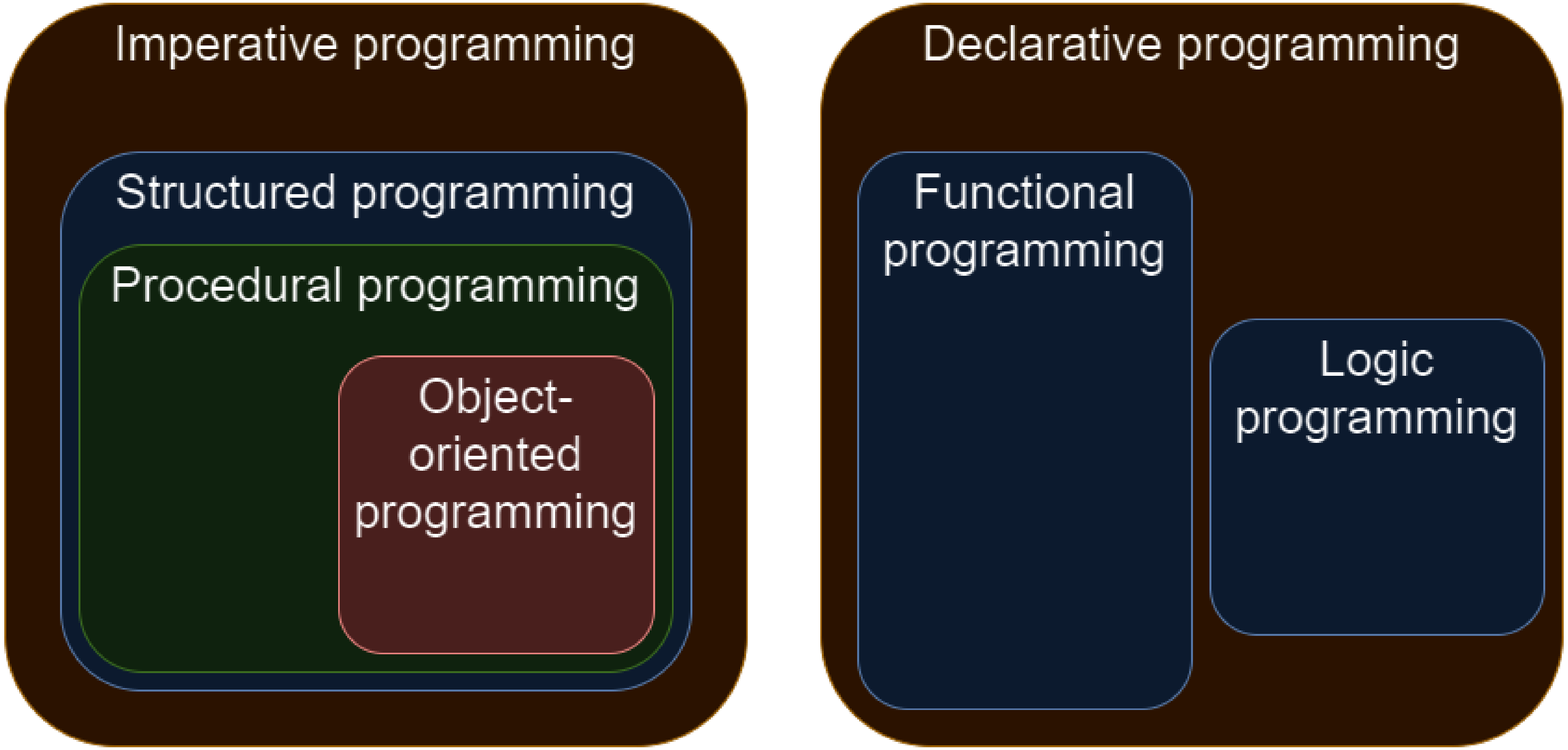
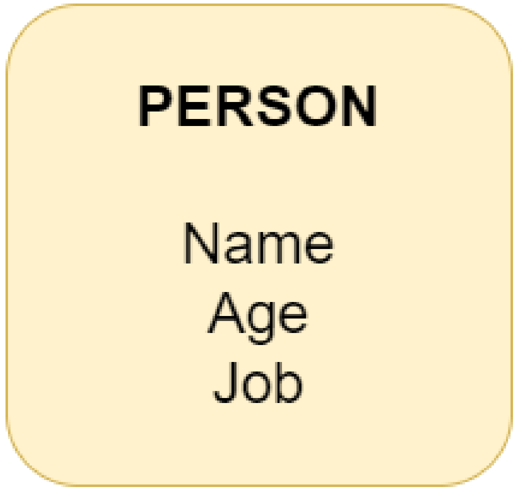
Programming for Social Scientists

Johan A. Dornschneider-Elkink

Object-oriented programming



Programming paradigms

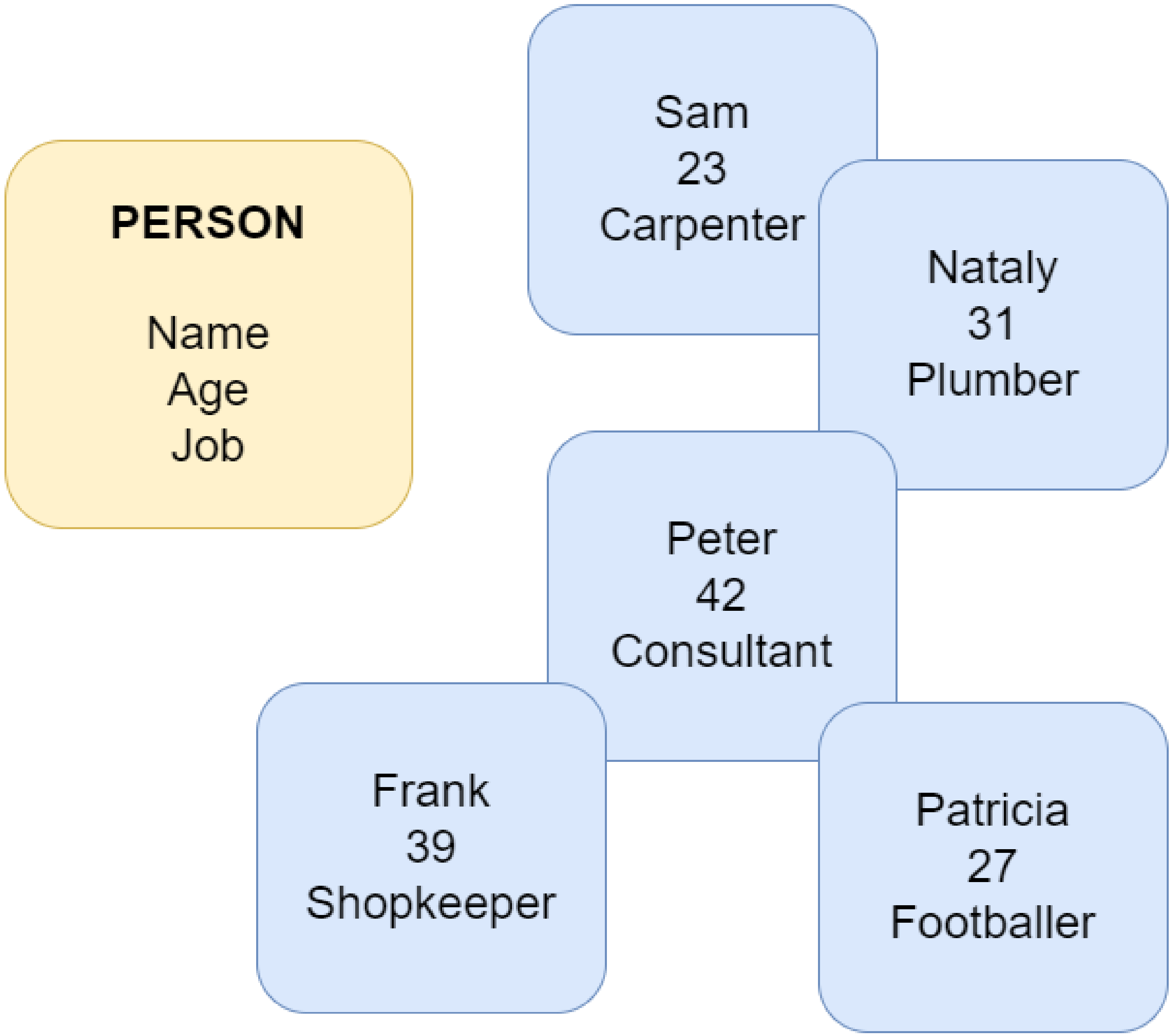


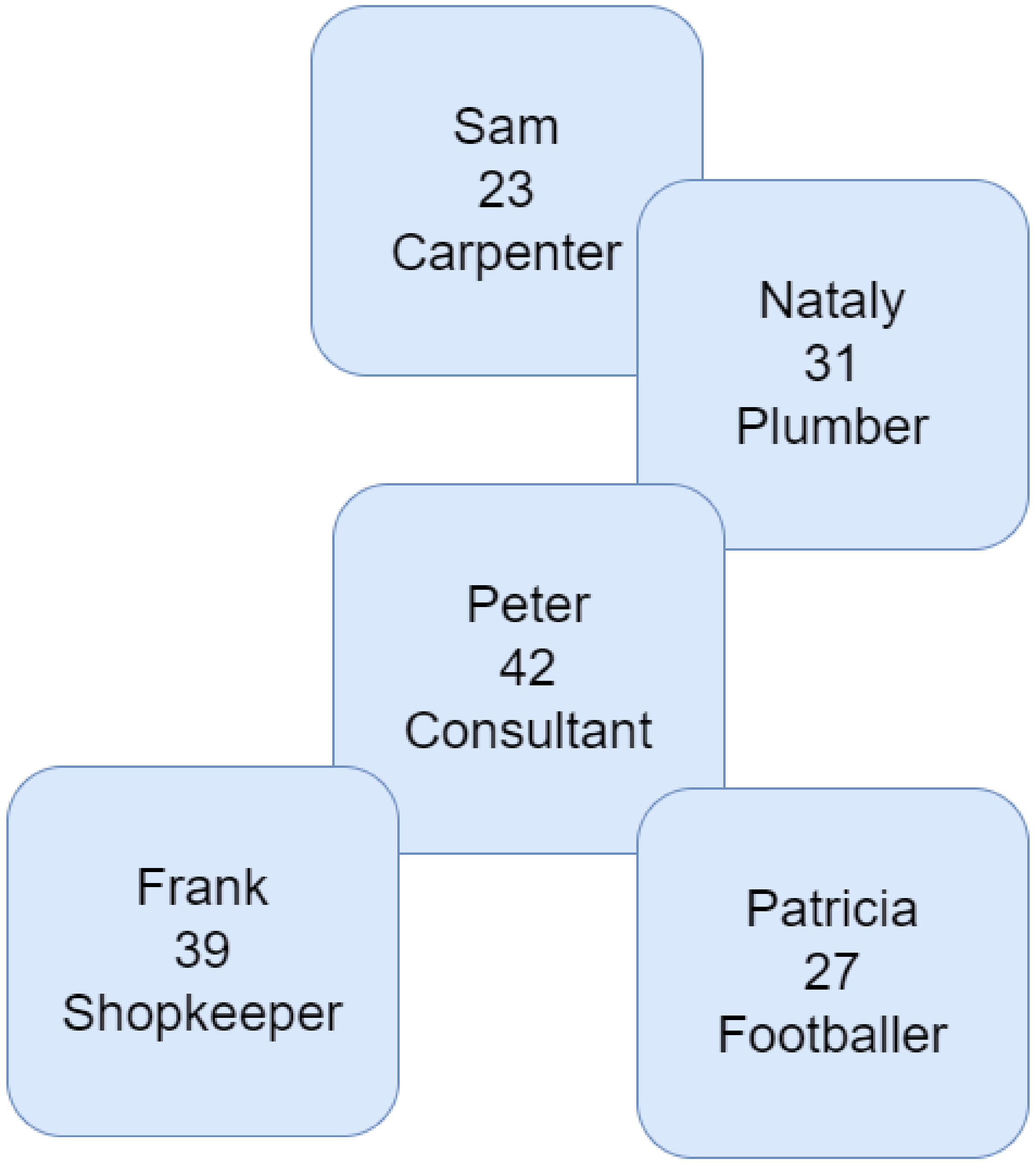
Class

Blueprint or template for user-defined data.

Defines data and functionality to be associated with each instance.

Does not yet reserve any memory space for data.

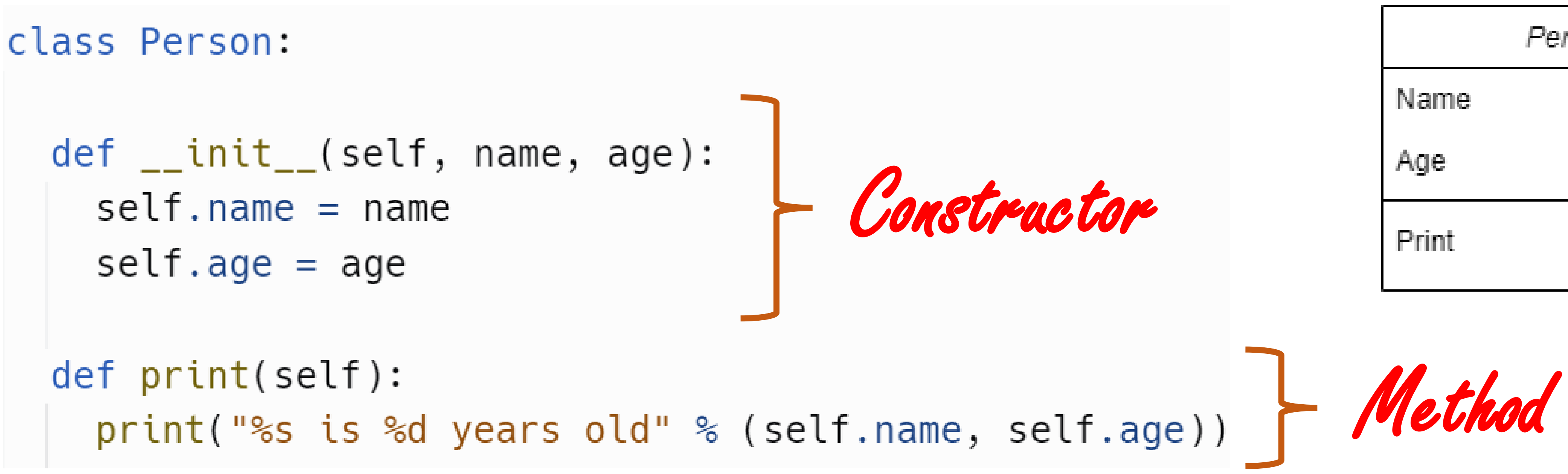

PERSON
Name
Age
Job
Sam
23
Carpenter
Nataly
31
Plumber
Peter
42
Consultant
Frank
39
Shopkeeper
Patricia
27
Footballer



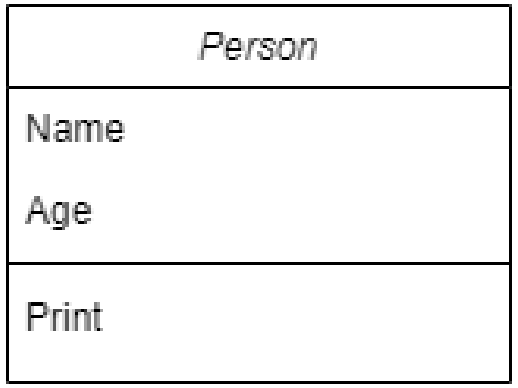
Object

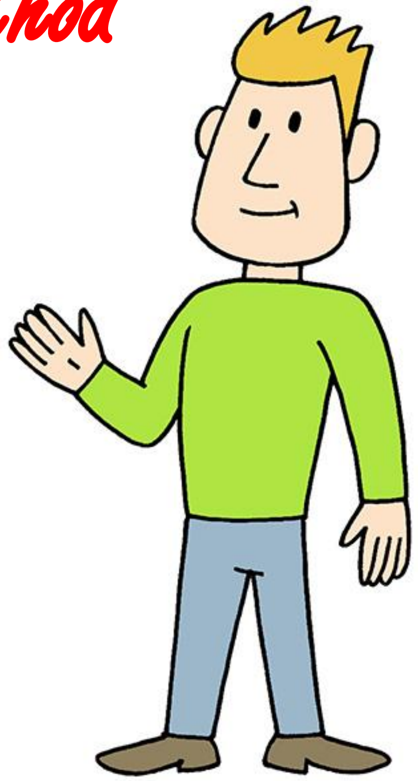
Instance of a specific object, based on the class definition.

Reserves specific memory space for the data, as any other variable type.



```python
class Person:

  def __init__(self, name, age):
    self.name = name
    self.age = age

  def print(self):
    print("%s is %d years old" % (self.name, self.age))
```





```
class Person:

  def __init__(self, name, age):
    self.name = name
    self.age = age

  def print(self):
    print("%s is %d years old" % (self.name, self.age))
```

Defining a class

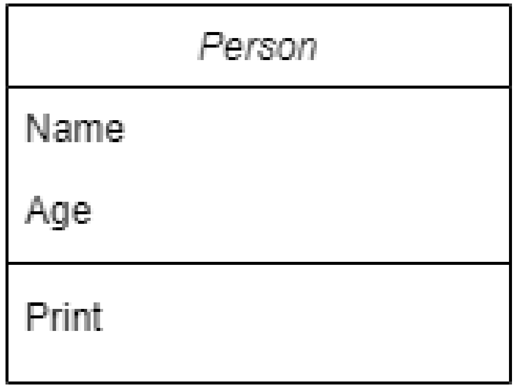
```
john = Person("John", 42)
peter = Person("Peter", 30)
```

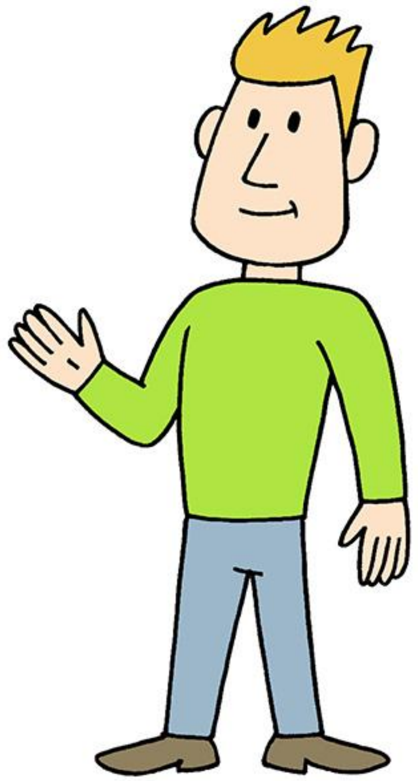
Creating objects

```
john.print()

peter.print()
```

Calling methods

```
print(type(john))
```





```
class Person:

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def print(self):
        print("%s is %d years old" % (self.name, self.age))
```

$\}$ *Instance variables*

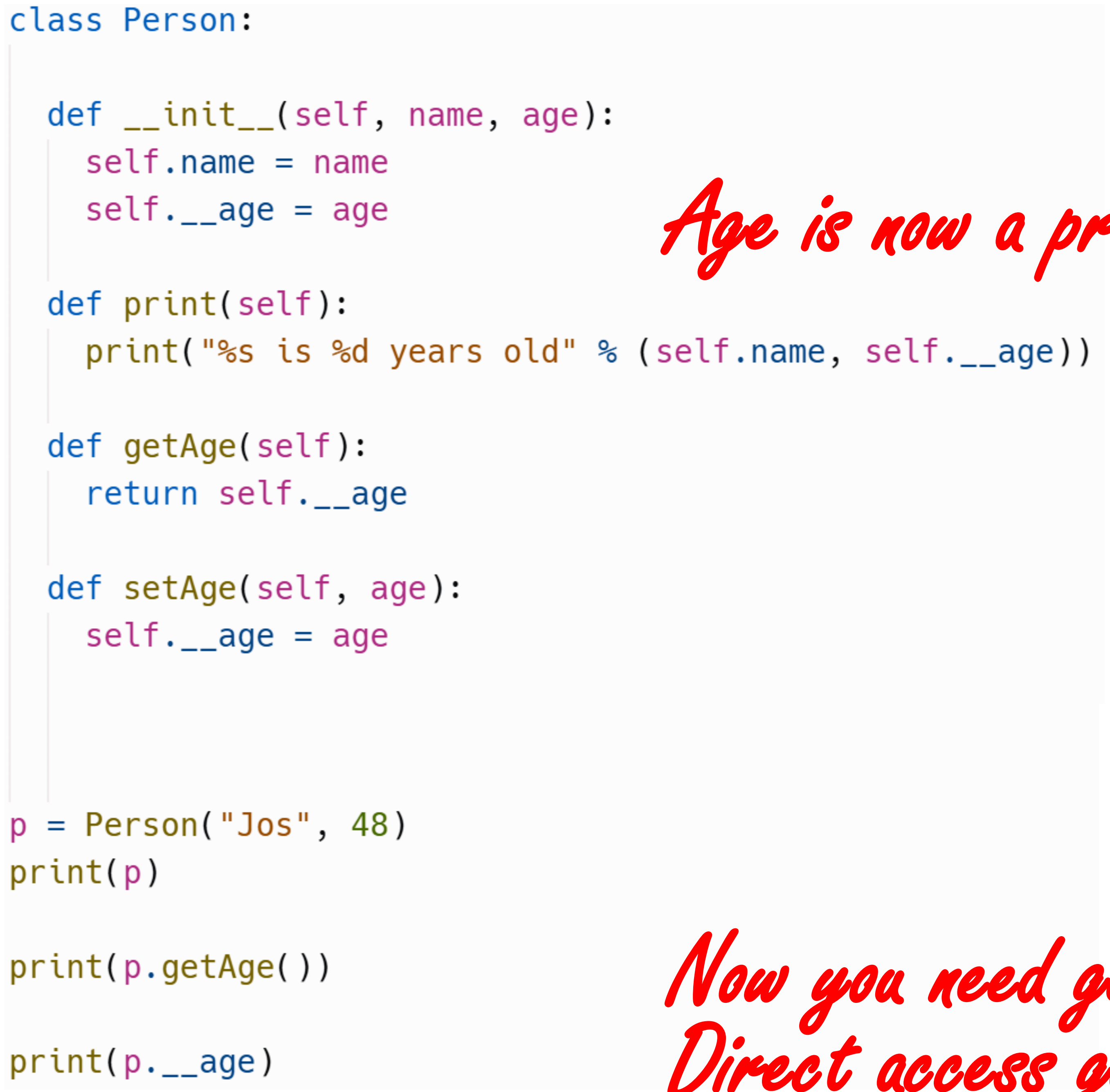
```
p = Person("Jos", 48)
print(p)

print(p.age)
```

Instance variables are publicly accessible

```python
class Person:

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def print(self):
        print("%s is %d years old" % (self.name, self.age))

    def getAge(self):
        return self.age

    def setAge(self, age):
        self.age = age
```

Getter- and setter-methods

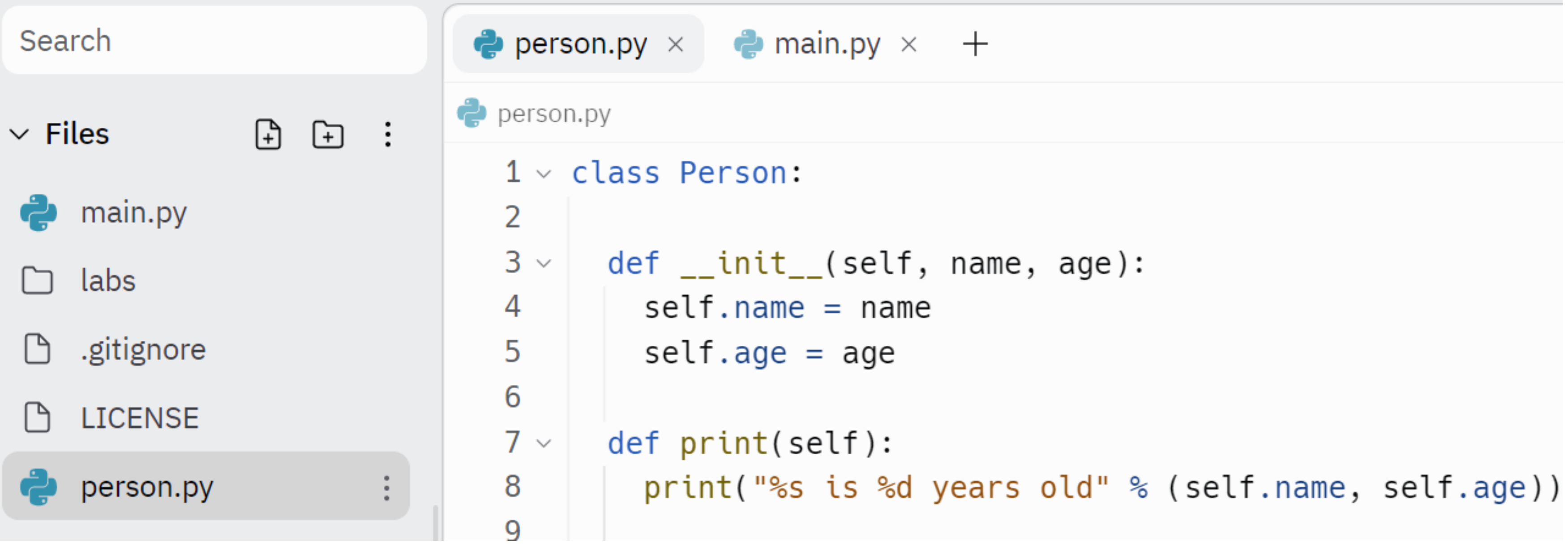
```python
p = Person("Jos", 48)
print(p)

print(p.getAge())
```



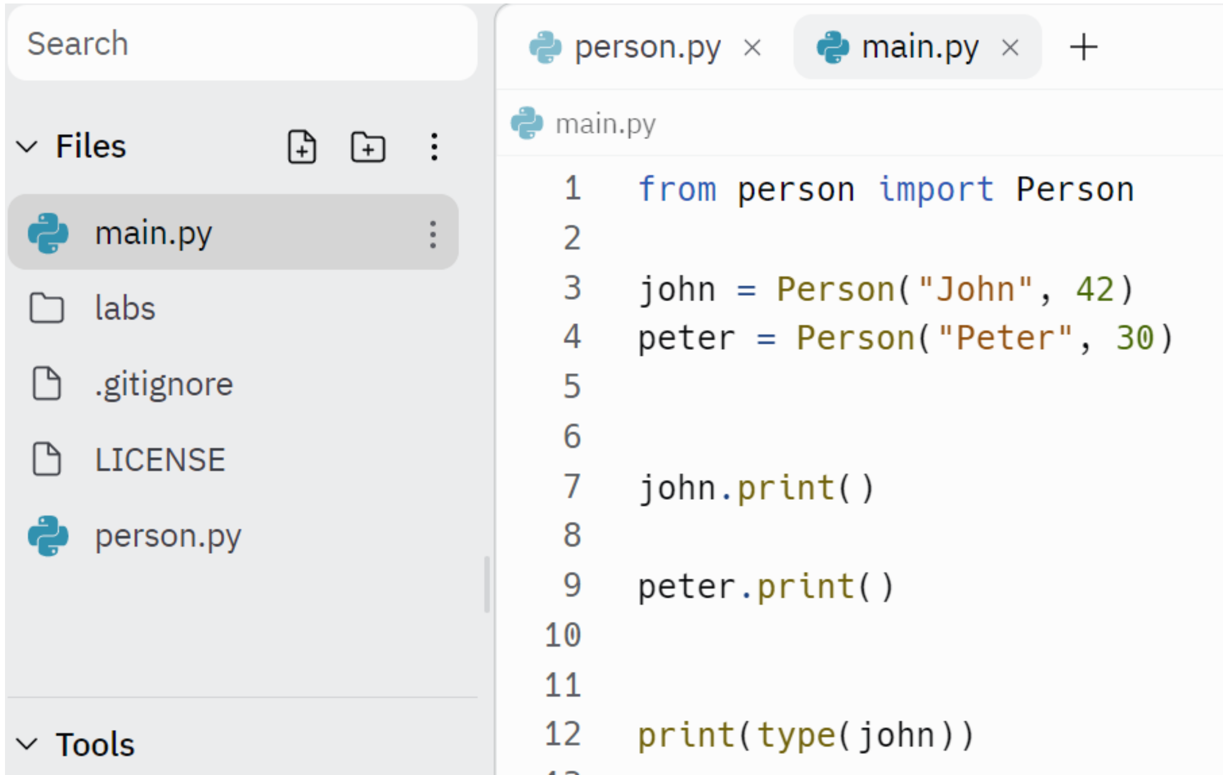
```
class Person:

  def __init__(self, name, age):
    self.name = name
    self.__age = age

  def print(self):
    print("%s is %d years old" % (self.name, self.__age))

  def getAge(self):
    return self.__age

  def setAge(self, age):
    self.__age = age



p = Person("Jos", 48)
print(p)

print(p.getAge())

print(p.__age)
```

Age is now a private instance variable

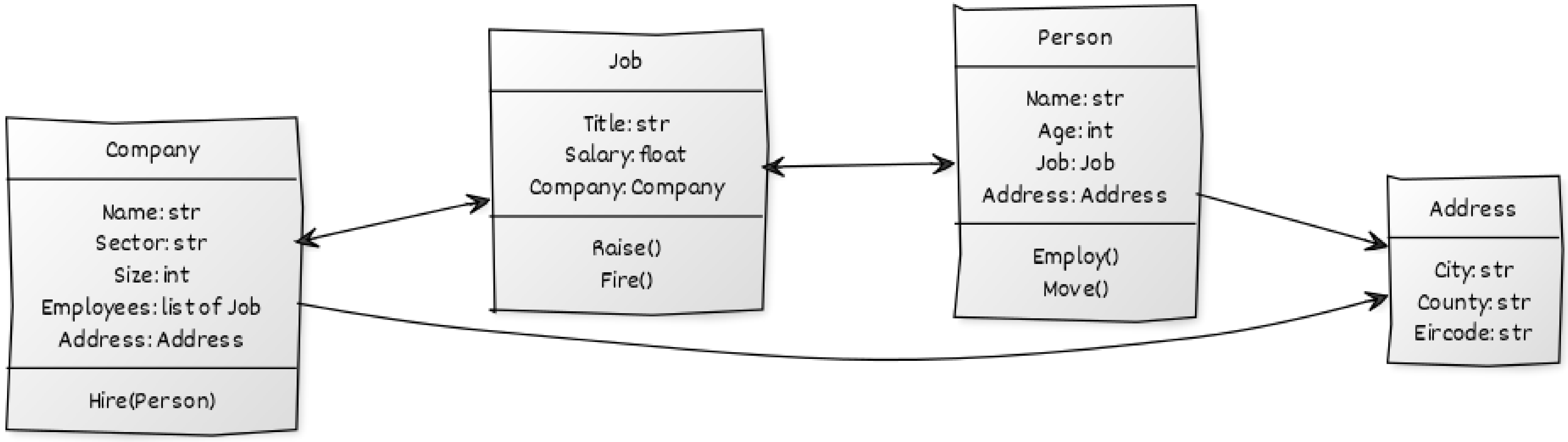
Now you need getter- and setter-methods
Direct access generates error



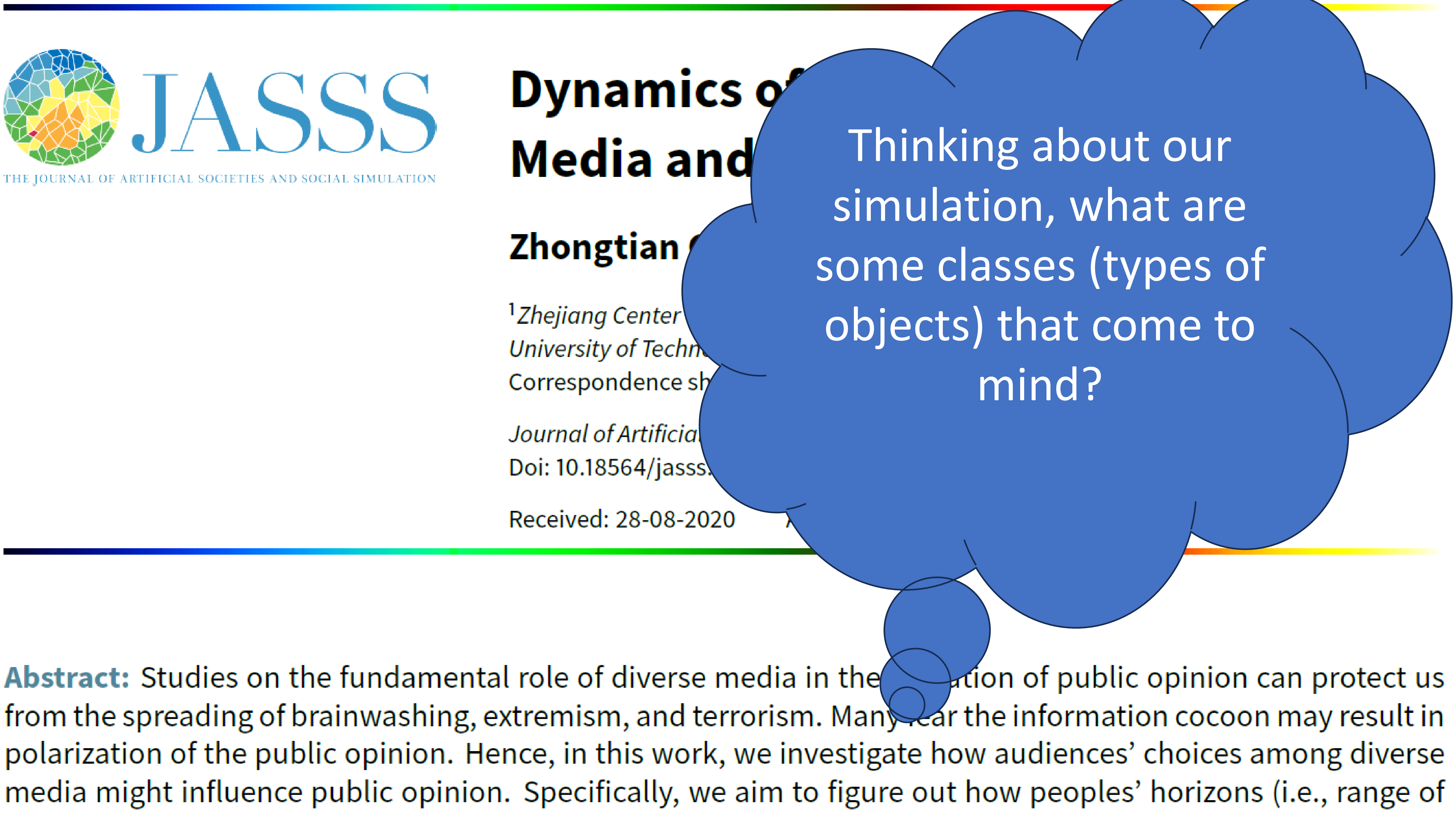
```python
class Person:

  def __init__(self, name, age):
    self.name = name
    self.age = age

  def print(self):
    print("%s is %d years old" % (self.name, self.age))

```



```python
from person import Person

john = Person("John", 42)
peter = Person("Peter", 30)


john.print()

peter.print()



print(type(john))
```



CREATED WITH YUML



JASSS

THE JOURNAL OF ARTIFICIAL SOCIETIES AND SOCIAL SIMULATION

Dynamics o
Media and

Zhongtian

¹*Zhejiang Center*
University of Techn
Correspondence sh

Journal of Artificia
Doi: 10.18564/jasss.

Received: 28-08-2020

Thinking about our simulation, what are some classes (types of objects) that come to mind?

Abstract: Studies on the fundamental role of diverse media in the tion of public opinion can protect us from the spreading of brainwashing, extremism, and terrorism. Many ear the information cocoon may result in polarization of the public opinion. Hence, in this work, we investigate how audiences' choices among diverse media might influence public opinion. Specifically, we aim to figure out how peoples' horizons (i.e., range of