

# Hello, my name is <name>

## An introduction to OOP in Python with UML

Kristoffer Nielbo

**Center for Humanities Computing Aarhus**|chcaa.io  
aarhus university, denmark



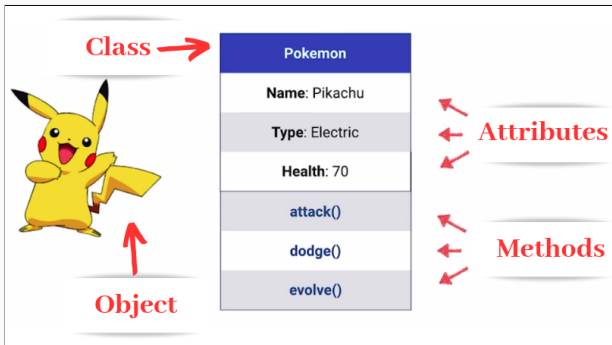
CENTER FOR HUMANITIES  
COMPUTING AARHUS



# IMPERATIVE

```
1 class Person:
2     def __init__(self, name, age=None, sex=None):
3         self.name = name
4         self.age = age
5         self.sex = sex
6
7     def says(self, message = '...'):
8         print(f'{self.name}: {message}')
9
10 class Researcher(Person):
11     def __init__(self, pay=10, areas=['research'], **kwargs):
12         super(Researcher, self).__init__(**kwargs)
13         self.areas = areas
14
15 if __name__ == '__main__':
16     kln = Researcher(
17         name='Kristoffer L. Nielbo',
18         age=44, sex='male',
19         areas=['humanities computing', 'interactive HPC']
20     )
21
22     kln.says(message = 'hello and welcome to PftH21!')
```

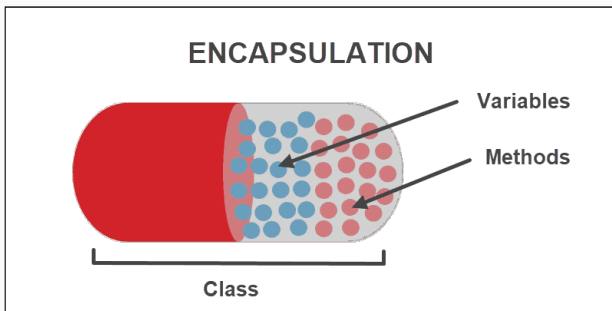
# OBJECT



A collection of data and associated behaviors, source: W. Robson

- ▶ an object is an **instance of a class** that defines a set of attributes and behaviors shared by all objects of that class
- ▶ a **context-specific model** of an type of entity in some system

# ENCAPSULATION



Bundling of data and methods operating on said data into one unit. Encapsulation provides the basic property to hide data, thereby providing security to user data.

# TASK-SPECIFIC ABSTRACTION #1

Software
+instructions +libraries +data
+load() +execute()

- the level of detail of the abstraction, is specific to the task

## OBJECT ANALYSIS

“Hello, my name is prof. Kristoffer Nielbo, I work at Aarhus University-DK with *humanities computing* and *culture analytics*. I currently have two primary research projects: *FabulaNet* and *News Information Decoupling*”

Implicit knowledge

- ▶ ‘Kristoffer Nielbo’ → (male & age)
- ▶ ‘at Aarhus University’ → (researcher → pay grade)
- ▶ ‘primary research projects’ → principal investigator
- ▶ principal investigator *is a* researcher *is a* person
- ▶ ...

# OBJECTS AND DATA

- object or instance diagram of a **Person**:

<u><b>Kristoffer: Person</b></u>
name = Kristoffer L. Nielbo
age = 44
sex = Male

- data represents **attributes** of an object

## OBJECT AND INHERITANCE

- the instance is also a `Researcher`, and a `Researcher` is a `Person`

### Kristoffer: Researcher

```
name = Kristoffer L. Nielbo  
age = 44  
sex = Male  
pay = 10  
areas = humanities computing, culture analytics
```

- `Researcher` is a subclass of `Person` and `inherits` attributes



## OBJECTS AND BEHAVIOR

- A PI is a Researcher is a Person, but a PI has additional 'Pain and Suffering'

<u>Kristoffer: Principal Investigator</u>
name = Kristoffer L. Nielbo
age = 44
sex = Male
pay = 10
areas = humanities computing, culture analytics
+pain & suffering: 10%

- an object is a collection of data and behaviors
- an object is a hierarchical composite that inherits data and behaviors

# CLASS DESIGN

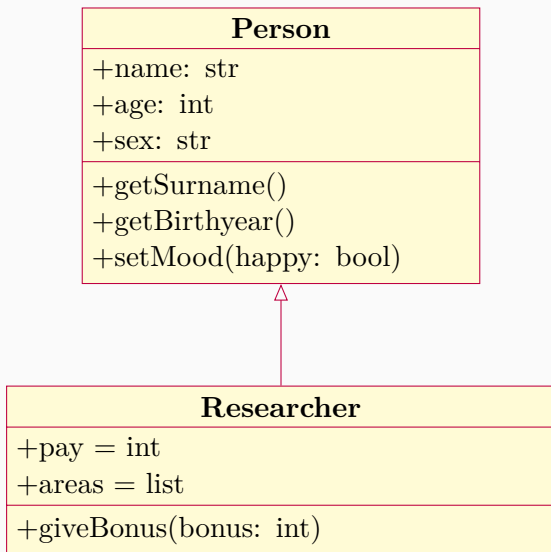
```
print(  
    "Hello, my name is <name>,  
    I work at Aarhus University-DK with <area-1> and <area-2>.  
    I currently have <count(projects)> primary research projects:  
    <project-1> and <project-2>"  
)
```

# PERSON

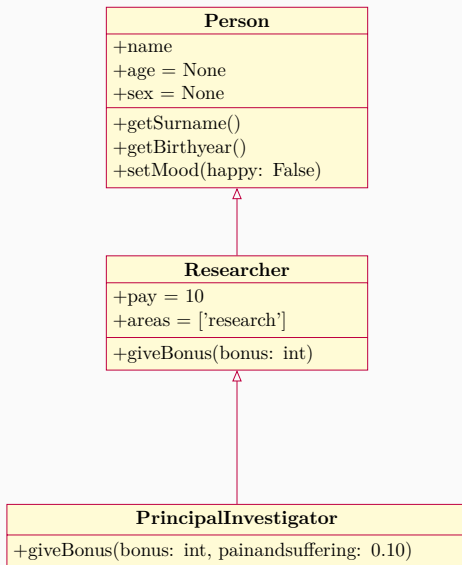
Person
+name: str +age: int +sex: str
+getSurname() +getBirthyear() +setMood()

- a class defines a set of attributes shared by all objects of that class

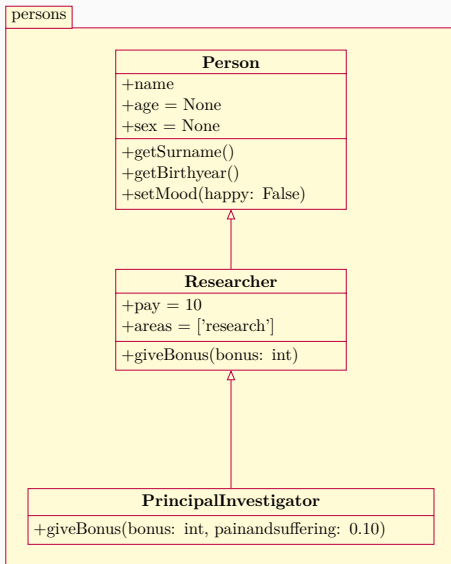
# INHERITANCE



# MULTIPLE INHERITANCE



# MODULE



# COMPOSITION

persons

