



Python | Main course

# Session 6

Review

OOP intro

OOP in python



Maktab  
Sharif

by Mohammad Amin H.B. Tehrani

[www.maktabsharif.ir](http://www.maktabsharif.ir)

# Review





# Contents

- 1) Variables
- 2) Types
- 3) Operators
- 4) String
- 5) Data structures (List, Dict, Tuple, Set )
- 6) Conditional Statements ( if .. else .. )
- 7) Loops (While, For)
- 8) Function
- 9) Built-in functions: map, sorted, filter, ...
- 10) List comprehension (inline for), Ternary expression (inline if), lambda

# Object-Oriented Programming Introduction





# OOP philosophy

**Object-oriented programming (OOP)** is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods)

**You should divide your program not into tasks,  
but into models of physical objects.**

Procedural programming

What does this program do?

vs.

**Object-oriented programming**

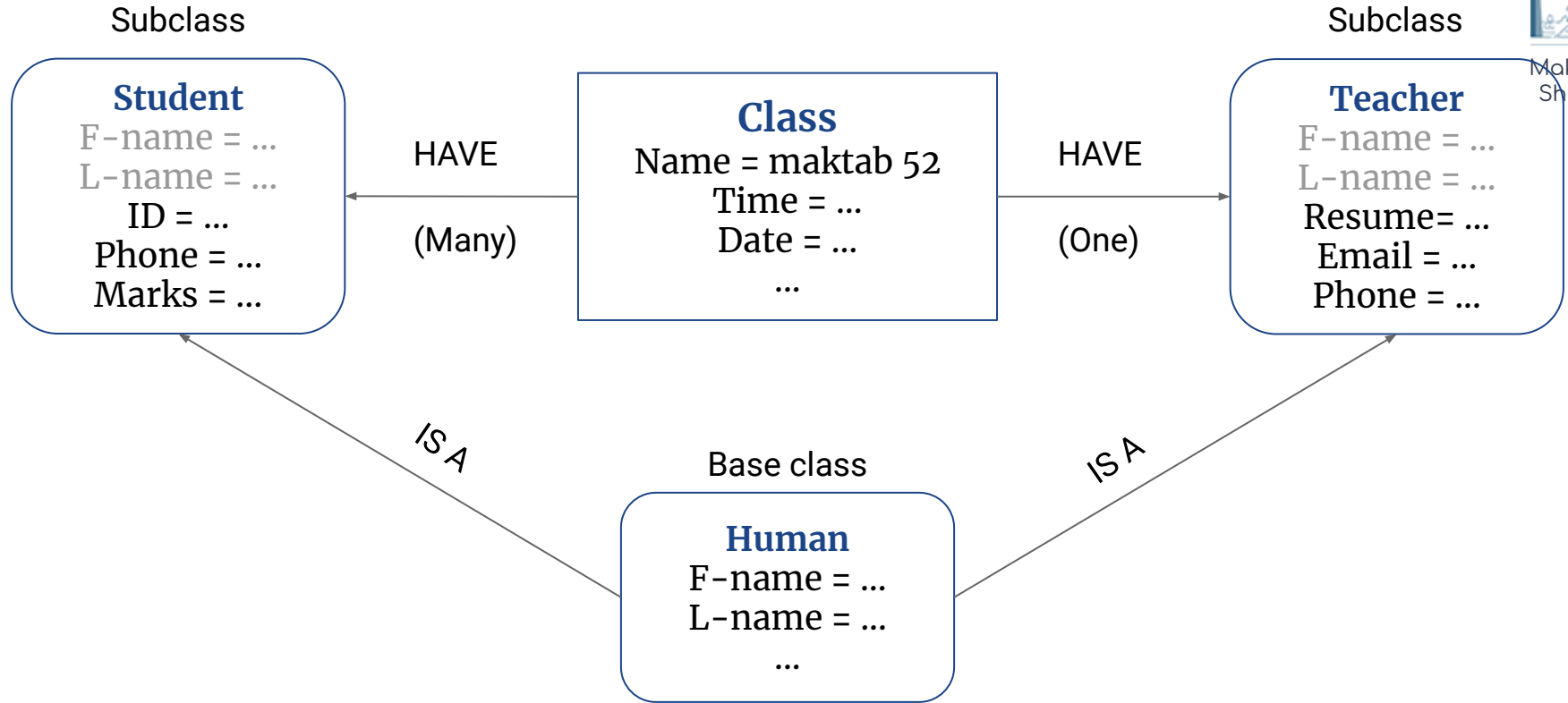
What real world objects am  
I modeling?



# Object-Oriented Fundamentals

Principles:

- 1) Class
- 2) Object
- 3) Hierarchy
- 4) Encapsulation
- 5) Abstraction
- 6) Inheritance
- 7) Polymorphism



# Class



A user-defined prototype for an object that defines a set of attributes that characterize any object of the class.

→ Create a class, which is like a **blueprint** for creating an object

Syntax

```
class ClassName:
```

```
    ...
```

```
class Square:
    x = 10
    y = 20
    ...
```

```
class Student:
    name = 'Akbar'
    marks = []
    ...
```





# Instantiate an Object in Python

**Instance:** An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle

Creating a new object from a class is called instantiating an object. You can instantiate a new object by typing the name of the class, followed by opening and closing parentheses:

Syntax

**ins = ClassName(...)**

```
class Square:
    x = 10
    y = 20
    ...
```

```
s = Square()
```

```
class Student:
    name = 'Akbar'
    marks = []
    ...
```

```
S = Student()
```



# Instance/Object Attributes (fields)

An instance/object attribute is a variable that belongs to one (and only one) object. Every instance of a class points to its own attributes variables.

```
class Human:
    first_name = ...
    last_name = ...
    age: int
    gender: str
    height: int
    ...
```

```
class Car:
    brand: str

    def __init__(self):
        self.model = ...
        self.color = ...
        self.fuel = ...
```



# Instance/Object Methods

Methods are functions defined inside the body of a class. They are used to define the behaviors of an object.

A method is a function that “belongs to” an object.

```
class Human:
    name = ...

    def sleep(self, time):
        ...

    def eat(self, food):
        ...
```

```
class Car:
    speed = ...

    def start(self):
        ...

    def brake(self):
        ...
```



# Initialize object (Constructor)

## Method: `__init__(self, ...)`

`__init__` is one of the reserved methods in Python. In object oriented programming, it is known as a constructor. The `__init__` method can be called when an object is created from the class, and access is required to initialize the attributes of the class.

```
class Human:

    def __init__(self, first_name, last_name, **extra_information):
        self.name = first_name + last_name
        self.extra_info = extra_information

akbar = Human('Akbar', 'Rezaii', age=25, height=168)
```



# Example

```
class Square:

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def area(self):
        return self.x * self.y

s = Square(2, 5)
print(s.area())
```

# Example



```
class Square:

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def area(self):
        return self.x * self.y

s = Square(2, 5)
print(s.area())
```