Introduction to Object Oriented Programming (OOP)

Tashfeen, Ahmad

Computer Science, Oklahoma City Community College

Interview Presentation, Fall 2022

Overview

Introduction: Object, Oriented & Programming

```c
#include <stdio.h>
int main() {
  printf("Hello World!");
  return 0;
}
```

Listing 1: C program to print "Hello World!"

Programming Setting up instructions for a computer.

Oriented Just a verb.

Object What is an object? E. g.,

Figure: A chair.

Introduction: Objects' Features Overlap

Legs Humans have two,
Chairs have four.

Geo. Objects Have area.

Circle πr^2

Square x^2

Figure: Geometrical Objects

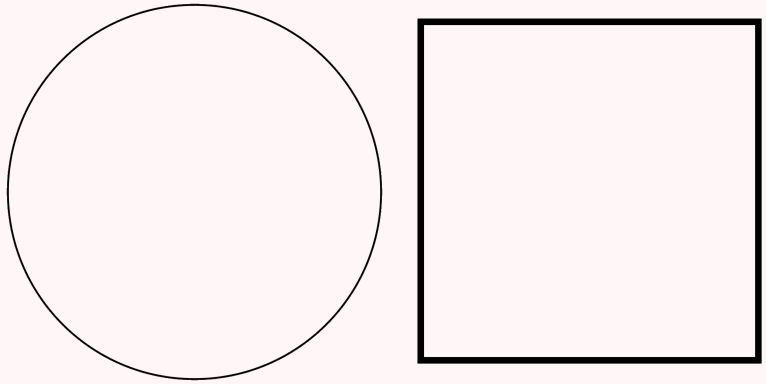

Figure: Geometrical Objects

Code: Not OOP

```
lengths = [1, 2, 3]
def area_circle(r):
    return 3.1415*r*r
def area_square(x):
    return x*x
for length in lengths:
    print(f'Circle\'s Area: {area_circle(length)}')
    print(f'Square\'s Area: {area_square(length)}')
```

Listing 2: Functional programming.

```
Circle's Area: 3.1415
Square's Area: 1
Circle's Area: 12.566
Square's Area: 4
Circle's Area: 28.2735
Square's Area: 9
```

Listing 3: Output.

Code: Object Definitions

```
1  from abc import ABC, abstractmethod
2
3  class GeometricalObject(ABC):# defining class of Geo. Object
4      def __init__(self, length):
5          self.length = length # that have a length attribute
6      @abstractmethod
7      def area(self): # and you can implemnet specific areas
8          pass
9  class Circle(GeometricalObject):
10     def area(self):
11         area_value = 3.1415*self.length*self.length
12         print(f'Circle\'s Area: {area_value}')
13 class Square(GeometricalObject):
14     def area(self):
15         area_value = self.length*self.length
16         print(f'Square\'s Area: {area_value}')
```

Listing 4: Object oriented programming.

Code: Objects' Usage

```
1 from lib import Circle, Square
2
3 circles = [Circle(1), Circle(2), Circle(3)]
4 squares = [Square(1), Square(2), Square(3)]
5 for circle, square in zip(circles, squares):
6     circle.area()
7     square.area()
```

Listing 6: Object oriented programming.

1. Same output as before.
2. Caller of `area()` does not need to know anything about how to calculate it.
3. Only need to keep track of one object and not the function and its parameters.
4. Easy to read.

Code: Objects' Usage

```
1 from sklearn.neural_network import MLPClassifier
2
3 net = MLPClassifier(max_iter=300)
4 net.fit([], [])  # no data
```

Listing 7: All you need to build a neural network (Lars Buitinck et al. [1]).

Review

1. What does the word *object* mean? Something with features.
2. Objects are not mutually exclusive.
3. The antonym to OOP: functional programming.
4. Hierarchical object definitions.
5. Using objects (not necessarily written by yourself.)

References

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux.
API design for machine learning software: experiences from the scikit-learn project.
In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

Thank You!
Questions?