

# Using loops to solve problems

---

Introduction to Computer Science, Fall 2019

Carolyn Jane Anderson

# How do you know when to use a loop?

Loops are useful when you want to repeat an instruction multiple times.

## Goals for today:

- ❖ Break problems down into sequences of steps
- ❖ Translate sequences of steps into code using loops

## Key concepts:

For loop, while loop, range, accumulator, precondition, loop invariant, termination condition

# Review: Loop syntax

Last week we learned the syntax for two kinds of loops in Python:

for-loops

and

while loops

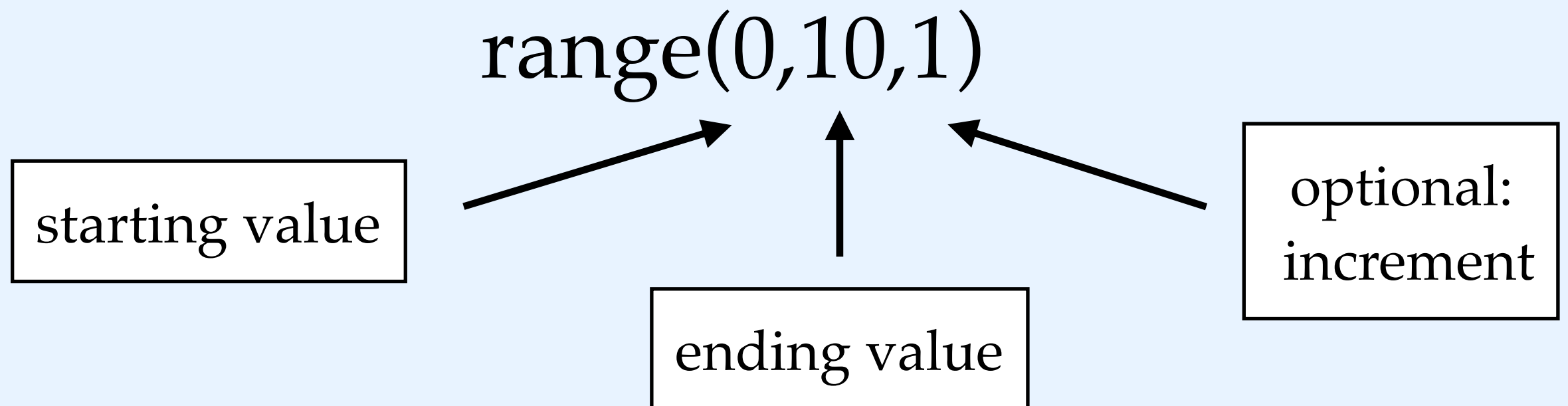
```
for i in range(0,10):  
    print(i)
```

```
i = 0  
while(i < 10):  
    print(i)  
    i = i+1
```

# For loops

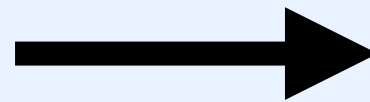
For loops run for a specified number of iterations. They take a sequence as an argument, and run once for every item in the sequence.

To run a set number of times, this argument should be a **range**:



# For loops

```
for i in range(0,10):  
    print(i)
```



0  
1  
2  
3  
4  
5  
6  
7  
8  
9

# While loops

While take a single argument and run until the argument evaluates to false.

```
x = 0
while(x < 10):
    print(x)
    x += 1
```

```
while(true):
    ...
```

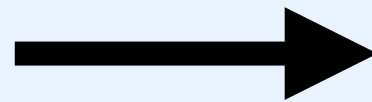
```
while(false):
    ...
```

runs forever!

never runs!

# While loops

```
x = 0  
while(x < 10):  
    print(i)  
    x += 1
```



0  
1  
2  
3  
4  
5  
6  
7  
8  
9

# How to use loops

Let's think about a game of tic-tac-toe.

How would we break the game down into a sequence of activities?

X	X	O
	O	



# How to use loops: key questions

What needs to happen before the loop?

What happens at each step through the loop?

When should the loop stop?

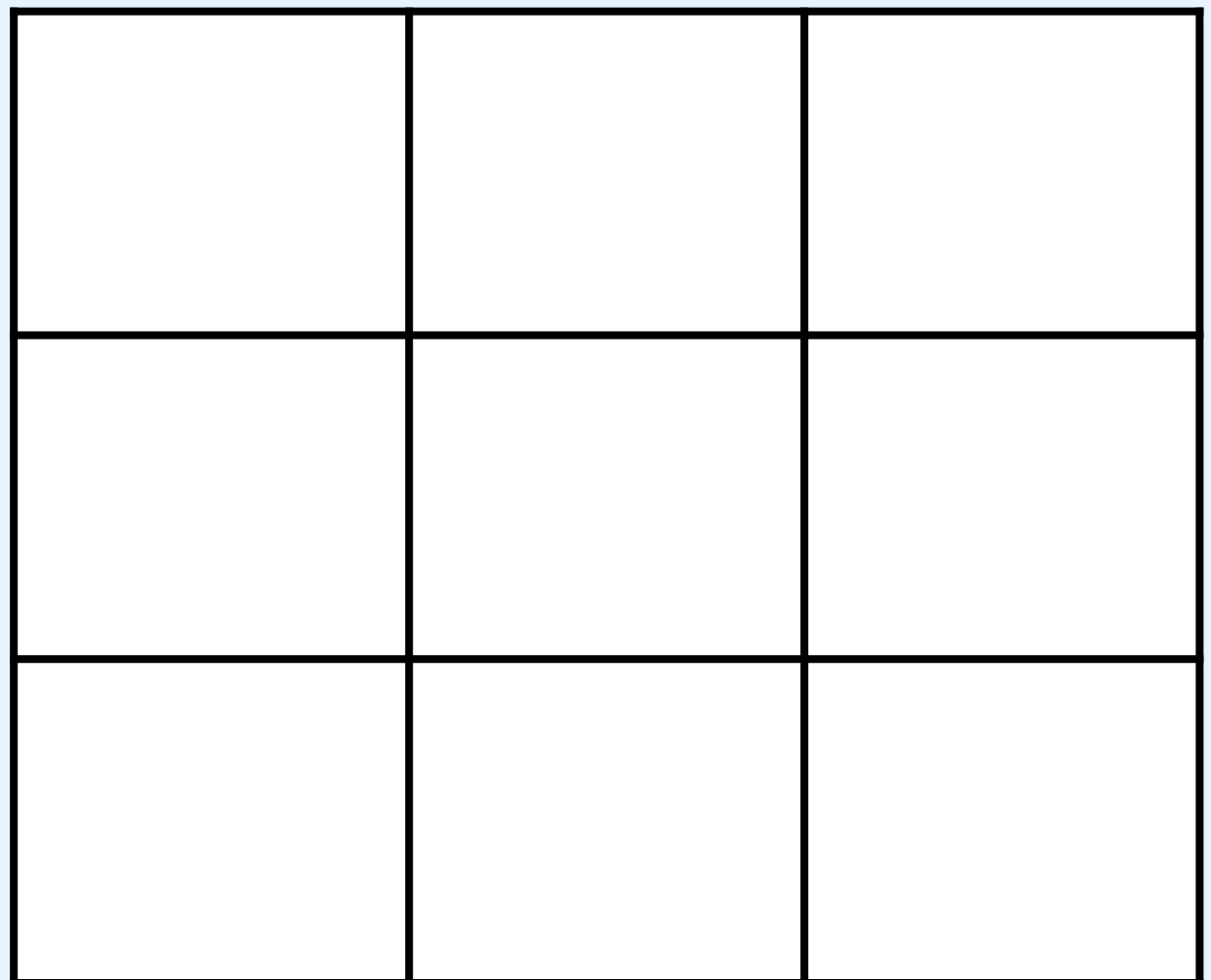
What kind of loop should be used?

# How to use loops: tic-tac-toe

First, what do we need to start a game of tic-tac-toe?

X X X  
X X

O O  
O O O



# How to use loops: tic-tac-toe

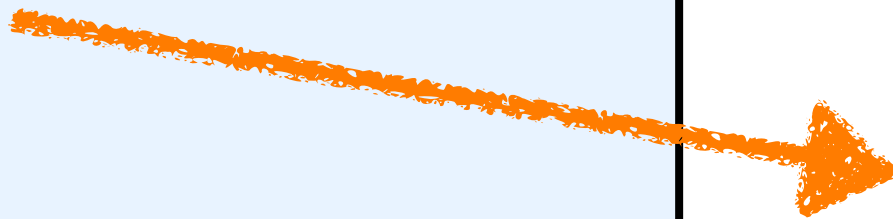
First, what do we need to start a game of tic-tac-toe?

- ❖ A blank tic-tac-toe board
- ❖ 2 players

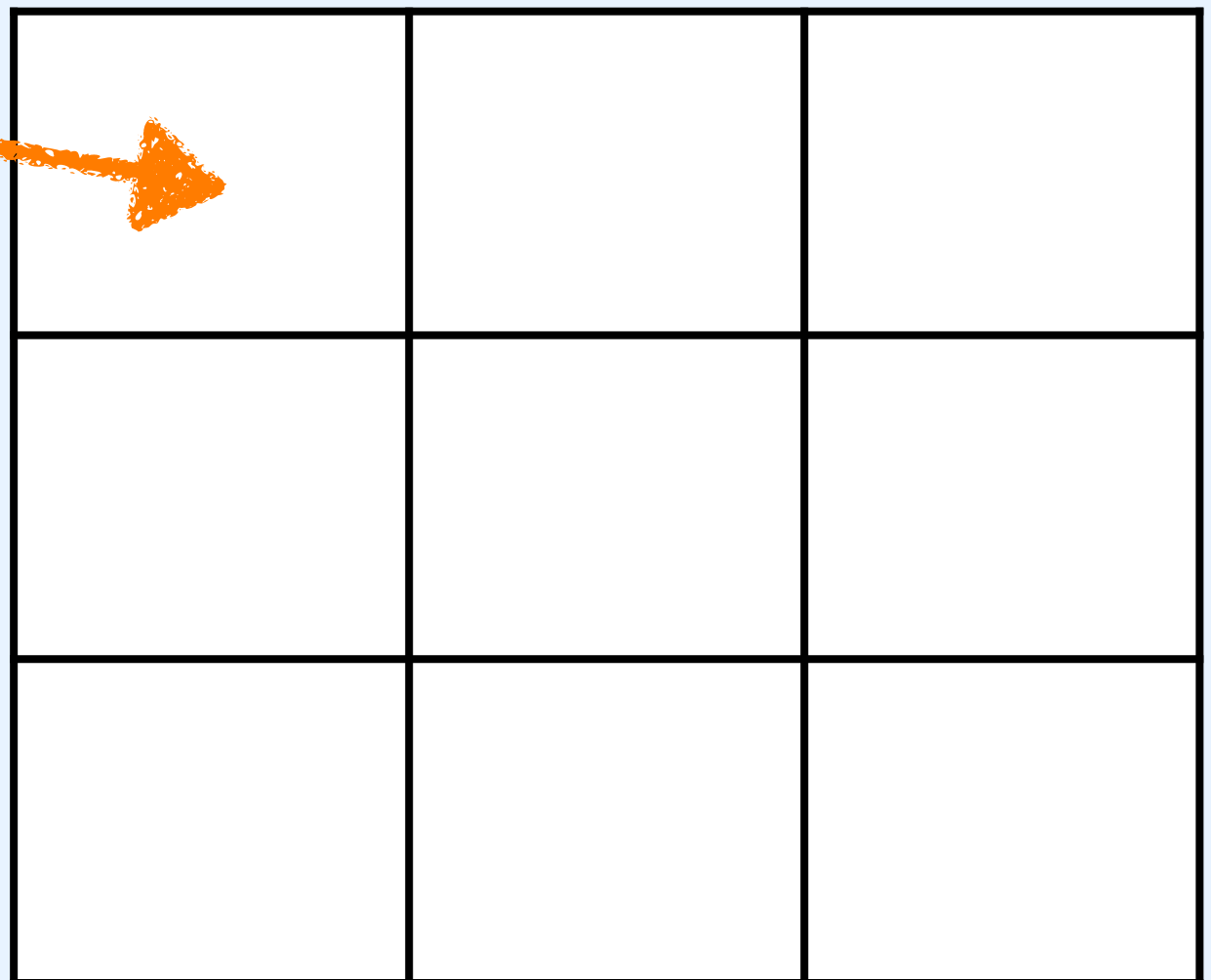
# How to use loops: tic-tac-toe

Next, what happens during each turn in the game?

X



O



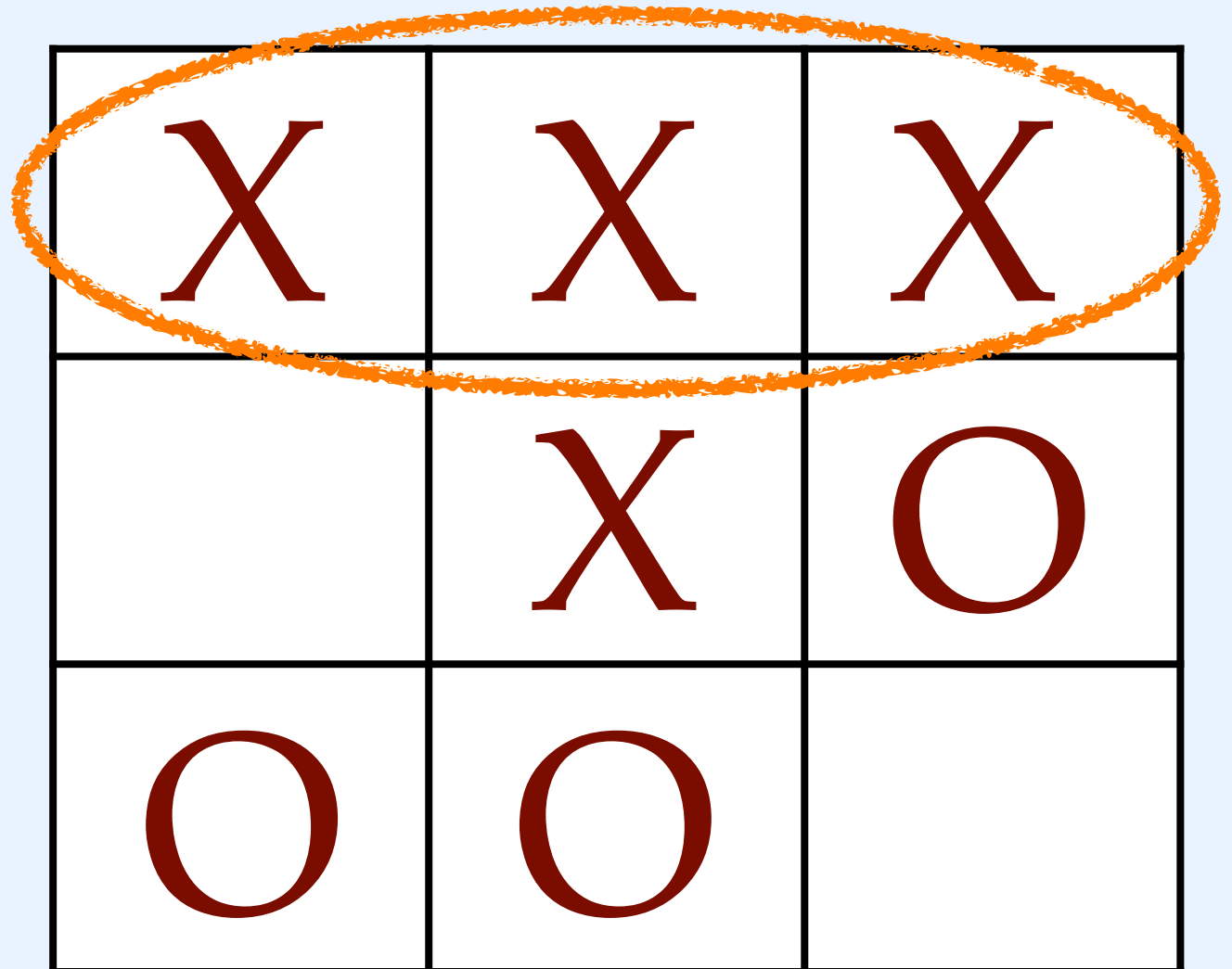
# How to use loops: tic-tac-toe

Next, what happens during each turn in the game?

- ❖ The current player puts a marker on an empty square
- ❖ The players switch turns

# How to use loops: tic-tac-toe

Last, how do we know when a game is over?



X	X	X
	X	O
O	O	

A 3x3 tic-tac-toe grid is shown. The top row contains three 'X' characters and is circled with a thick orange oval, indicating a winning condition for 'X'. The middle row contains an empty cell, an 'X', and an 'O'. The bottom row contains an 'O', another 'O', and an empty cell.

# How to use loops: tic-tac-toe

Last, how do we know when a game is over?

- ❖ If there are no more empty squares
- ❖ If one player gets 3 in a row

# Steps for tic-tac-toe

Before the loop:

- ❖ Create a new game board
- ❖ Set player 1
- ❖ Set player 2
- ❖ Set current player = player 1



# Steps for tic-tac-toe

During the loop:

- ❖ Current player makes a move
- ❖ Check if game is over
- ❖ Update current player

# Steps for tic-tac-toe

End the loop if:

- ❖ There are no more spaces
- ❖ A player wins

# Steps for tic-tac-toe

Before the loop:

- ❖ Create a new game board
- ❖ Set player 1
- ❖ Set player 2
- ❖ Set current player = player 1

During the loop:

- ❖ Current player makes a move
- ❖ Check if game is over
- ❖ Update current player

End the loop if:

- ❖ There are no more spaces
- ❖ A player wins

What kind of loop  
should we use?

# Steps for tic-tac-toe

Before the loop:

- ❖ Create a new game board
- ❖ Set player 1
- ❖ Set player 2
- ❖ Set current player = player 1

During the loop:

- ❖ Current player makes a move
- ❖ Check if game is over
- ❖ Update current player

End the loop if:

- ❖ There are no more spaces
- ❖ A player wins

What kind of loop  
should we use?

# Tic-tac-toe in pseudo-code

create new game board

set player 1

set player 2

current player = player 1

winner = false

while(board has spaces and winner==false):

    current player makes a move

    if 3 in a row:

        winner = current player

    if current player==player 1:

        current player==player 2

    else:

        current player==player 1

if winner==false:

    print("Tie!")

else:

    print(winner)

# Tic-tac-toe in pseudo-code

create new game board

set player 1

set player 2

current player = player 1

**winner = false**

while(board has spaces and winner==false):

    current player makes a move

    if 3 in a row:

        winner = current player

    update current player

if winner is false:

    report tie

else:

    print(winner)

# How to use loops: key questions

What needs to happen before the loop?

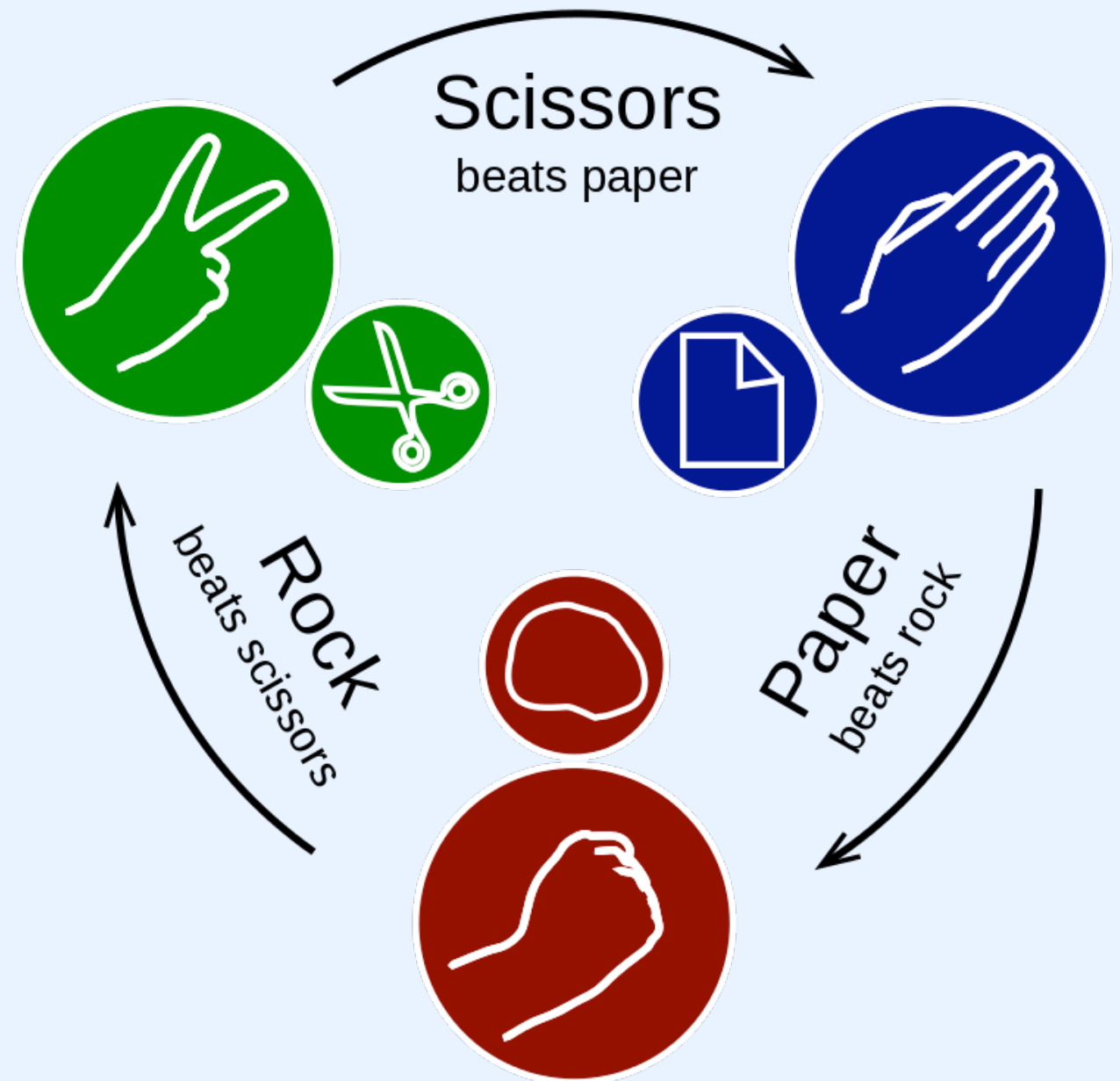
What happens at each step through the loop?

When should the loop stop?

What kind of loop should be used?

# How to use loops: rock-paper-scissors

Let's consider another simple game: a best two out of three rock-paper-scissors tournament.





# Rock-paper-scissors: 3 game tournament

Let's consider another simple game: a best two out of three rock-paper-scissors tournament.

What needs to happen **before the loop**?

What happens at **each step** through the loop?

When should the loop **stop**?

What **kind of loop** should be used?

# Steps for tic-tac-toe

Before the loop:

- ❖ Create a new game board
- ❖ Set player 1
- ❖ Set player 2
- ❖ Set current player = player 1

During the loop:

- ❖ Current player makes a move
- ❖ Check if game is over
- ❖ Update current player

End the loop if:

- ❖ There are no more spaces
- ❖ A player wins

# Rock-paper-scissors: 3 game tournament

Let's consider another simple game: a best two out of three rock-paper-scissors tournament.

What needs to happen **before the loop**?

What happens at **each step** through the loop?

When should the loop **stop**?

What **kind of loop** should be used?

# Rock-paper-scissors: 3 game tournament

Let's consider another simple game: a best two out of three rock-paper-scissors tournament.

What **kind of loop** should be used?

# Rock-paper-scissors: 3 game tournament

Let's consider another simple game: a best two out of three rock-paper-scissors tournament.

What needs to happen **before the loop**?

# Rock-paper-scissors: 3 game tournament

Let's consider another simple game: a best two out of three rock-paper-scissors tournament.

What happens at **each step** through the loop?

# Rock-paper-scissors: 3 game tournament

Let's consider another simple game: a best two out of three rock-paper-scissors tournament.

When should the loop **stop**?

# Rock-paper-scissors: 3 game tournament

What needs to happen **before the loop**?

- ❖ Set up the 2 players
- ❖ Set the score to 0

What happens at **each step** through the loop?

- ❖ Get player 1 move
- ❖ Get player 2 move
- ❖ Compare
- ❖ Update score

When should the loop **stop**?

- ❖ After 3 games



# Rock-paper-scissors in pseudo-code

```
set player 1
set player 2
score = 0
for i in range(0,3):
    move1 = get player 1 move
    move2 = get player 2 move
    if move1 beats move2:
        score = score + 1
    else:
        score = score - 1
if score > 0:
    print "Player 1 wins!"
else:
    print "Player 2 wins!"
```

# Rock-paper-scissors in pseudo-code

```
set player 1
set player 2
score = 0
for i in range(0,3):
    move1 = get player 1 move
    move2 = get player 2 move
    if move1 beats move2:
        score = score + 1
    elif move2 beats move1:
        score = score - 1
if score > 0:
    print "Player 1 wins!"
elif score < 0:
    print "Player 2 wins!"
else:
    print "Tie!"
```

# Rock-paper-scissors in pseudo-code

```
set player 1
set player 2
score = 0
winner = false
for i in range(1,4):
    move1 = get player 1 move
    move2 = get player 2 move
    if move1 beats move2:
        score = score + 1
    else:
        score = score - 1
if score > 0:
    print "Player 1 wins!"
else:
    print "Player 2 wins!"
```

Accumulator: a variable that keeps track of the result of the computation

# Loop Terminology

## Accumulator:

A variable that keeps track of the result of the computation

## Precondition:

What is true before the loop executes.

## Loop invariant:

What is true at the beginning and end of each iteration through the loop.

## Termination condition:

The stopping conditions for the loop.

# Analyzing tic-tac-toe

## Precondition:

Game board is empty

Current player is player 1

## Loop invariant:

There are 1 or fewer 3-in-a-rows

## Termination condition:

There are no more empty spaces

There is a 3-in-a-row

# How to use loops: key questions

What needs to happen before the loop?

What happens at each step through the loop?

When should the loop stop?

What kind of loop should be used?

# Writing programs with loops

How would we write a string-reverse function using a loop?

What kind of loop should we use?

# Writing programs with loops

How would we write a string-reverse function using a **for loop**?

Before the loop:

During the loop:

Termination condition:



# String-reverse with a for loop

Before the loop:

- ❖  $s1$  = string to reverse
- ❖  $s2$  = empty string



Accumulator

During the loop:

- ❖ Take the  $i$ th letter of  $s1$  and prepend it to  $s2$

Termination condition:

- ❖ When  $i == \text{length}(s1)$

# Writing programs with loops

How would we write a string-reverse function using a **while loop**?

Before the loop:

During the loop:

Termination condition:

# String-reverse with a while loop

Before the loop:

- ❖ s1 = string to reverse
- ❖ s2 = empty string



Accumulator

During the loop:

- ❖ Take the 1st letter of s1 and prepend it to s2
- ❖ Delete the 1st letter of s1

Termination condition:

- ❖ If s1 is empty

# Reasoning about Loops

What needs to happen **before the loop**?

What happens at **each step** through the loop?

When should the loop **stop**?

What **kind of loop** should be used?

# Loop Terminology

## Accumulator:

A variable that keeps track of the result of the computation

## Precondition:

What is true before the loop executes.

## Loop invariant:

What is true at the beginning and end of each iteration through the loop.

## Termination condition:

The stopping conditions for the loop.

# Wrap up

Loops are useful when you want to repeat an instruction multiple times.

## Goals for today:

- ❖ Break problems down into sequences of steps
- ❖ Translate sequences of steps into code using loops

## Key concepts:

For loop, while loop, range, accumulator, precondition, loop invariant, termination condition

# String-reverse in place

How would we write a string-reverse function that reverses the string in place (no accumulator)?

What needs to happen **before the loop**?

What happens **at each step** through the loop?

When should the loop **stop**?


What **kind of loop** should be used?

# String-reverse in place

How would we write a string-reverse function that reverses the string in place (only uses one variable)?

During the loop (version 1):

- ❖ Move the last letter to the front

G O O S **E**            **E** G O O S



# String-reverse in place

How would we write a string-reverse function that reverses the string in place (only uses one variable)?

During the loop (version 1):

- ❖ Move the last letter to the front

G O O S **E**            **E** G O O S

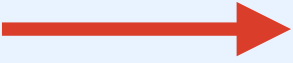
E G O O **S**            **S** E G O O

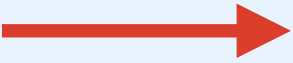
# String-reverse in place

How would we write a string-reverse function that reverses the string in place (only uses one variable)?

During the loop (version 2):

- ❖ Move the last letter to the end of the part of the string that is already reversed

G O O S **E**            **E** G O O S

E G O O **S**            E **S** G O O

# String-reverse in place

How would we write a string-reverse function that reverses the string in place (only uses one variable)?

During the loop (version 2):

- ❖ Move the last letter to the end of the part of the string that is already reversed (at nth step, this is the nth position)

Step 0	G O O S <b>E</b>	→	<sup>0 1</sup> <b>E</b> G O O S
Step 1	E G O O <b>S</b>	→	E <b>S</b> G O O

# String-reverse in place

How would we write a string-reverse function that reverses the string in place (only uses one variable)?

Termination condition:

- ❖ When we've moved all the letters

# String-reverse in place

How would we write a string-reverse function that reverses the string in place (only uses one variable)?

Termination condition:

- ❖ When we've moved all the letters (when number of iterations == length of string)

# String-reverse in place

How would we write a string-reverse function that reverses the string in place (only uses one variable)?

Precondition:

- ❖ String to reverse

# String-reverse in place

How would we write a string-reverse function that reverses the string in place (only uses one variable)?

Precondition:

- ❖ String to reverse

Loop invariant:

- ❖ Move the last letter to the nth position in the string

Termination condition:

- ❖ Number of iterations == length of string