# OBJECT-ORIENTED-PROGRAMMING (OOP) IN PYTHON

ELEG4701 – IIRP

Term 2, 2023-24

WANG An
EE, CUHK

# MODELLING

The data

```
ages = [43, 47, 10, 7, 3]
```

Computations

```
def average(numbers):
    return sum(numbers)/len(numbers)
```

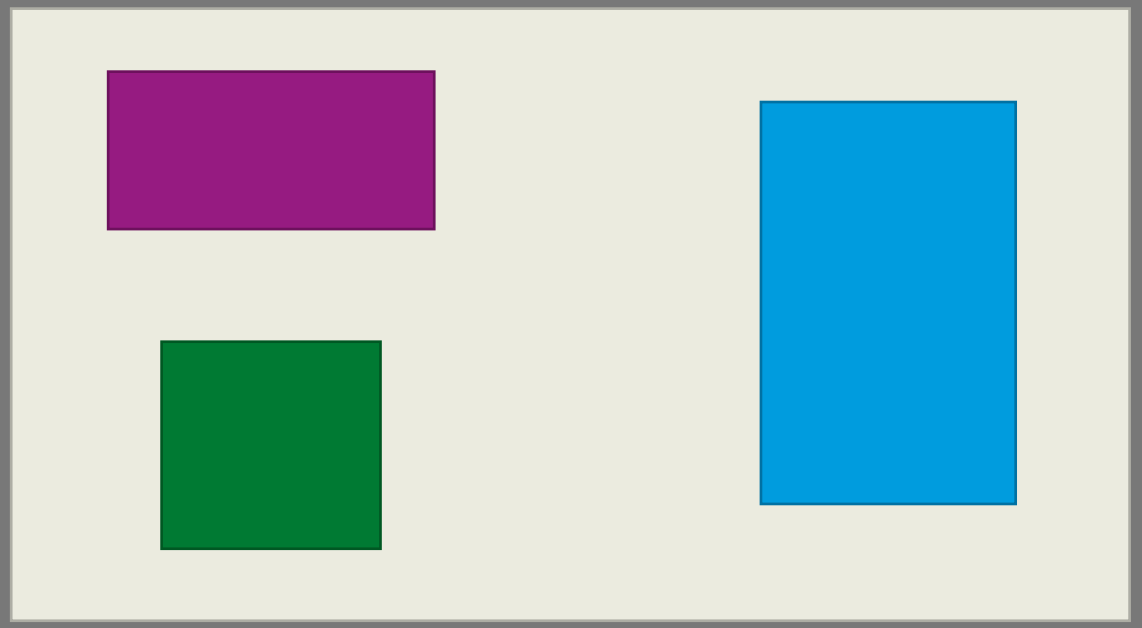
User Interface

```
print("Average age: "+str(average(ages)))
```

Object-Oriented Programming keeps **data** and **functions** together.

# CLASSES AND OBJECTS

A class is a description of/template for *something*.



Screen.

Use a class to describe a rectangle:

- Width & height
- Position
- Color

Create 3 instances to represent your actual rectangles.

Objects are instances of classes.

# YOU HAVE ALREADY USED OOP

All values in Python are objects.

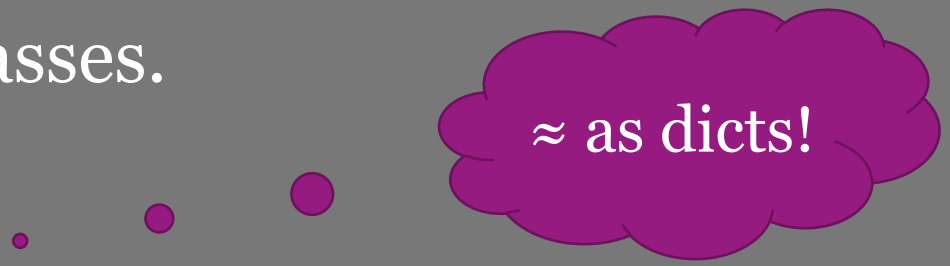
```
>>> a = 123
>>> type(a)
<class 'int'>
>>> b = "abc"
>>> type(b)
<class 'str'>
>>> c = [1, 2]
>>> type(c)
<class 'list'>
```

# CLASSES AND OBJECTS

Objects are instances of classes.

Consists of:

- Data fields (store values).




A class is a description of/template for objects.

Consists of:

- Methods (defines what you can do with the objects).
  - Constructor.
  - Operations.

# A SIMPLE CLASS

```
class MyClass:
  pass
```

The name of the class.

```
object_a = MyClass()
object_a.three = 3
print(object_a.three)
object_b = MyClass()
object_b.three = "three"
print(object_b.three)
print(object_a.three)
```

Creates a new instance of the class.

Prints: 3

Prints: three

Prints: 3

# THE CONSTRUCTOR

```
class Circle:
  pass
```

```
class Circle:
  def __init__(self):
    self.radius = 10
```

```
circle_a = Circle()
circle_a.radius = 10
print(circle_a.radius)
circle_b = Circle()
circle_b.radius = 10
print(circle_b.radius)
```

```
circle_a = Circle()
print(circle_a.radius)
circle_b = Circle()
print(circle_b.radius)
```

# THE CONSTRUCTOR

```
class Circle:
  def __init__(self, radius):
    self.radius = radius
```

```
class Circle:
  def __init__(self):
    self.radius = 10
```

```
small_circle = Circle(10)
print(small_circle.radius)
big_circle = Circle(200)
print(big_circle.radius)
```

```
circle_a = Circle()
print(circle_a.radius)
circle_b = Circle()
print(circle_b.radius)
```

# METHODS

```
class Circle:
  def __init__(self, radius):
    self.radius = radius
  def get_area(self):
    return (self.radius ** 2) * 3.14
  def get_perimeter(self):
    return self.radius * 2 * 3.14
```

```
my_circle = Circle(30)
print(my_circle.get_area())      # Prints 2826.
print(my_circle.get_perimeter()) # Prints 188.4.
```

# EXAMPLE

```
class Rectangle:
  def __init__(self, width, height):
    self.width = width
    self.height = height
  def get_area(self):
    return self.width * self.height
  def get_perimeter(self):
    return self.width*2 + self.height*2
```

```
rectangle = Rectangle(10, 20)
print(rectangle.get_area())      # Prints 200.
print(rectangle.get_perimeter()) # Prints 60.
```

# ROOM EXAMPLE

```
class Room:
  def __init__(self, name, side_length_1, side_length_2):
    self.name = name
    self.side_length_1 = side_length_1
    self.side_length_2 = side_length_2
 def get_area(self):
    return self.side_length_1 * self.side_length_2
```

```
room1 = Room("Kitchen", 7, 5)
print(room1.name+" is "+str(room1.get_area()+" m^2 big."))
room2 = Room("Bed Room", 3, 4)
print(room2.name+" is "+str(room2.get_area()+" m^2 big."))
```

# HOUSE EXAMPLE

```
class House:
  def __init__(self):
    self.rooms = []
  def add_room(self, room):
    self.rooms.append(room)
  def get_area(self):
    sum = 0
    for room in self.rooms:
      sum += room.get_area()
    return sum
```

```
my_house = House()
my_house.add_room(Room("Kitchen", 7, 5))
my_house.add_room(Room("Bed Room", 3, 4))
area = my_house.get_area()
print("Area of my house: "+str(area)+".")
```

# LAB SHEET

- Now, please complete the lab sheet using the OOP philosophy.