Inheriting From Built-In’s and Overloading

Check In

What did you do over the long weekend? :)

Students, draw anywhere on this slide!

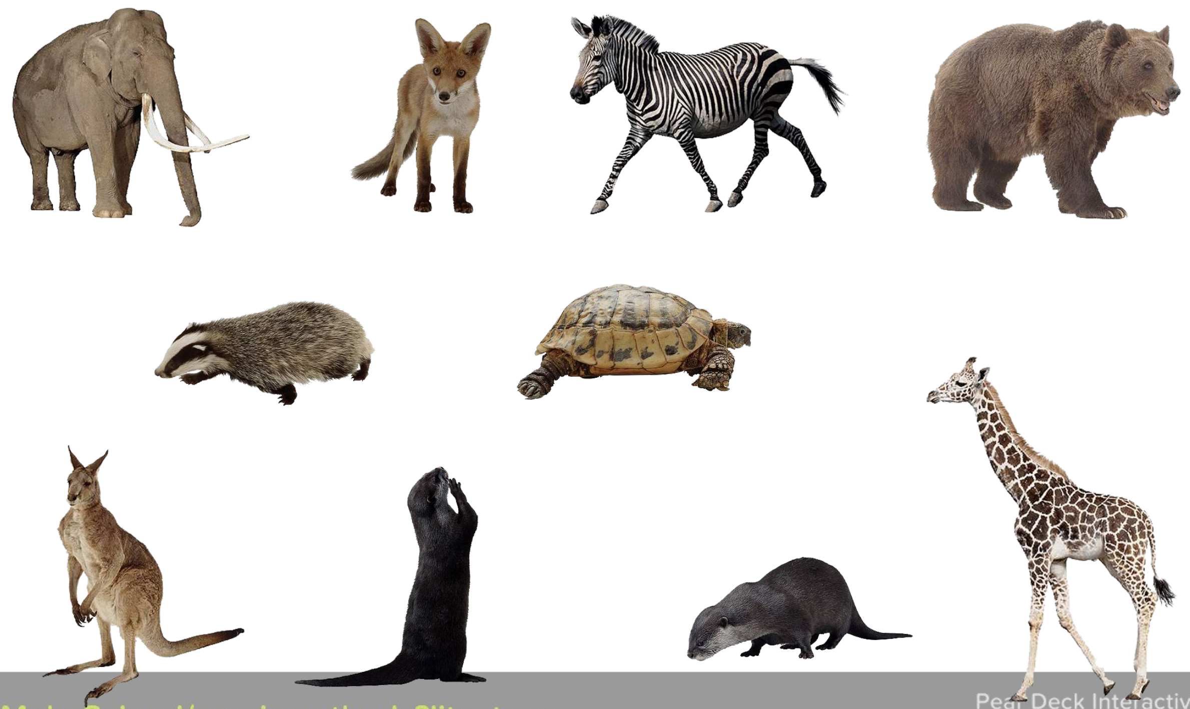
Pear Deck Interactive Slide
Do not remove this bar

What we’re going to learn

- Overriding Python magic methods
- The process of inheriting from python built-in’s
- Overloading in OOP

Python Magic Methods

What happens when we print out an Animal object? What do we need to do to make it look nicer?



Students browse: repl.it/@MakeSchool/magicmethods?lite=true

Pear Deck Interactive Slide
Do not remove this bar

Python Magic Methods

- Two prefix and suffix __ in the name
- __init__ is an example of a magic method we have used
- Magic methods are not meant to be called directly but we can call some of them if we want to
- Use dir(class_name) to see the method of a class and see how many are "magic methods"
- We can override the __str__() magic method to get something pretty to print out when we print a custom object
- [Here's a list of common magic methods]()

Python Magic Methods

Let’s override the __str__ method!

Students browse: repl.it/@MakeSchool/magicmethods?lite=true

Pear Deck Interactive Slide
Do not remove this bar

Python Magic Methods

MAKE SCHOOL

Let’s override the __add__ method!

Inheritance

Previously we learned about how we can inherit from superclasses that we defined

Today we are going to learn how we can inherit from python built in classes and classes from external libraries!

We can use our knowledge of overriding and magic methods to create custom functionality using code that was already written!

Inheriting from Built-In’s

Lists, dictionaries, data types, even functions are all represented by objects in Python!

As we have learned objects are instantiated using classes

We can actually look at the source code behind some of our favorite Python classes

Let’s look at a list!

String Only List

MAKE SCHOOL

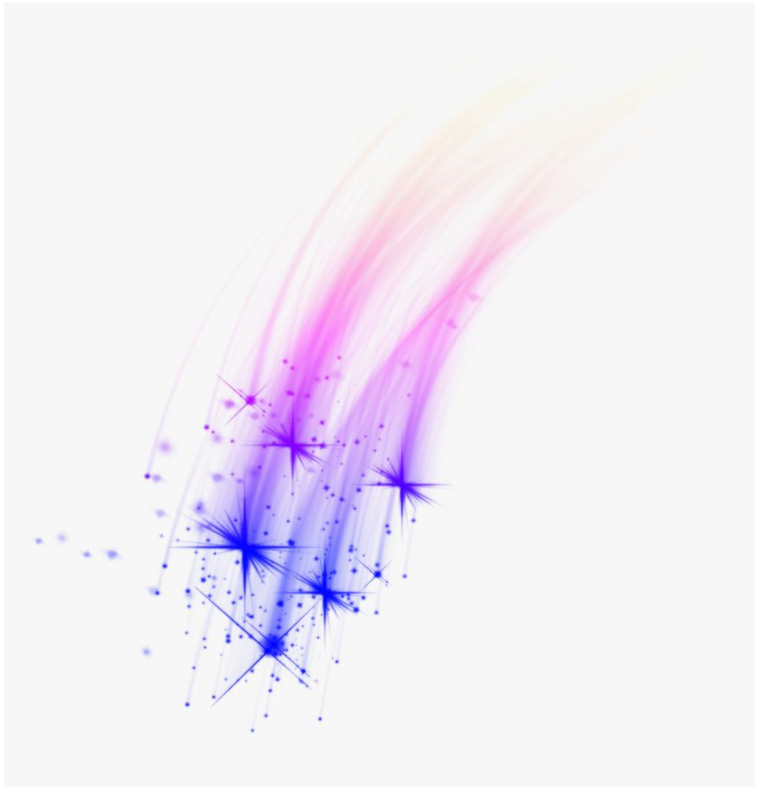
Let’s see how we can override some of the functionality of the Python’s built-in list

Students browse: repl.it/@MakeSchool/stringonlylist?lite=true

Pear Deck Interactive Slide
Do not remove this bar

Now it’s your turn

Try inheriting from one of Python’s built-in classes and give it some custom functionality!



Overloading

- Overriding is when we want to replace the functionality of something with the same name
- What do you think “Overloading” means in OOP?

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

Overloading

- Overloading refers to the ability to have functions with the same name but different functionality based on the parameters
- The idea is that Python would look at the arguments being passed and then decide what implementation to use
- Unlike other languages Python doesn’t directly support true overloading
- However there are *several* ways to implement something in Python that works very similarly to overloading at the usage level!

MAKE
SCHOOL

When might overloading be useful?

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

“Overloading” Techniques

- Optional arguments and conditionals
- @overload decorator from overloading package
- Multiple dispatch

Let’s look at an example!

Let’s try to overload an process() function to behave differently when it gets different kinds of arguments

MAKE SCHOOL

Shout Outs

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar