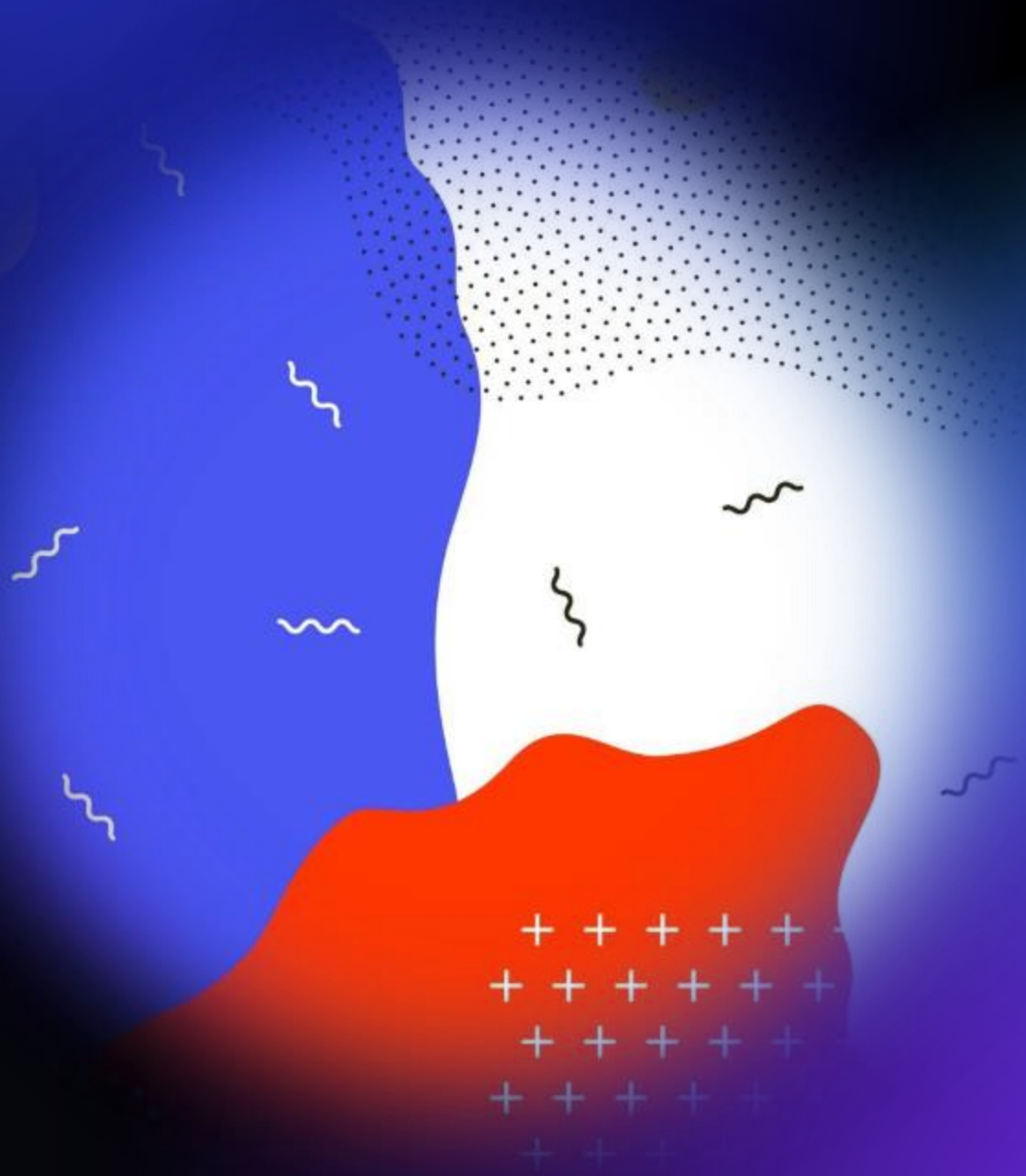
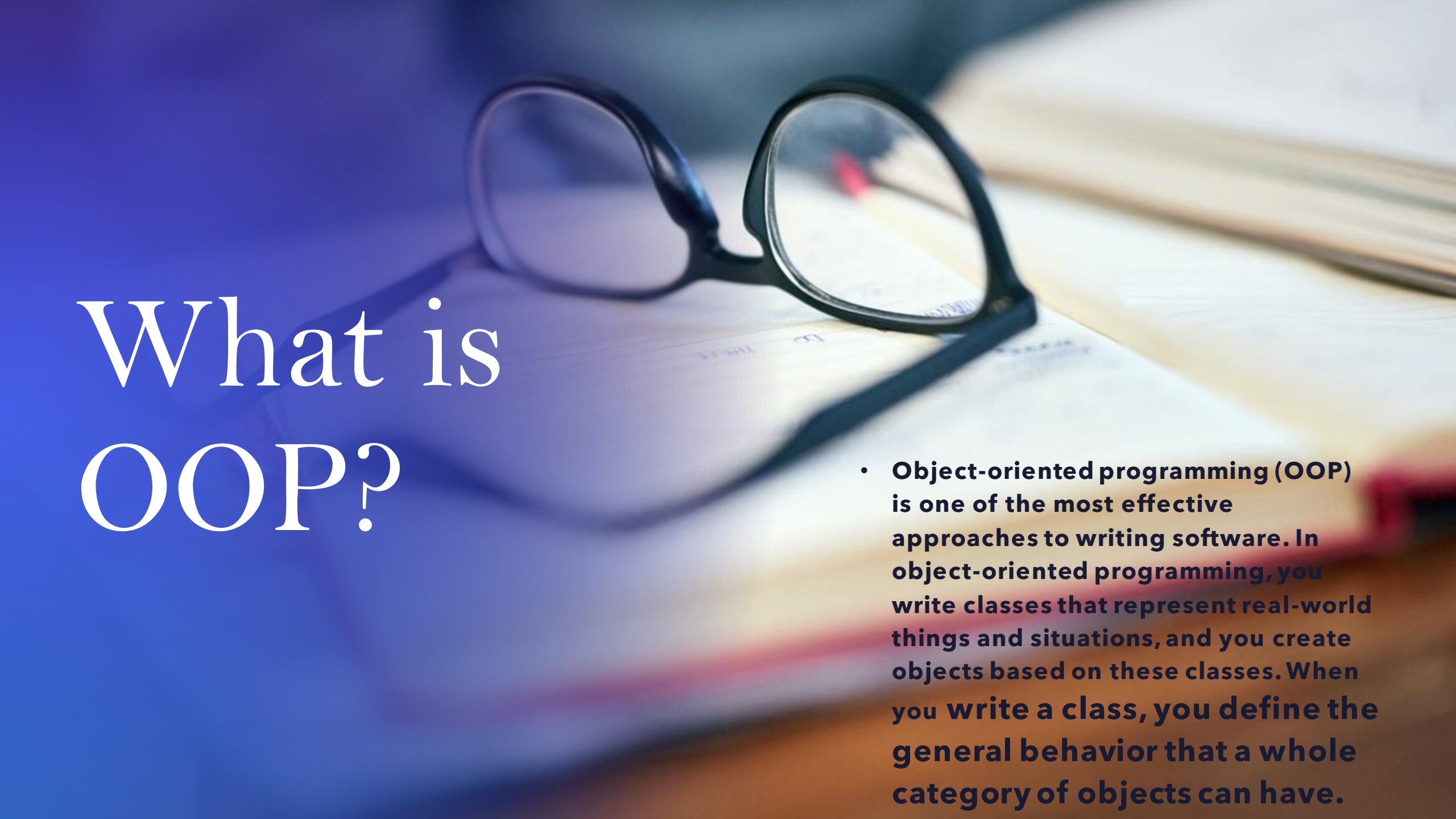
ENG: AHMED HEGAZY

oop

OBJECT-
ORIENTED PROGRAMMING



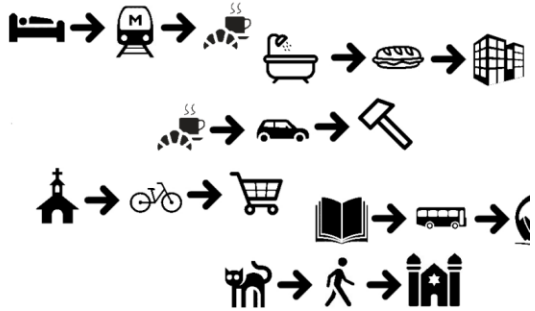
What is OOP?



- **Object-oriented programming (OOP) is one of the most effective approaches to writing software. In object-oriented programming, you write classes that represent real-world things and situations, and you create objects based on these classes. When you write a class, you define the general behavior that a whole category of objects can have.**

How oop actually works:

Thinking in sequences



Procedural programming

- Code as a sequence of steps
- Great for data analysis and scripts

Object-oriented programming

- ***Code as interactions of objects***
- Great for building frameworks and tools
- *Maintainable and reusable code!*

Objects in Python

- *Everything in Python is an object*
- Every object has a class
- Use `type()` to find the class

```
import numpy as np
a = np.array([1,2,3,4])
print(type(a))
```

```
numpy.ndarray
```

Object	Class
`5`	`int`
`"Hello"`	`str`
`pd.DataFrame()`	`DataFrame`
`np.mean`	`function`
…	…

Attributes and methods

State ↔ attributes

```
import numpy as np
a = np.array([1,2,3,4])
# shape attribute
a.shape
```

```
(4,)
```

Behavior ↔ methods

```
import numpy as np
a = np.array([1,2,3,4])
# reshape method
a.reshape(2,2)
```

```
array([[1, 2],
       [3, 4]])
```

- Use `obj.` to access attributes and methods

The __init__() Method

A function that's part of a class is a method. Everything you learned about functions applies to methods as well; the only practical difference for now is the way we'll call methods. The __init__() method 2 is a special method that Python runs automatically whenever we create a new instance based on the Dog class. This method has two leading underscores and two trailing underscores, a convention that helps prevent Python's default method names from conflicting with your method names. Make sure to use two underscores on each side of __init__(). If you use just one on each side, the method won't be called automatically when you use your class, which can result in errors that are difficult to identify.

```
class Customer:

    def identify(self, name):
      print("I am Customer " + name)

cust = Customer()
cust.identify("Laura")
```

What is self?

- classes are templates, how to refer data of a particular object?
- `self` is a stand-in for a particular object used in class definition
- should be the first argument of any method
- Python will take care of `self` when method called from an object:

`cust.identify("Laura")` *will be interpreted as* `Customer.identify(cust, "Laura"`

Enjoy coding

ENG: AHMED HEGAZY