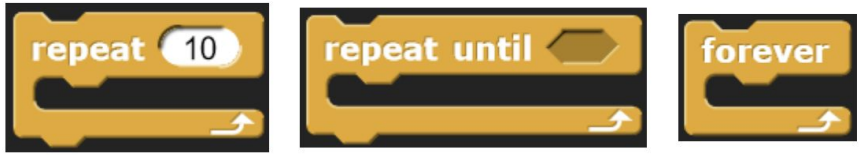
Programming Basics

0. Variable
    a. A named storage location for data
1. Sequencing
    a. In order (top-down)
2. Selection
    a. Conditional Structures
3. Iteration
    a. Loop Structures

Recall

Components of an Algorithm

0. Variable
    a. A named storage location for data
1. **Sequencing**
    a. In order (top-down)
2. **Selection**
    a. Conditional Structures
3. **Iteration**
    a. Loop Structures



Boolean (or Logical) Function

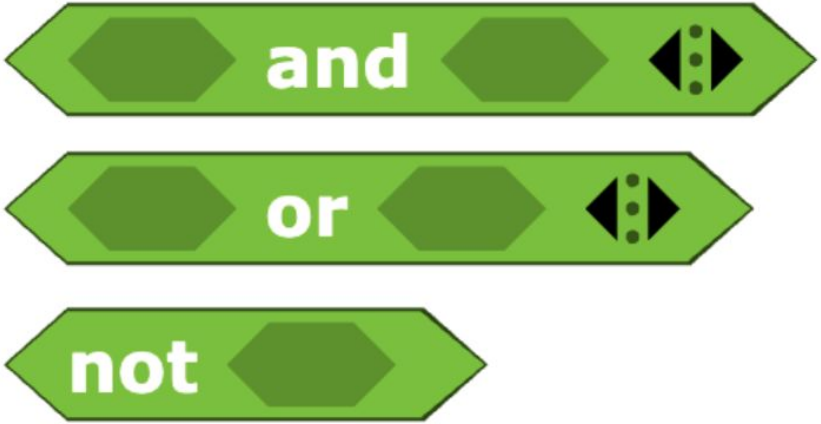
Boolean (or Logical) Function

At the very lowest level, computer circuitry is made of wires, and each wire either has power or it does not: meaning it's either **on or off**.

The only operations that can be performed at the lowest level are those that operate on **single-bit values**

0 or 1 on or off

Boolean (or Logical) Function



Notice that both the blocks themselves and the input slots in the blocks are hexagonal.

Boolean functions take Boolean values (True or False) as **inputs** and report a new Boolean value as **output**.

Boolean → Data Representation

OR Blocks | **Evaluates to**

False

True

True

True

<u>OR</u> evaluates to **true**, as long as **<u>one</u>** <u>operand evaluates to **true**</u>

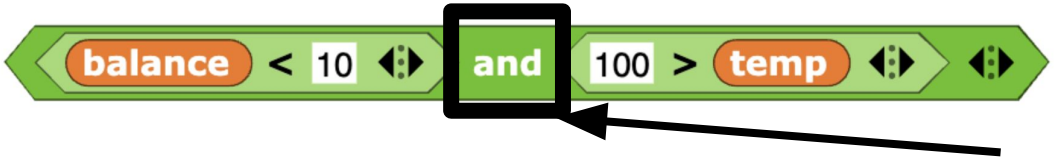
Boolean → Data Representation

OR Blocks	Evaluates to	AND Blocks	Evaluates to
✗ or ✗	False	✗ and ✗	False
✗ or ✓	True	✗ and ✓	False
✓ or ✗	True	✓ and ✗	False
✓ or ✓	True	✓ and ✓	True

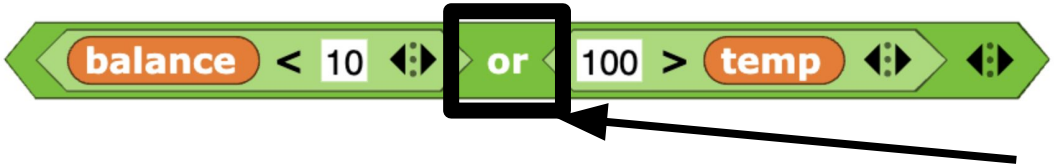
OR evaluates to **true**, as long as **one** operand evaluates to **true**
AND evaluates to **true**, only if **all** operands are **true**

Boolean Function Examples

balance and temp are variables



if balance is less than 10 **AND**
if temp is greater than 100

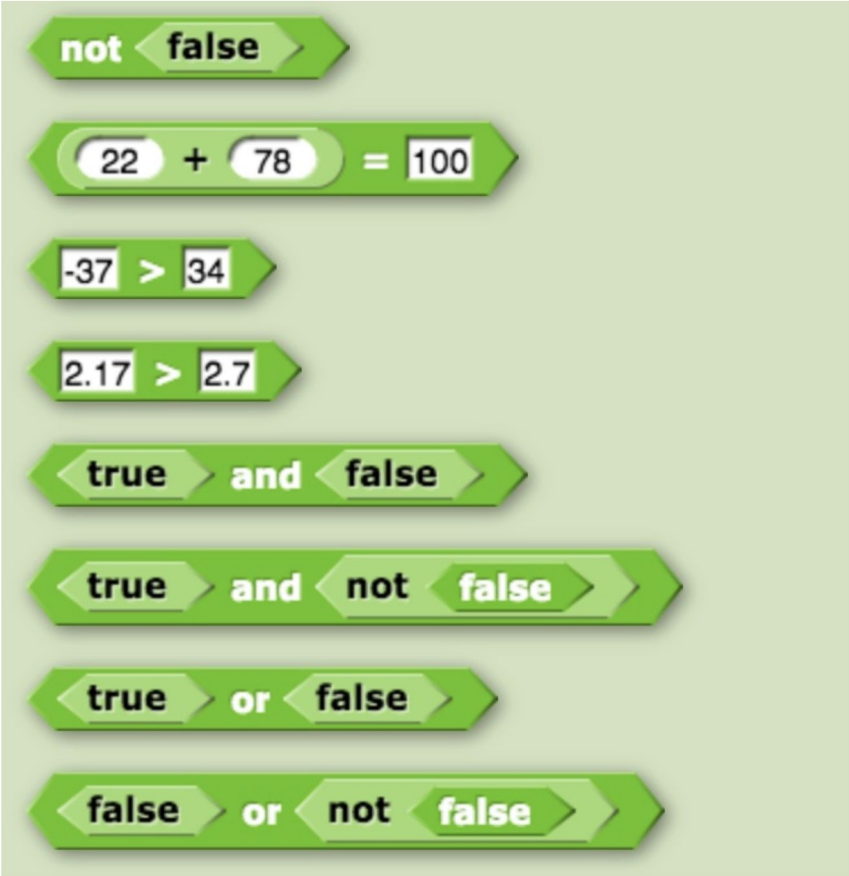


if balance is less than 10 **OR**
if temp is greater than 100

if balance is **NOT** less than 10

Class Activity

What does each block evaluate to?



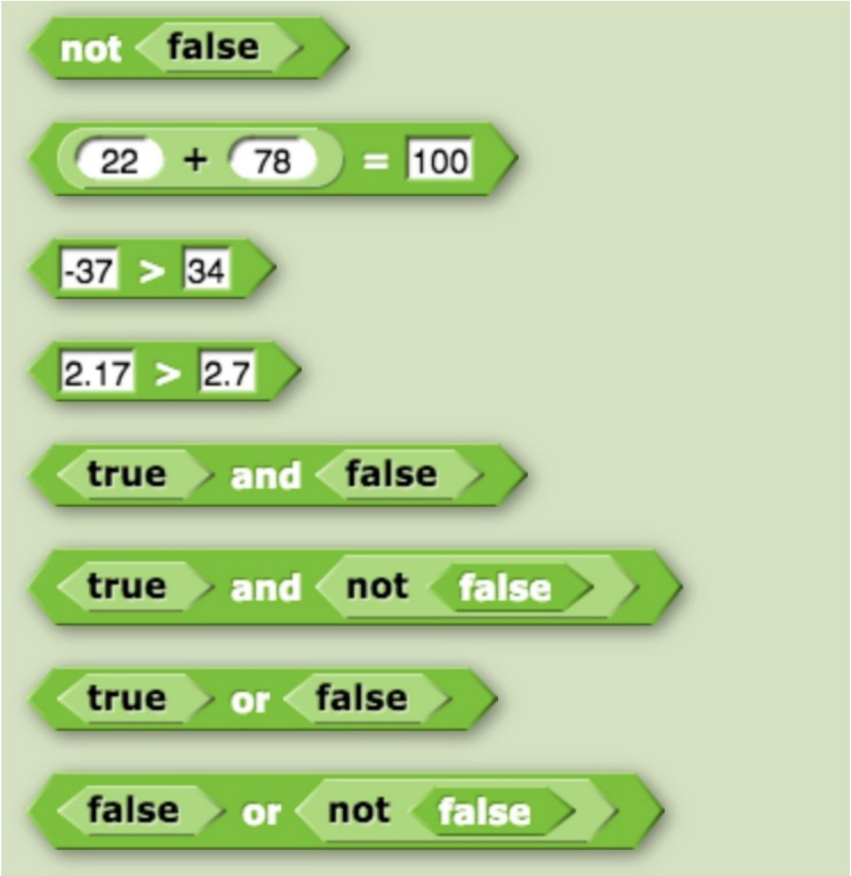
True?

OR

False?

What does each block evaluate to?

True.



True?

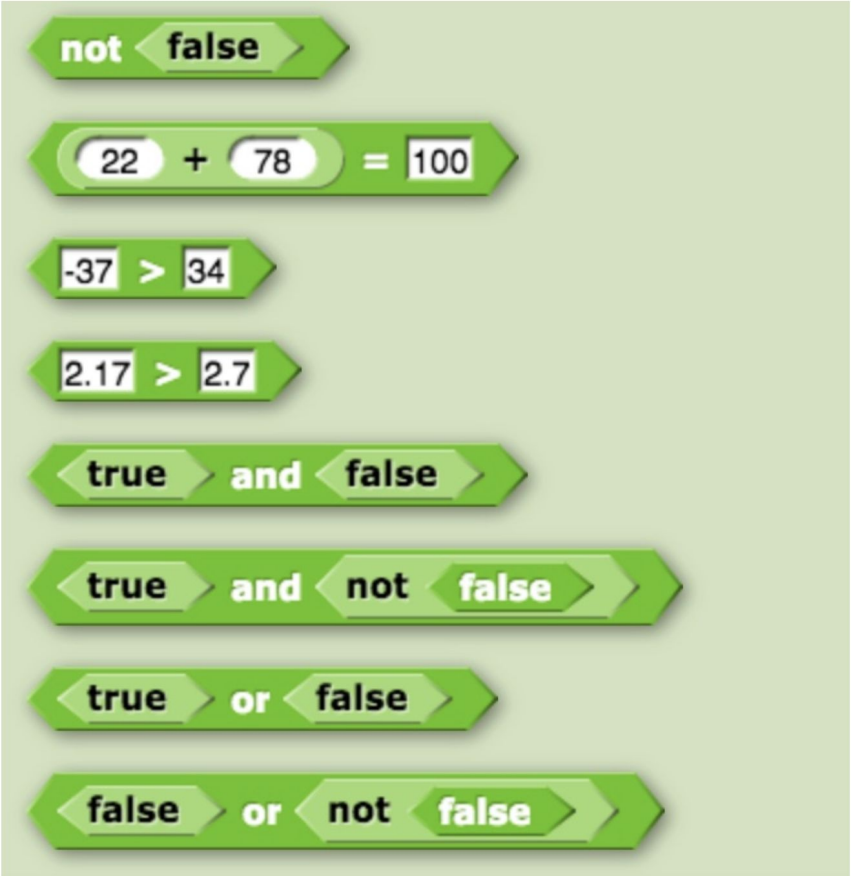
OR

False?

What does each block evaluate to?

True.

True.



True?

OR

False?

What does each block evaluate to?

True.	not false
True.	22 + 78 = 100
False.	-37 > 34
	2.17 > 2.7
	true and false
	true and not false
	true or false
	false or not false

True?

OR

False?

What does each block evaluate to?

True.	not false
True.	22 + 78 = 100
False.	-37 > 34
False.	2.17 > 2.7
	true and false
	true and not false
	true or false
	false or not false

True?

OR

False?

What does each block evaluate to?

True.	not false
True.	22 + 78 = 100
False.	-37 > 34
False.	2.17 > 2.7
False.	true and false
	true and not false
	true or false
	false or not false

True?

OR

False?

What does each block evaluate to?

True.	not false
True.	22 + 78 = 100
False.	-37 > 34
False.	2.17 > 2.7
False.	true and false
True.	true and not false
	true or false
	false or not false

True?

OR

False?

What does each block evaluate to?

True.	not false
True.	22 + 78 = 100
False.	-37 > 34
False.	2.17 > 2.7
False.	true and false
True.	true and not false
True.	true or false
	false or not false

True?

OR

False?

What does each block evaluate to?

True.	not false
True.	22 + 78 = 100
False.	-37 > 34
False.	2.17 > 2.7
False.	true and false
True.	true and not false
True.	true or false
True.	false or not false

True?

OR

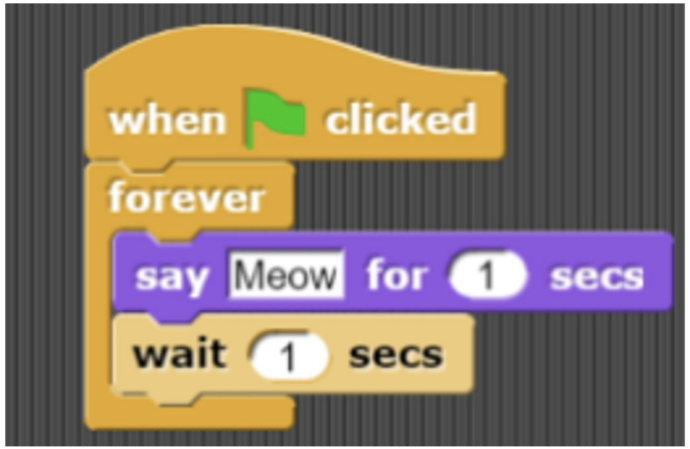
False?

Recall

Iteration

What if you want to do a task over and over again?

A **loop** allows you to do the same task over & over again, sometimes with a **stopping** condition, sometimes **forever!**



Repeat Blocks

Repeat some code a finite number of time

Repeat UNTIL a particular condition has been met.

If the condition is never met, then, it goes on *forever.*

Q: What is the value in total and num when this code block is run?

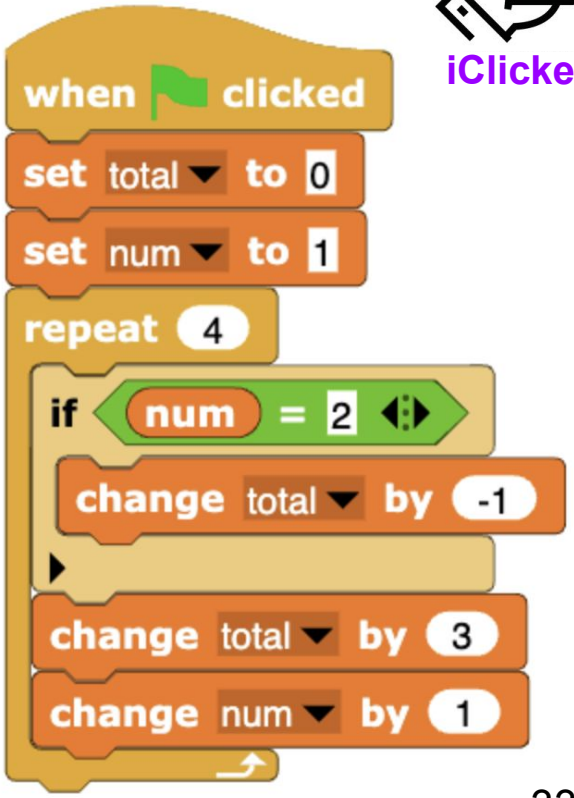
A. total = 11; num = 5

B. total = 12; num = 5

C. total = 9; num = 4

D. total = 10; num = 5

E. total = 11; num = 6



Q: What is the value in total and num when this code block is run?

iClicker

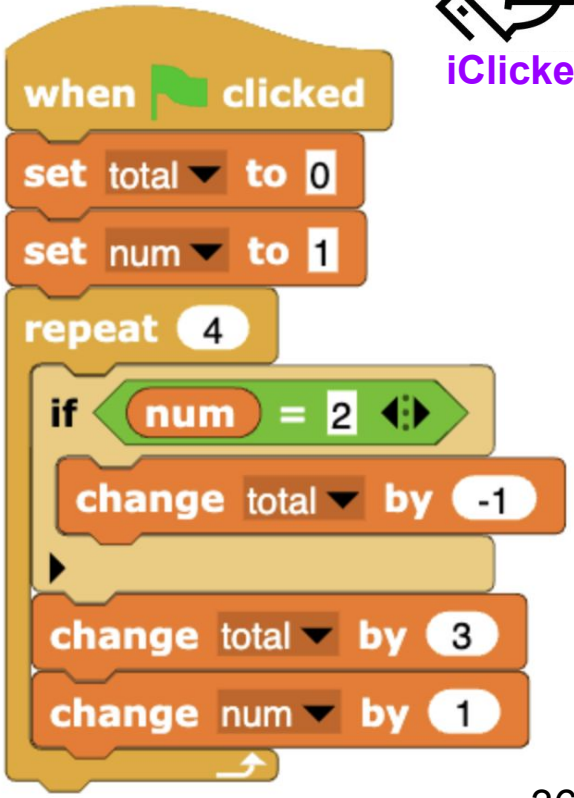
Iteration	num (Before)	Condition Check (num = 2?)	Change to total (-1 if true)	total +3	total (After)	num +1	num (After)
1st	1	No (skip -1)	0	+3	3		
2nd							
3rd							
4th							

```
when [flag] clicked
set total to 0
set num to 1
repeat 4
    if <num = 2>
        change total by -1
    change total by 3
    change num by 1
```

Q: What is the value in total and num when this code block is run?

iClicker

Iteration	num (Before)	Condition Check (num = 2?)	Change to total (-1 if true)	total +3	total (After)	num +1	num (After)
1st	1	No (skip -1)	0	+3	3	+1	2
2nd							
3rd							
4th							



Q: What is the value in total and num when this code block is run?

iClicker

Iteration	num (Before)	Condition Check (num = 2?)	Change to total (-1 if true)	total +3	total (After)	num +1	num (After)
1st	1	No (skip -1)	0	+3	3	+1	2
2nd	2	Yes (-1 applied)	3 - 1 = 2	+3	5	+1	3
3rd	3	No (skip -1)	5	+3	8	+1	4
4th	4	No (skip -1)	8	+3	11	+1	5

```
when [flag] clicked
set total to 0
set num to 1
repeat 4
    if <num = 2>
        change total by -1
    change total by 3
    change num by 1
```

Q: What is the value in i when the code is run, assuming user input = 3?

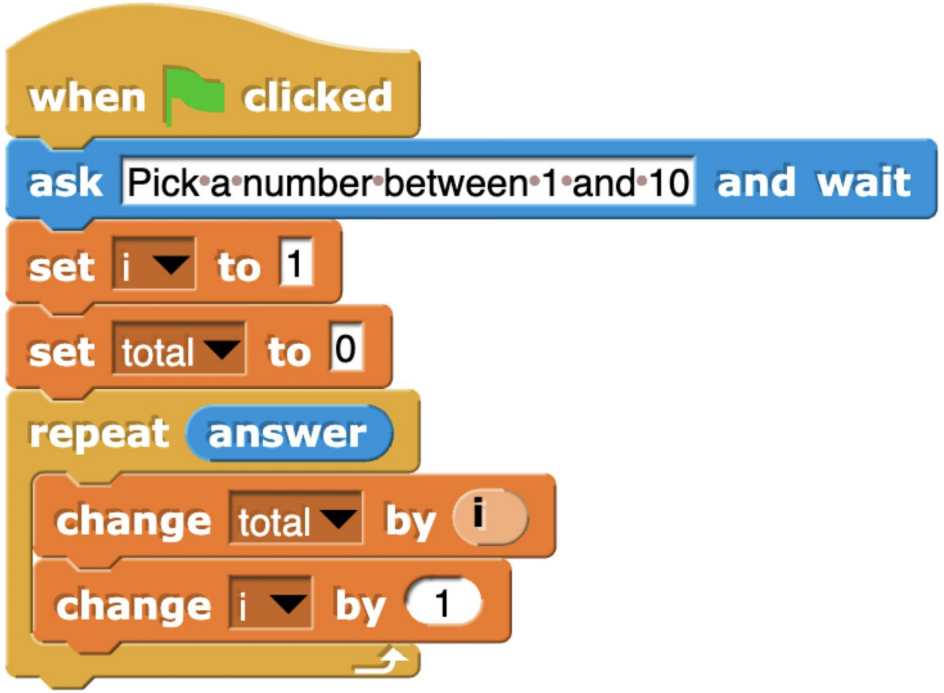
A. 1

B. 2

C. 3

D. 4

E. 5



Q: What is the value in i when the code is run, assuming user input = 3?

iClicker

```
when [flag] clicked
ask "Pick a number between 1 and 10" and wait
set i to 1
set total to 0
repeat answer
    change total by i
    change i by 1
```

Iteration	i (Before)	total (Before)	Add i to total	total (After)	i +1	i (After)
1st	1	0	0 + 1	1	+1	2
2nd	2	1	1 + 2	3	+1	3
3rd	3	3	3 + 3	6	+1	4

Q: Will this program ever say “I still haven’t found what I’m looking for”?

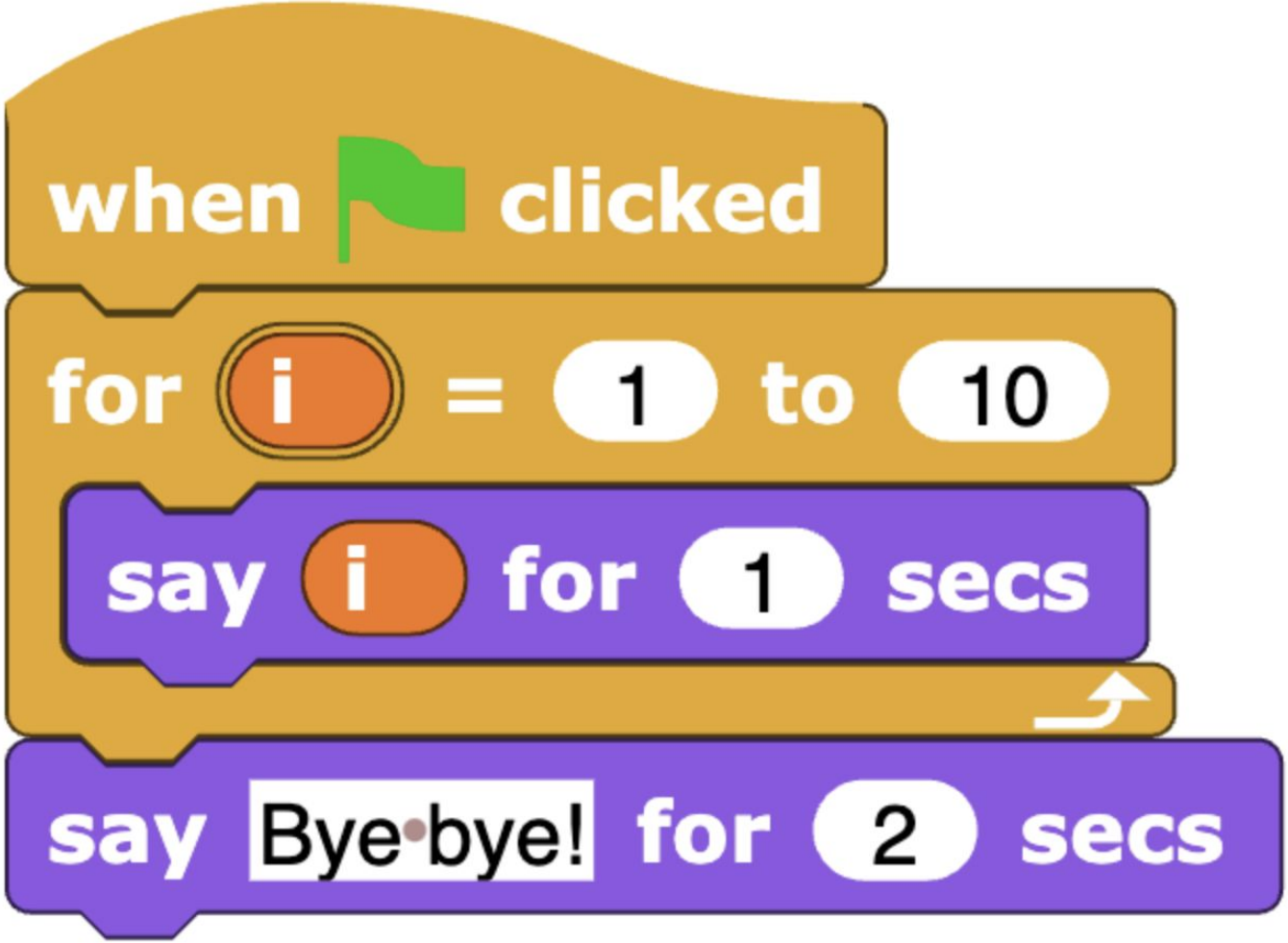
iClicker

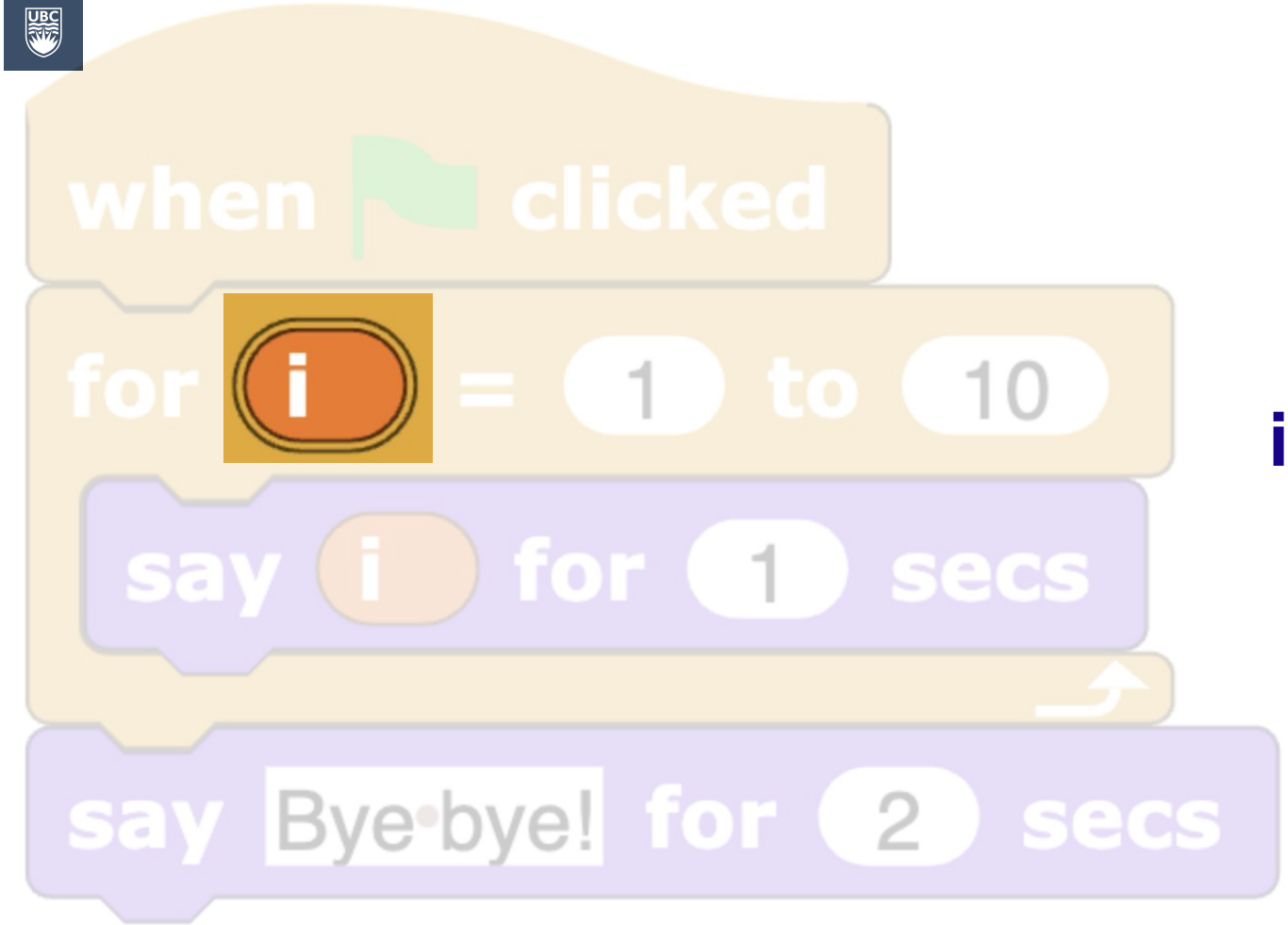
A. Yes

B. No

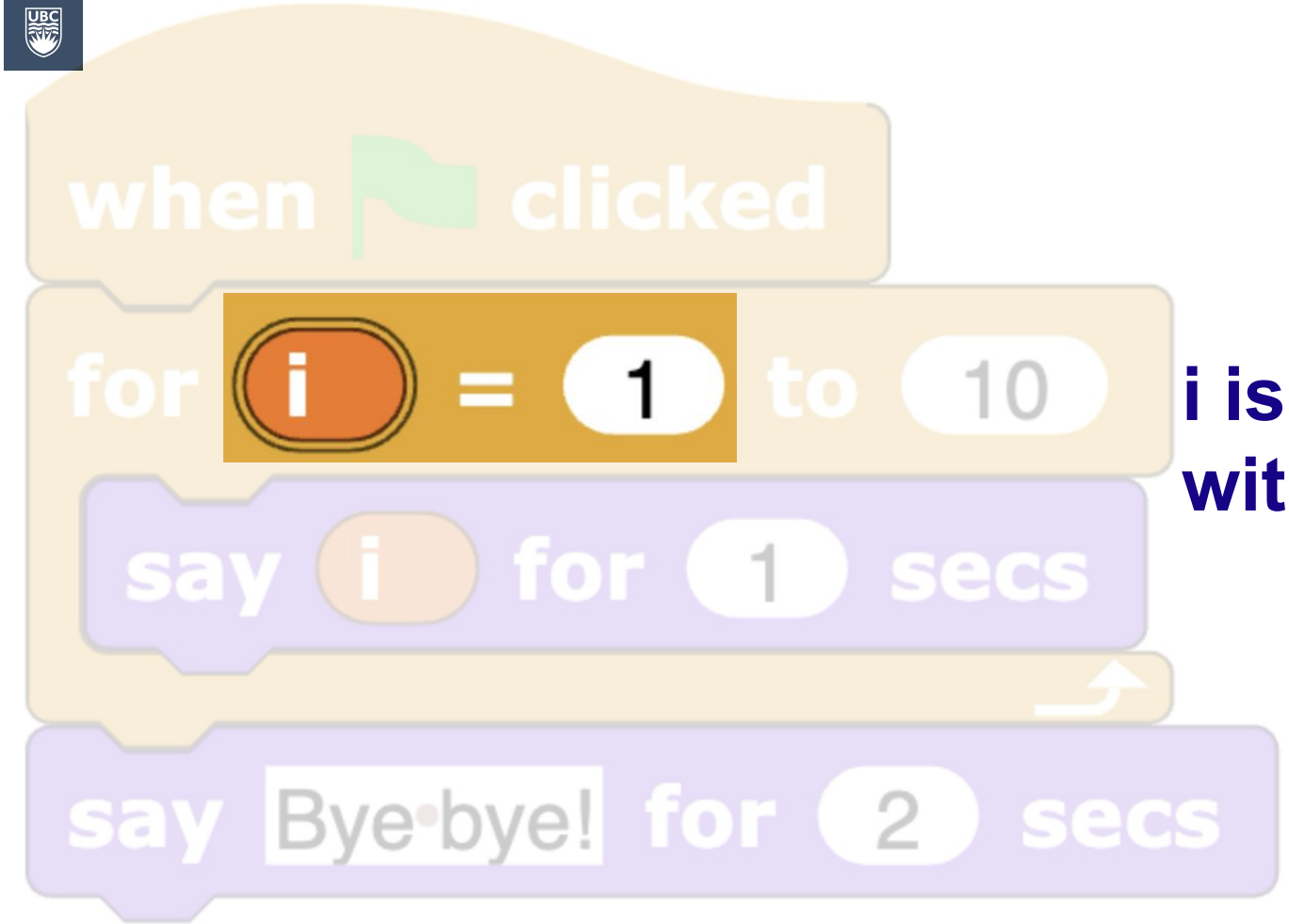
C. Sometimes

For loops

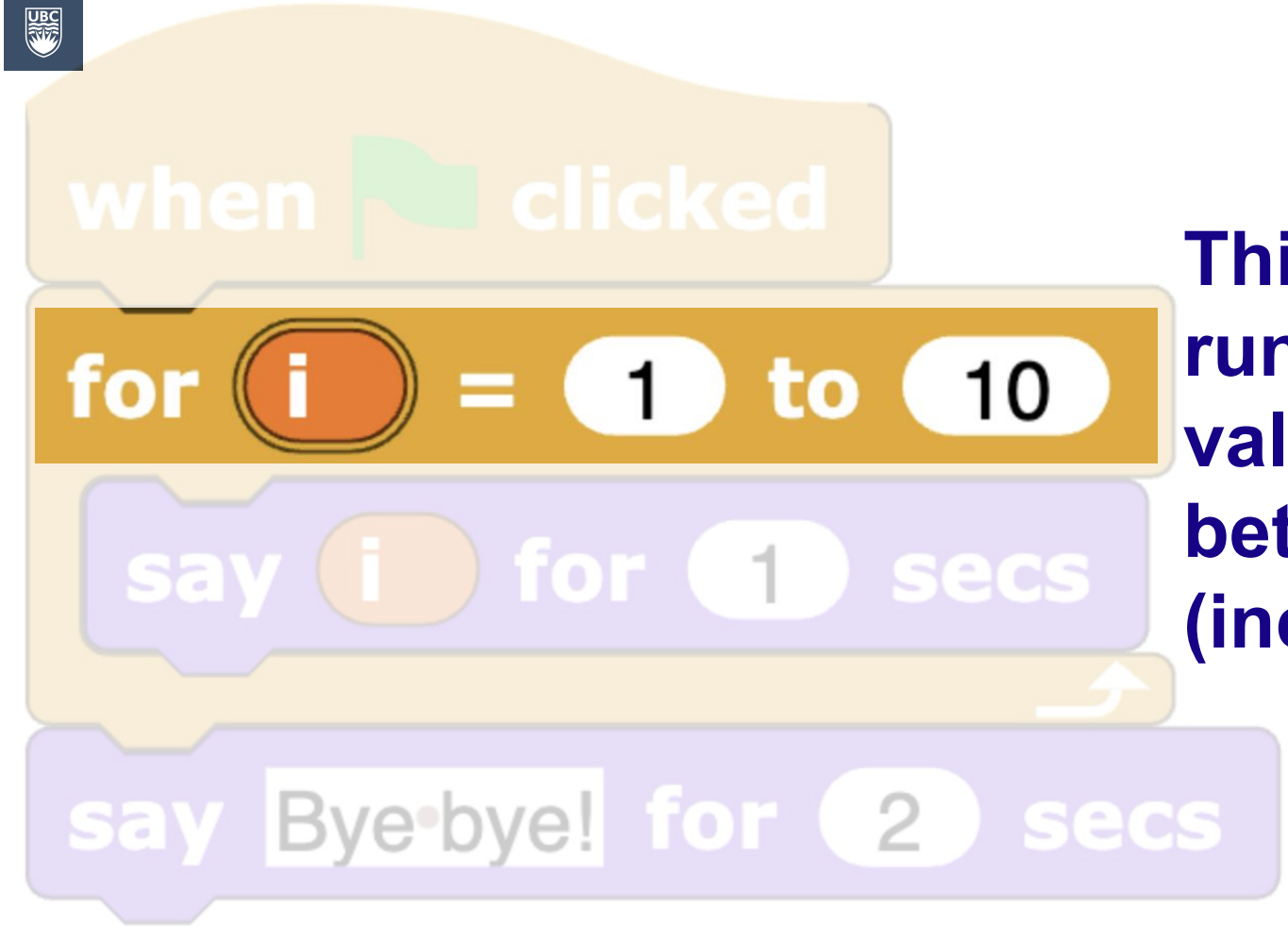

when clicked
for i = 1 to 10
say i for 1 secs
say Bye bye! for 2 secs



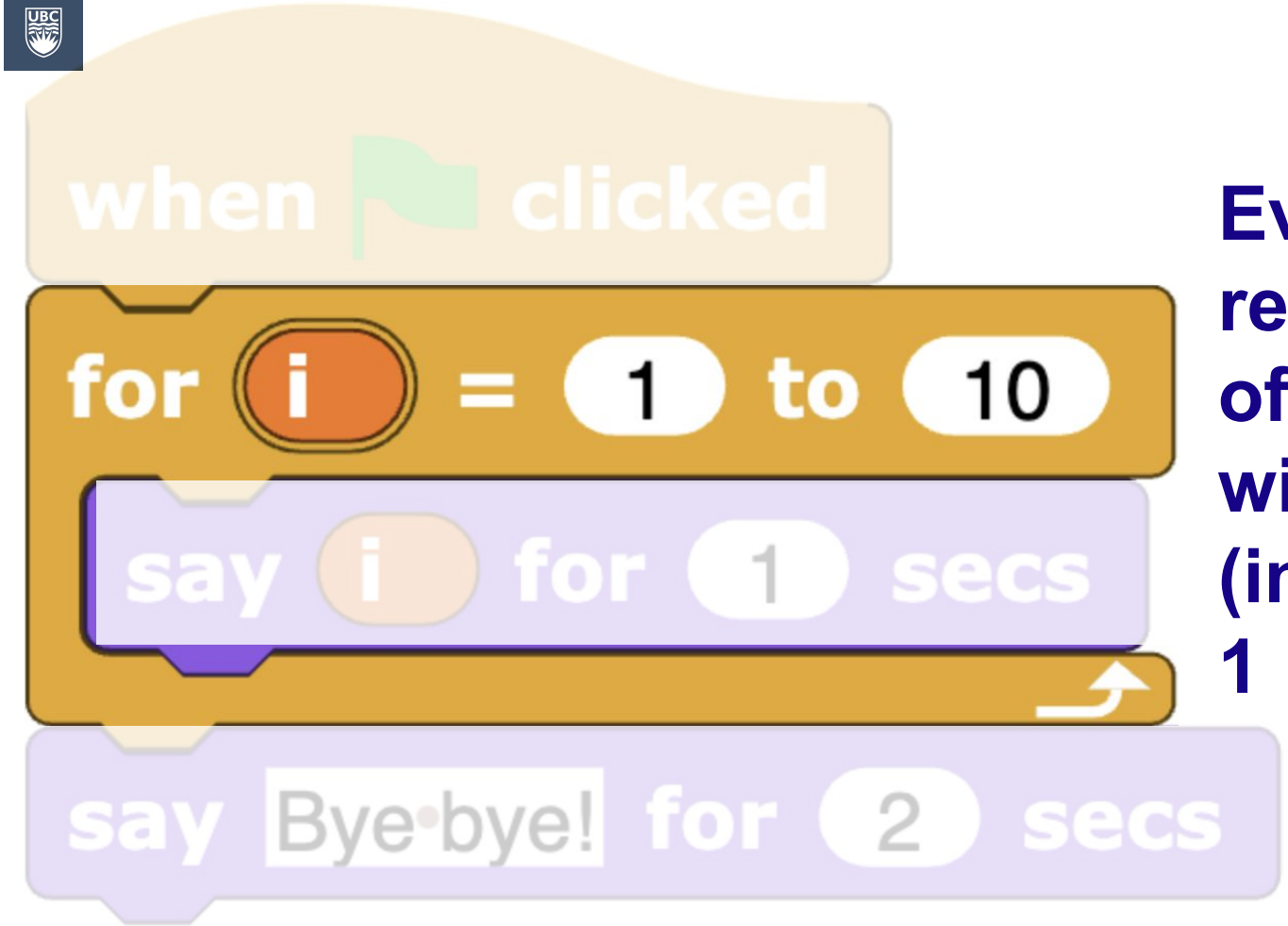
i is a variable



i is initialized with a value of 1



This loop will run until i has a value that is not between 1 to 10 (inclusive)



Every time we reach the end of the loop, i will increase (increment) by 1

Q: What is the value in x when the code is run?

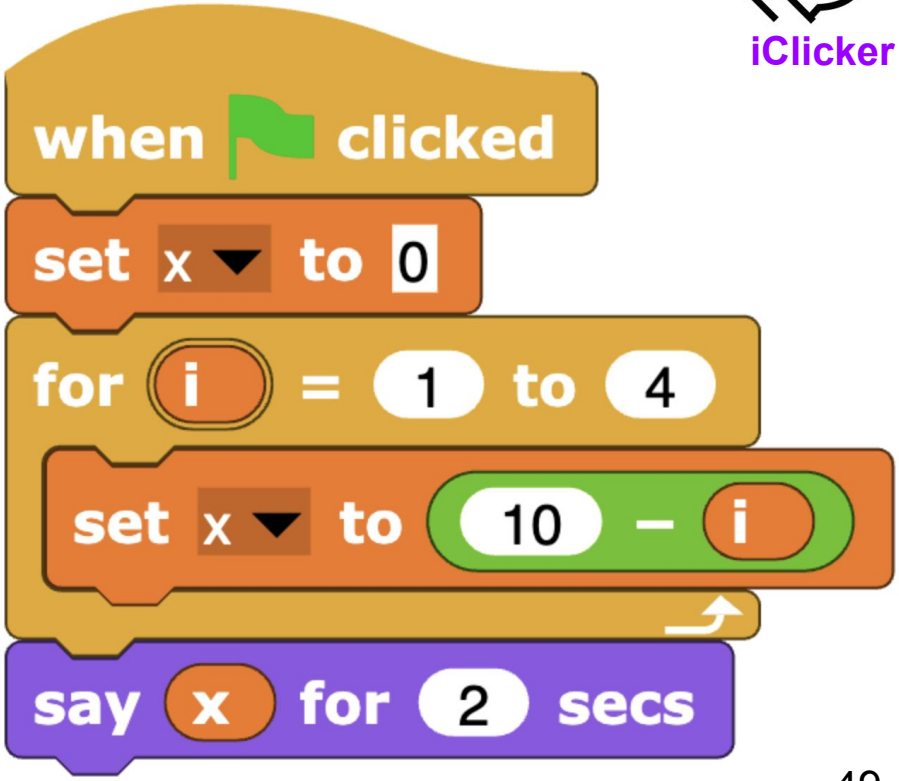
A. 1

B. 3

C. 4

D. 6

E. 10



Loops

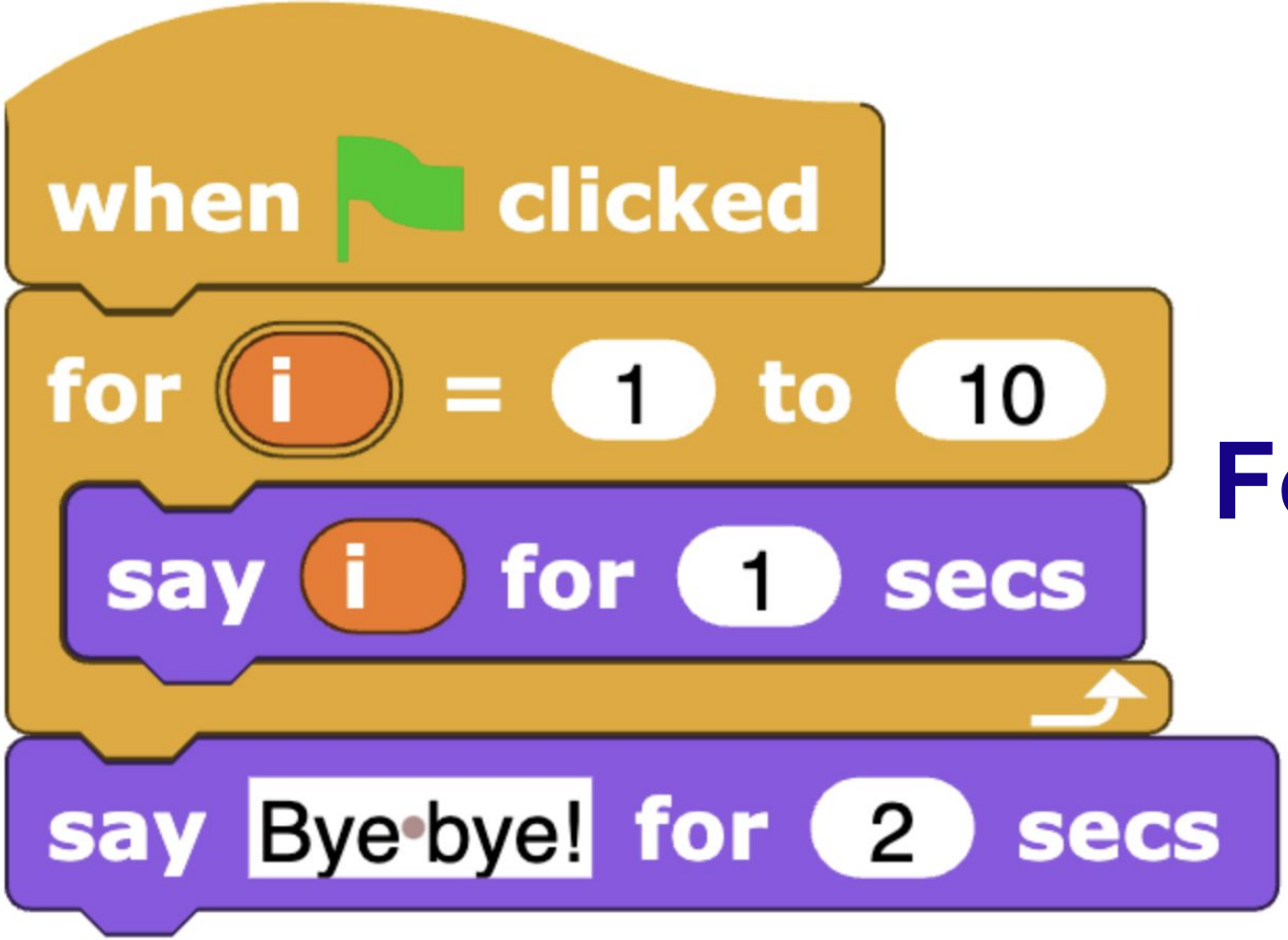
Recall

Repeat Blocks

Repeat some code a finite number of time

Repeat UNTIL a particular condition has been met.

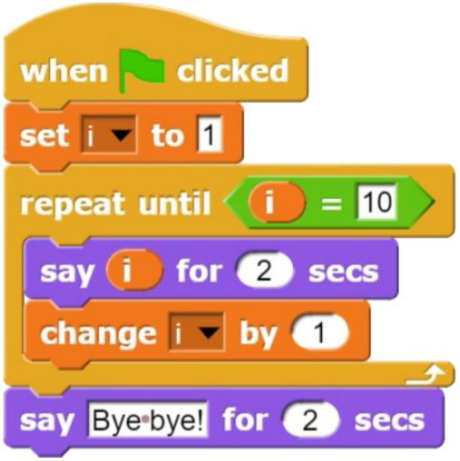
If the condition is never met, then, it goes on *forever.*

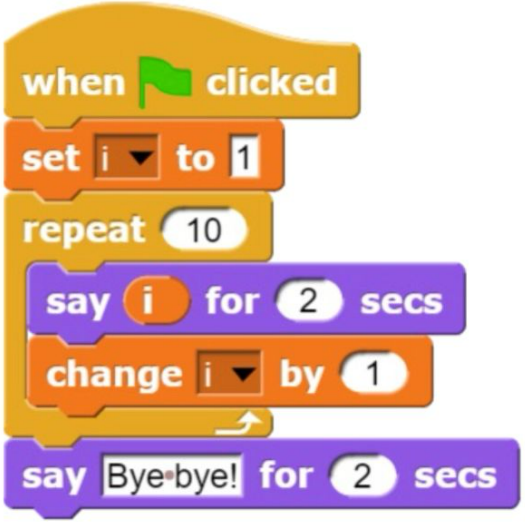


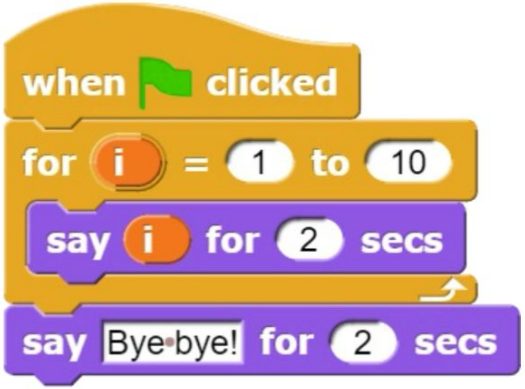
For Loops

Comparing Loops

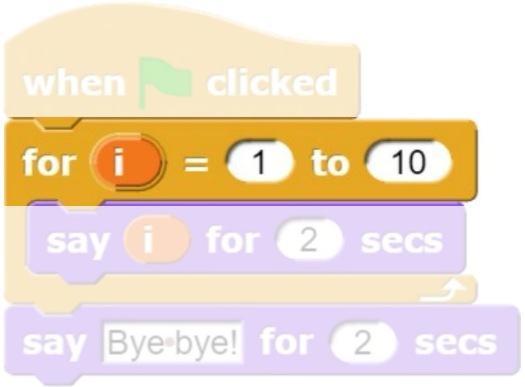
What's the difference between these loops?







What's the difference between these loops?



Wrap up