

# CompSci 94

## Writing your own Functions

### October 22, 2024



Prof. Susan Rodger

# Announcements

- Canvas QZ and videos for next Tuesday
- Assignment 4 due Oct 29
- More review for Exam 2
- Exam 2 on Oct 24
  - See old exams on calendar page
  - See notes from last time

# Review arrays

# Looping in Array – when and how to use each one

- For each in
- Each in together

# Looping in Array – when and how to use each one

- For each in
  - Use with an array, to get each item in the array to do something one at a time
- Each in together
  - Use with an array, for each item at the same time to do something

# Arrays

- How do you create an array?
- Where should you create an array?
- How do you access a value in an array?
- What is the advantage of using an array?

# Arrays

- How do you create an array?
  - Create a variable/property and check the box for array
- Where should you create an array?
  - In Scene Properties
- How do you access a value in an array?
  - With a loop variable in an array loop
- What is the advantage of using an array?
  - Issue one instruction and apply it to every element in the array

Now on to new material!  
Today material not on Exam 2



# Function vs Procedure

- What is the difference between a function and a procedure?

# Function vs Procedure

- What is the difference between a function and a procedure?
  - Procedure is something to do – turn, move, dance
  - Function is a calculated value – a number, an object, a direction
  - A function by itself is not very useful, a function has to be used in some way based on the type of value it calculates

Write a function called `tallerHeight` to compute the height of the tallest of two objects.

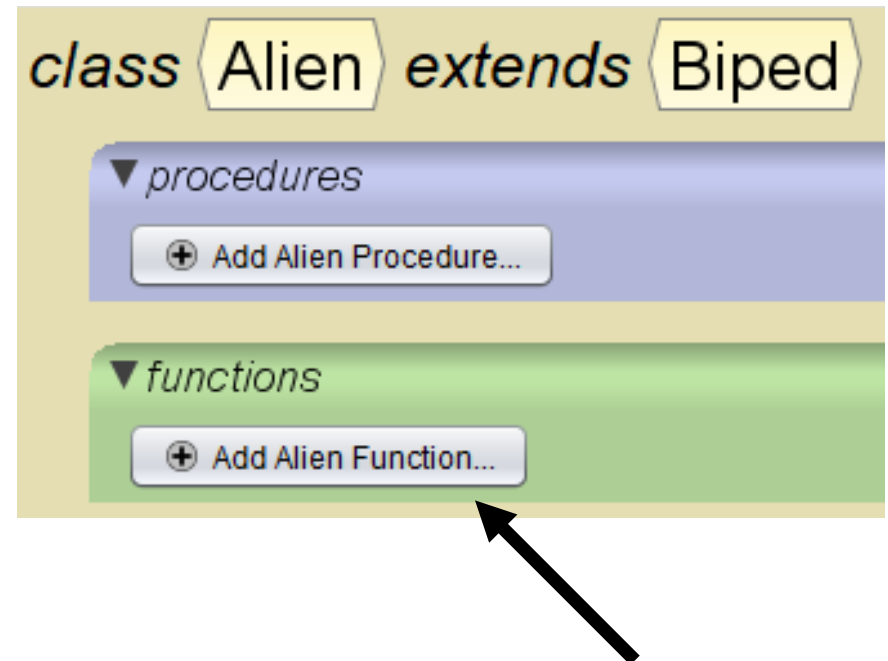
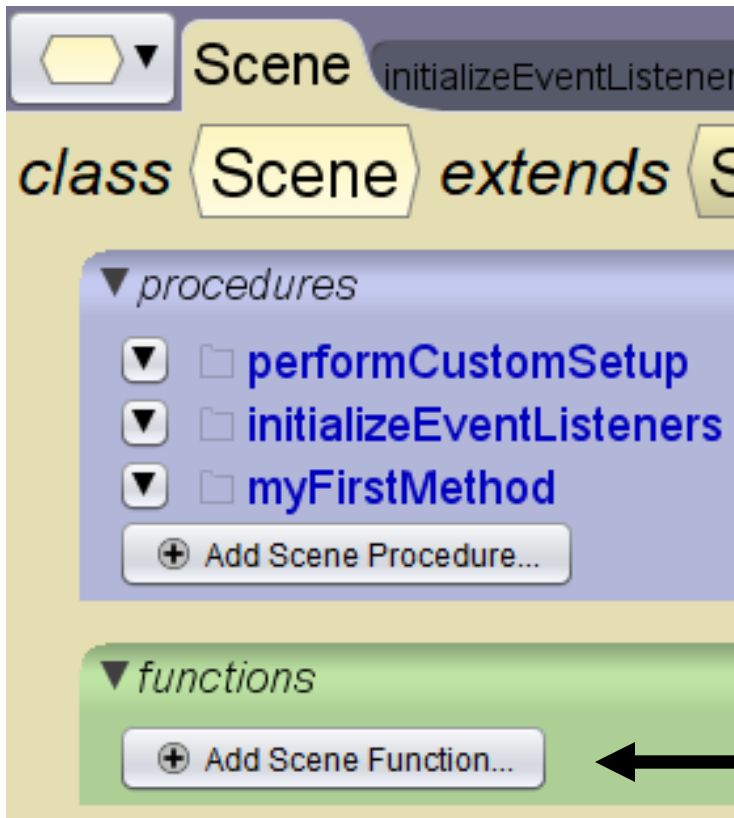
- What type of function should it be? Where do you create it?
- What is the return type?
- Need two parameters, what are their types?

Write a function called tallerHeight to compute the height of the tallest of two objects

- What type of function should it be? Where do you create it?
  - Scene function
    - Like to be able to use it for any two objects
- What is the return type?
  - DecimalNumber
- Need two parameters, what are their types?
  - SJointedModel
    - Then works for any creatures

# Can write your own functions

Function for Scene OR Function for character



Use scene function if it involves multiple objects

# Create Scene function tallerHeight

- Inputs: two objects
- Output (return value): the height of the taller object
- Return type: decimalNumber


Add Scene Function

preview: declare **DecimalNumber** function **tallerHeight**

return type: **DecimalNumber** ☐ is array

name: **tallerHeight**

# Parameters - SJointedModel

 Gallery Class

## Filtering

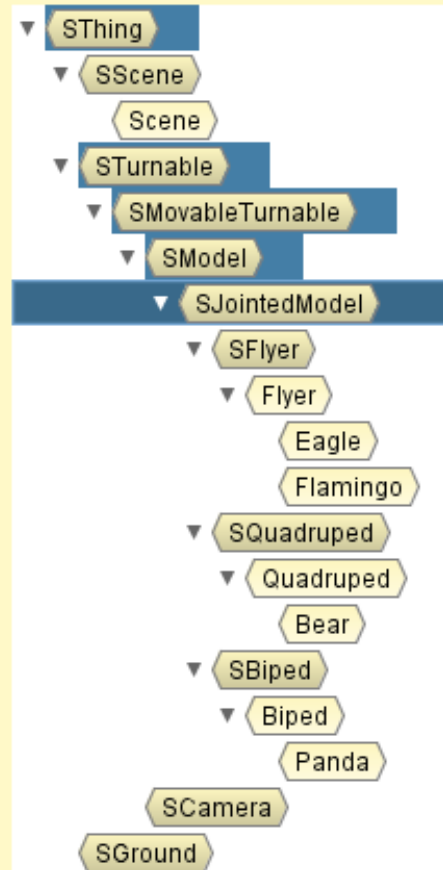
Assignable From **Contains**

Select class via the lowest common ancestor assignable from the items below:

myScene  
ground  
camera

☒ eagle  
☒ flamingo  
☒ bear  
☒ panda

## Selection



## Available Procedures, Functions, and Properties

### class SJointedModel

#### procedures

- straightenOutJoints
- say
- think

### class SModel (inherit)

#### procedures

- setVehicle
- setPaint
- setOpacity
- setWidth
- setHeight
- setDepth
- resize
- resizeWidth
- resizeHeight
- resizeDepth

#### functions

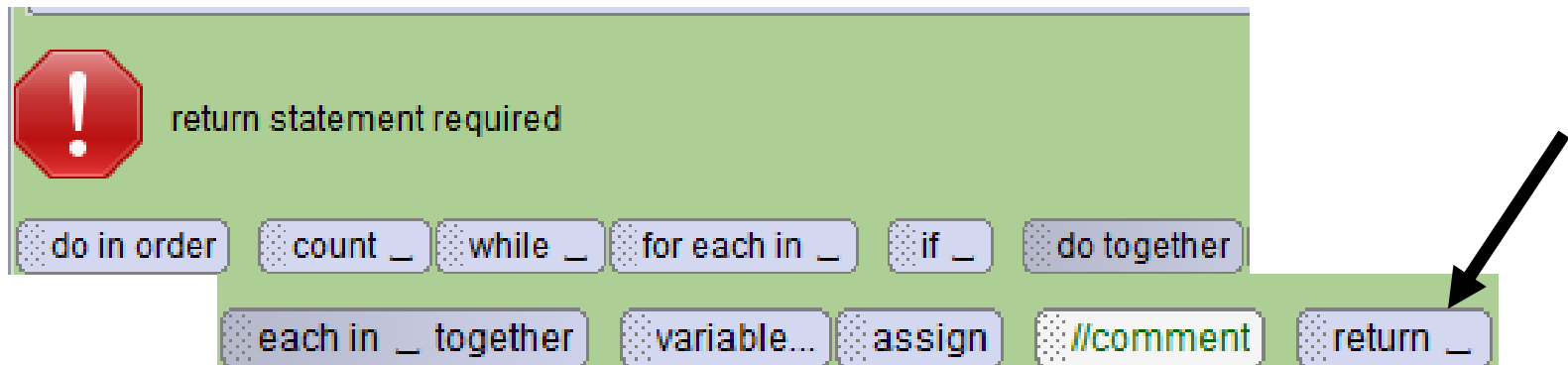
- getPaint
- getOpacity
- getWidth
- getHeight

Q1. What line of code do we have to put in every function?



# Q1. What line of code do we have to put in every function?

- Return statement!
  - Must return the same type as the specified return value.



# Q2 What is the code for tallerHeight?

# Q2 What is the code for tallerHeight?

The image shows a Scratch code editor interface. At the top, there is a tab bar with four tabs: 'Scene' (selected), 'initializeEventListeners', 'myFirstMethod', and 'tallerHeight' (which has a close button). Below the tabs, the code area is green and contains the following text:   
*declare* **DecimalNumber** *function* **tallerHeight**   
*with parameters:* **SJointedModel** **animal1** , **SJointedModel** **animal2** **Add Parameter...**

# Q2 What is the code for tallerHeight?

declare **DecimalNumber** function **tallerHeight**

with parameters: **SJointedModel** **animal1** , **SJointedModel** **animal2** Add Parameter...

do in order

- if **animal1** **getHeight** **>** **animal2** **getHeight** is true then
  - return **animal1** **getHeight**
- else
  - return **animal2** **getHeight**

# Q3 Given a bear and a flamingo, how does one use the function tallerHeight?

- Have panda say what the taller height is of the bear and flamingo.

# Q3 Given a bear and a flamingo, how does one use the function tallerHeight?

- Have panda say what the taller height is of the bear and flamingo.

```
declare procedure myFirstMethod
```

```
do in order
```

```
this.panda
```

```
say
```

```
"The taller height of bear and flamingo is "
```

```
+
```

```
this
```

```
tallerHeight
```

```
animal/1:
```

```
this.bear
```

```
,
```

```
animal/2:
```

```
this.flamingo
```

# Q3 Given a bear and a flamingo, how does one use the function tallerHeight?

- Have panda say what the taller height is of the bear and flamingo.

```
declare procedure myFirstMethod
```

```
do in order
```

```
this.panda say "The taller height of bear and flamingo is " +
```

```
this tallerHeight animal/1: this.bear , animal/2: this.flamingo
```

The taller height of bear and flamingo is 1.647032954975202



Q4. Write a function called tallerObject to return the object who is taller of two objects.

- What type of function should it be? Where do you create it?
- What is the return type?
- Need two parameters, what are their types?



Q4. Write a function called tallerObject to return the object who is taller of two objects.

- What type of function should it be? Where do you create it?
  - Scene function
    - Like to be able to use it for any two objects
- What is the return type?
  - SJointedModel
- Need two parameters, what are their types?
  - SJointedModel
    - Then works for any creatures

Q5 What is the code for tallerObject?

# Q5 What is the code for tallerObject?

*declare* SJointedModel *function* tallerObject

*with parameters:* SJointedModel creature1 , SJointedModel creature2 [Add Parameter...](#)

# Q5 What is the code for tallerObject?

The image shows a Scratch code editor with a green header bar and a light blue workspace. The code is as follows:

```
declare SJointedModel function tallerObject
with parameters: SJointedModel creature1 , SJointedModel creature2 Add Parameter...
do in order
  if (creature1.getHeight > creature2.getHeight) is true then
    return creature1
  else
    return creature2
```

The code defines a function named `tallerObject` that takes two parameters, `SJointedModel creature1` and `SJointedModel creature2`. The function's body is enclosed in a `do in order` block. Inside this block, there is an `if` statement that checks if `creature1.getHeight` is greater than `creature2.getHeight`. If the condition is true, the function returns `creature1`. Otherwise, it returns `creature2`.

Q6 How do you get the taller of the bear and flamingo to say they are taller using function tallerObject?

# Q6 How do you get the taller of the bear and flamingo to say they are taller using function tallerObject?

```
this tallerObject creature1: this.flamingo , creature2: this.bear say "I'm taller"
```



# Q7 How do you write code for ?

- The taller of the bear and flamingo to turn around once
- The bear to double in size (so it is taller)
- The taller of the bear and flamingo to turn around once.

# Q7 How do you write code for ?

do in order

`this` tallerObject creature1: `this.bear` , creature2: `this.flamingo` turn RIGHT 1.0

`this.bear` resize 2.0 add detail

`this` tallerObject creature1: `this.bear` , creature2: `this.flamingo` turn RIGHT 1.0

Use tallerObject function in place of an object.



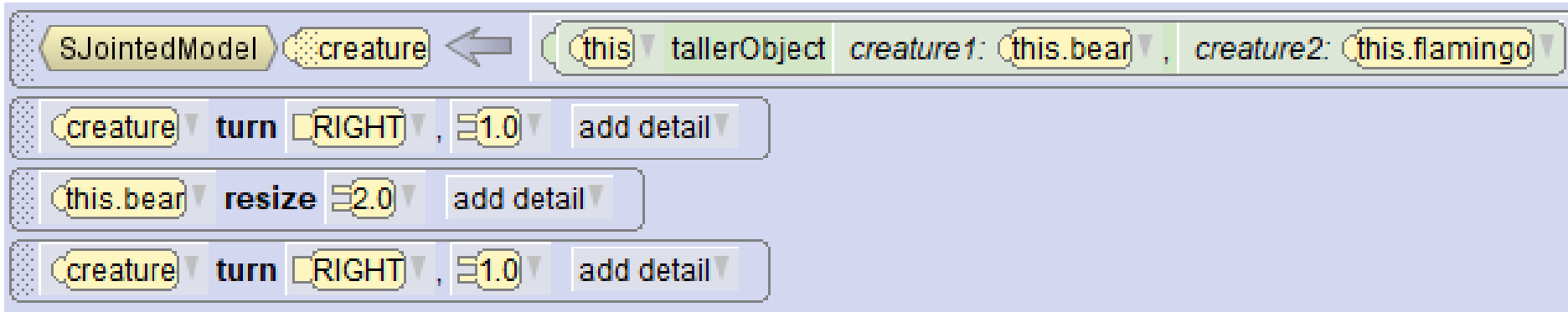
# Q7 When code runs...



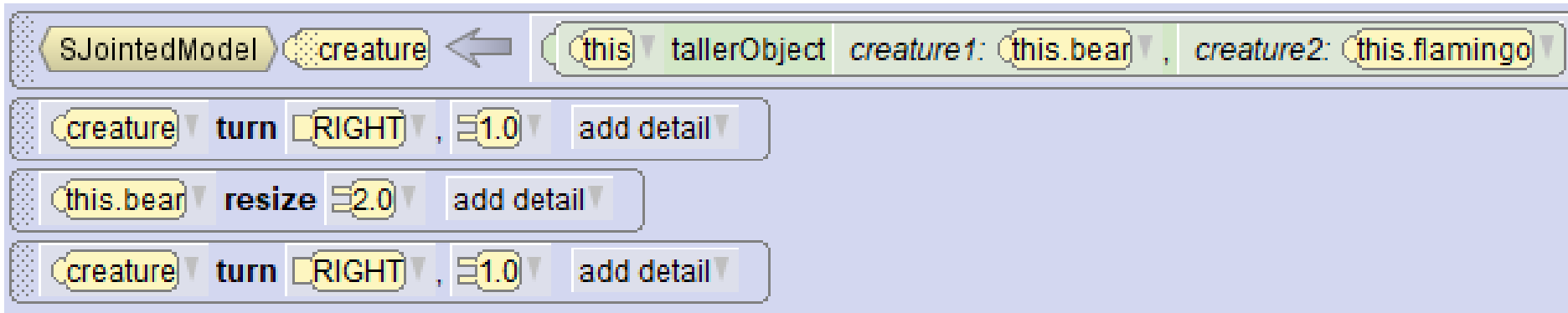
1. Flamingo turns
2. Bear gets bigger
3. Bear turns

# One more Question

# What does this code do?

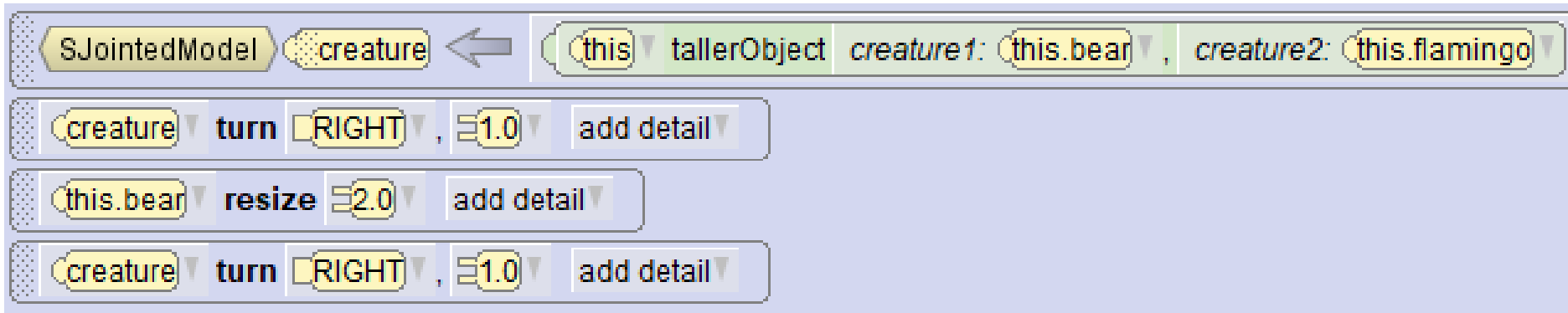


# What does this code do?



- The taller animal (flamingo) is stored in variable `creature`.
- Flamingo turns around, then **bear** gets bigger.
- Then Flamingo turns around again!

# What does this code do?



- In the last line if we want the taller of the two to turn around, we **MUST** call the function again to recalculate the taller one, since the bear changed its height

# Class Today

- Jumping cat calculating how high and how far to jump, and other things...

