

Introduction to Inheritance and Object Oriented Programming

Example Instruction Demo

Tyler Westland

Outline

1 High Level

2 Examples

- The Circle
- The Rectangle
- The Generic Shape

What Problems Do Objects Solve

- Programs consistent of numbers. . .
 - Simple: numbers and letters
 - Complex: measurements associated with timestamps

What Problems Do Objects Solve

- Programs consistent of numbers. . .
 - Simple: numbers and letters
 - Complex: measurements associated with timestamps
- and functions
 - Simple: addition and concatenation
 - Complex: sampling measurements to have timestamps in 5 second intervals

What Problems Do Objects Solve

- Programs consistent of numbers. . .
 - Simple: numbers and letters
 - Complex: measurements associated with timestamps
- and functions
 - Simple: addition and concatenation
 - Complex: sampling measurements to have timestamps in 5 second intervals
- Objects combine data with functions useful for that data
 - Simple:

```
1 >>> "cat" + "dog"
2 'catdog'
```

- Complex:

```
1 >>> measurements.resample("5s").bfill()
2 0      1.0
3 5      4.0
4 10     2.0
5 dtype: float64
```

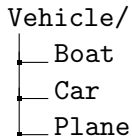
What Problems Does Inheritance Solve

- Allows for reuse of data structures and functions
 - Reused portions can be modified

What Problems Does Inheritance Solve

- Allows for reuse of data structures and functions
 - Reused portions can be modified
- Describes what to expect from every type of an object
 - Every number should allow math operations
 - Every "Vehicle" in a video game should allow "driving"

Example of a Vehicle being the parent to various drivable objects



The Circle Class

code/shapes/circle.py

```
1 from shapes.shape import Shape
2 import math
3
4 class Circle(Shape):
5     def __init__(self, radius: float):
6         self.radius = radius
7
8     def area(self):
9         return math.pi * math.pow(self.radius, 2)
```


Our First Circle

code/main.py

```
1 circ = Circle(3)
2 print(f"circ area: {circ.area():.2f}")
```

output

```
1 circ area: 28.27
2 circ2 area: 78.54
```

Code Snippets

main.py and the output will be shown in snippets. The full text is in the appendix.

Two Circles

code/main.py

```
1 circ = Circle(3)
2 print(f"circ area: {circ.area():.2f}")
3
4 circ2 = Circle(5)
5 print(f"circ2 area: {circ2.area():.2f}")
6
7 print(f"circ.area() > circ2.area(): {circ.area() >
    circ2.area()}")
```

output

```
1 circ area: 28.27
2 circ2 area: 78.54
3 circ.area() > circ2.area(): False
4 rect area: 20
```

The Rectangle Class

code/shapes/rectangle.py

```
1 from shapes.shape import Shape
2
3 class Rectangle(Shape):
4     def __init__(self, length: float, width: float):
5         self.length = length
6         self.width = width
7
8     def area(self):
9         return self.length * self.width
```

Comparing Our Shapes

code/main.py

```
circ = Circle(3)
print(f"circ area: {circ.area():.2f}")

rect = Rectangle(4, 5)
print(f"rect area: {rect.area()}")

print(f"circ.area() > rect.area(): {circ.area() > rect
    .area()}")
```

output

```
circ area: 28.27
rect area: 20
circ.area() > rect.area(): True
```

The Shape Class

code/shapes/shape.py

```
1 from abc import ABC, abstractmethod
2
3 class Shape(ABC):
4     @abstractmethod
5     def area(self):
6         pass
7
8     def __gt__(self, other) -> bool:
9         if isinstance(other, Shape):
10             return self.area() > other.area()
11         else:
12             raise ValueError("Can only compare with
    shapes")
```

Comparing Our Shapes, With Style

code/main.py

```
circ = Circle(3)
print(f"circ area: {circ.area():.2f}")

circ2 = Circle(5)
print(f"circ2 area: {circ2.area():.2f}")

rect = Rectangle(4, 5)
print(f"rect area: {rect.area()}")

print(f"circ > circ2: {circ > circ2}")
print(f"circ > rect: {circ > rect}")
```

output

```
circ area: 28.27
circ2 area: 78.54

rect area: 20

circ > circ2: False
circ > rect: True
```

Folder Structure

```
code/  
├── main.py  
├── shapes/  
│   ├── shape.py  
│   ├── circle.py  
│   └── rectangle.py
```

main.py

```
1 #!/bin/env python3
2 from shapes.rectangle import Rectangle
3 from shapes.circle import Circle
4
5
6 circ = Circle(3)
7 print(f"circ area: {circ.area():.2f}")
8
9 circ2 = Circle(5)
10 print(f"circ2 area: {circ2.area():.2f}")
11
12 print(f"circ.area() > circ2.area(): {circ.area() >
    circ2.area()}")
13
14 rect = Rectangle(4, 5)
15 print(f"rect area: {rect.area()}")
```


main.py – cont

```
1
2 print(f"circ.area() > rect.area(): {circ.area() > rect
   .area()}")
3
4
5 print(f"circ > circ2: {circ > circ2}")
6 print(f"circ > rect: {circ > rect}")
```

shapes/shape.py

```
1 from abc import ABC, abstractmethod
2
3 class Shape(ABC):
4     @abstractmethod
5     def area(self):
6         pass
7
8     def __gt__(self, other) -> bool:
9         if isinstance(other, Shape):
10             return self.area() > other.area()
11         else:
12             raise ValueError("Can only compare with shapes")
```

shapes/circle.py

```
1 from shapes.shape import Shape
2 import math
3
4 class Circle(Shape):
5     def __init__(self, radius: float):
6         self.radius = radius
7
8     def area(self):
9         return math.pi * math.pow(self.radius, 2)
```

shapes/rectangle.py

```
1 from shapes.shape import Shape
2
3 class Rectangle(Shape):
4     def __init__(self, length: float, width: float):
5         self.length = length
6         self.width = width
7
8     def area(self):
9         return self.length * self.width
```

output

```
1 circ area: 28.27
2 circ2 area: 78.54
3 circ.area() > circ2.area(): False
4 rect area: 20
5 circ.area() > rect.area(): True
6 circ > circ2: False
7 circ > rect: True
```