ITI1120 – Lab#10
Classes and objects

Objectives

- Review the main concepts of Object oriented conception
- Example: The time class
- Exercise 1: Modify the Time class
- Exercise 2: The Car class
- Exercise 3: The Bank class

Some concepts

- Class
  - A class has variables (attributs with values) and methods (operations that execute on variables).
  - Think about the class as a type of complexe variable that is used as a bucket to create objects.
  - The object contain complex (several variables of different types) as well as possible operations with those variables (the methods).
- Objects
  - The variable of reference contains the address (the reference) of the object.

```
obj1 = NameOfTheClass()
obj2 = NameOfTheClass()
```

Some concepts (suite)

- The creation of the object uses the constructor that exists by default with zero parameter,
- Or we can define the constructor __init__

```
obj1 = NameOfTheLaClass( arg list )
```

The role of the constructor is initialize variables of each object.

- Instance variables: the variables defined in the class and created inside the object.
- Instance method: the code that offers oprrations with the objects variables.

Example: A class "Time"

- Assume we want to be able to work with values representing the *time* with one second of precision.
  - What informations do we have to represent the time? How can we keep them?
  - What would those operations using values of type « Time » be.

What to keep in "Time"?

- Two integers:
  - **hour**: number of hours ($0 \leq$ **hour** ≤ 23)
  - **minute**: number of minutes ($0 \leq$ **minute** ≤ 59)
  - **second**: number of secondes ($0 \leq$ **second** ≤ 59)

```
class Time:
    def __init__(self, hh=12, mm=0, s=0):
     '''(Time)-> None'''
     self.hour = hh
     self.minute = mm
     self.second = s
```

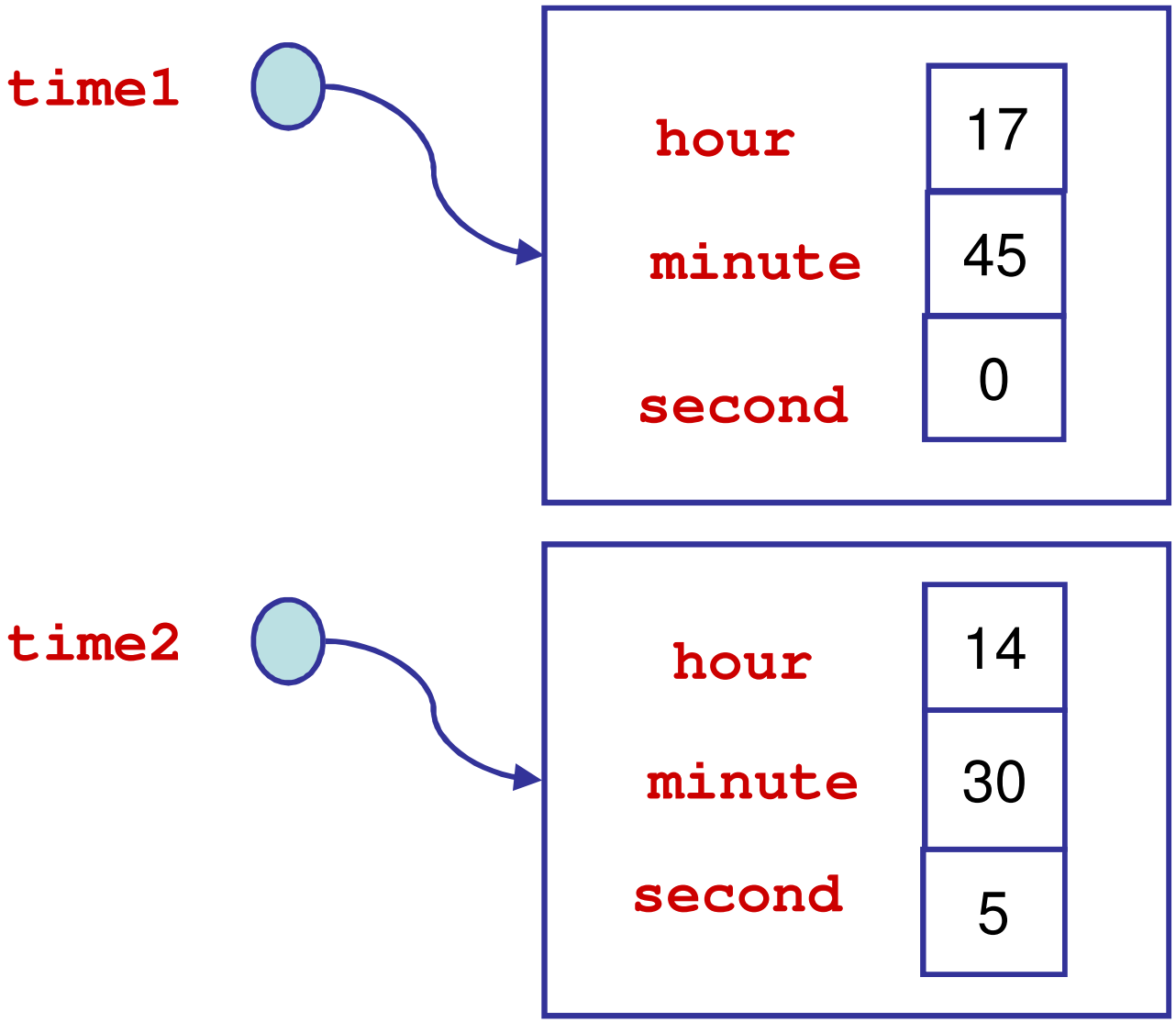
- Are there other ways?
  - Only the number of minutes (or secondes) since midnight?

Object creation of the Time class (use the constructor __init__)

```
>>> time1 = Time(17,45) # Now we have an object Time
                        # that contains ? How many values

>>> time2 = Time(14, 30, 5)  # time2 is ?
```

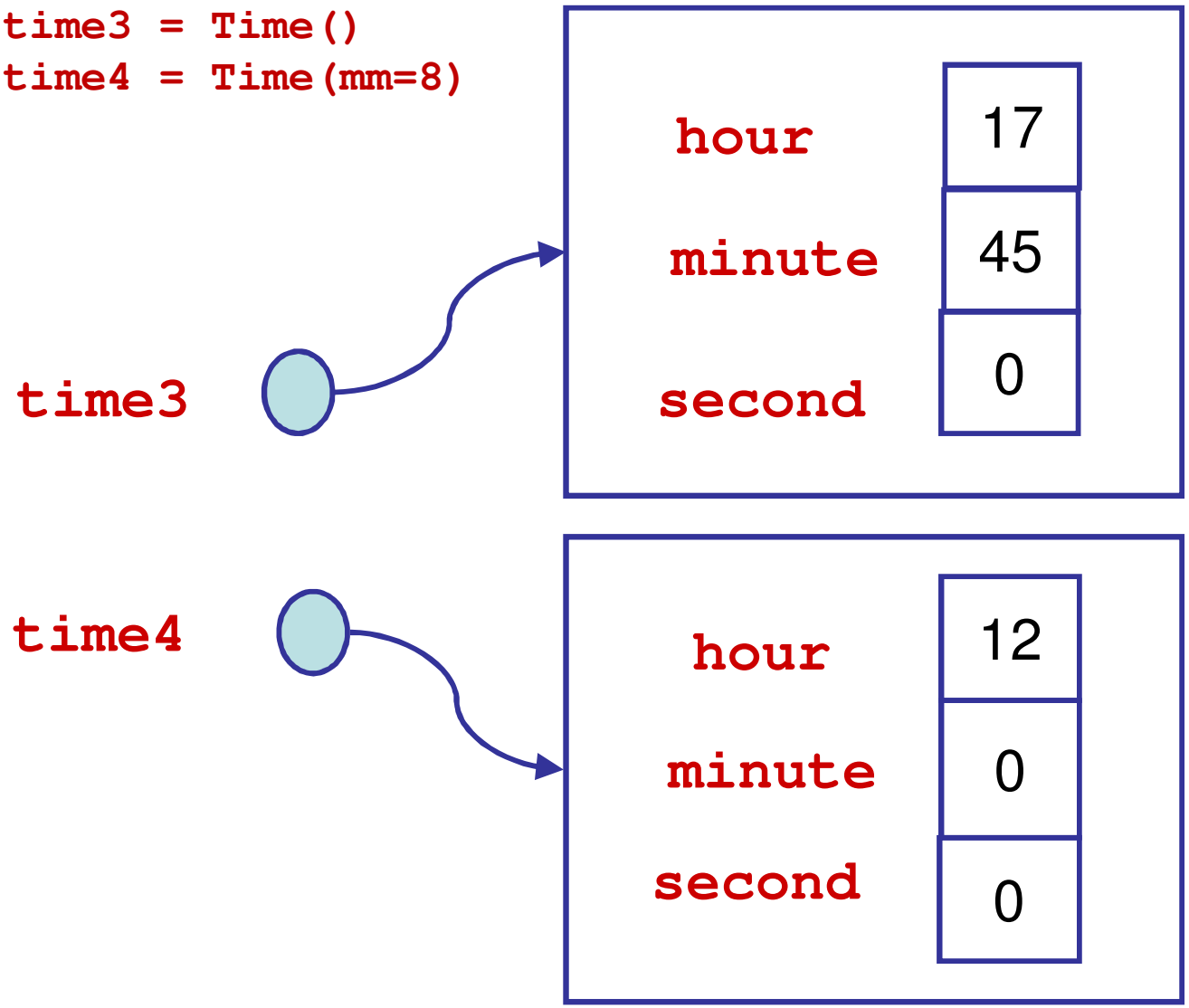
The constructor is used once for each object, when we create un object!!!

What do we have now?



Values by default

```
>>> time3 = Time()
>>> time4 = Time(mm=8)
```



Change the time

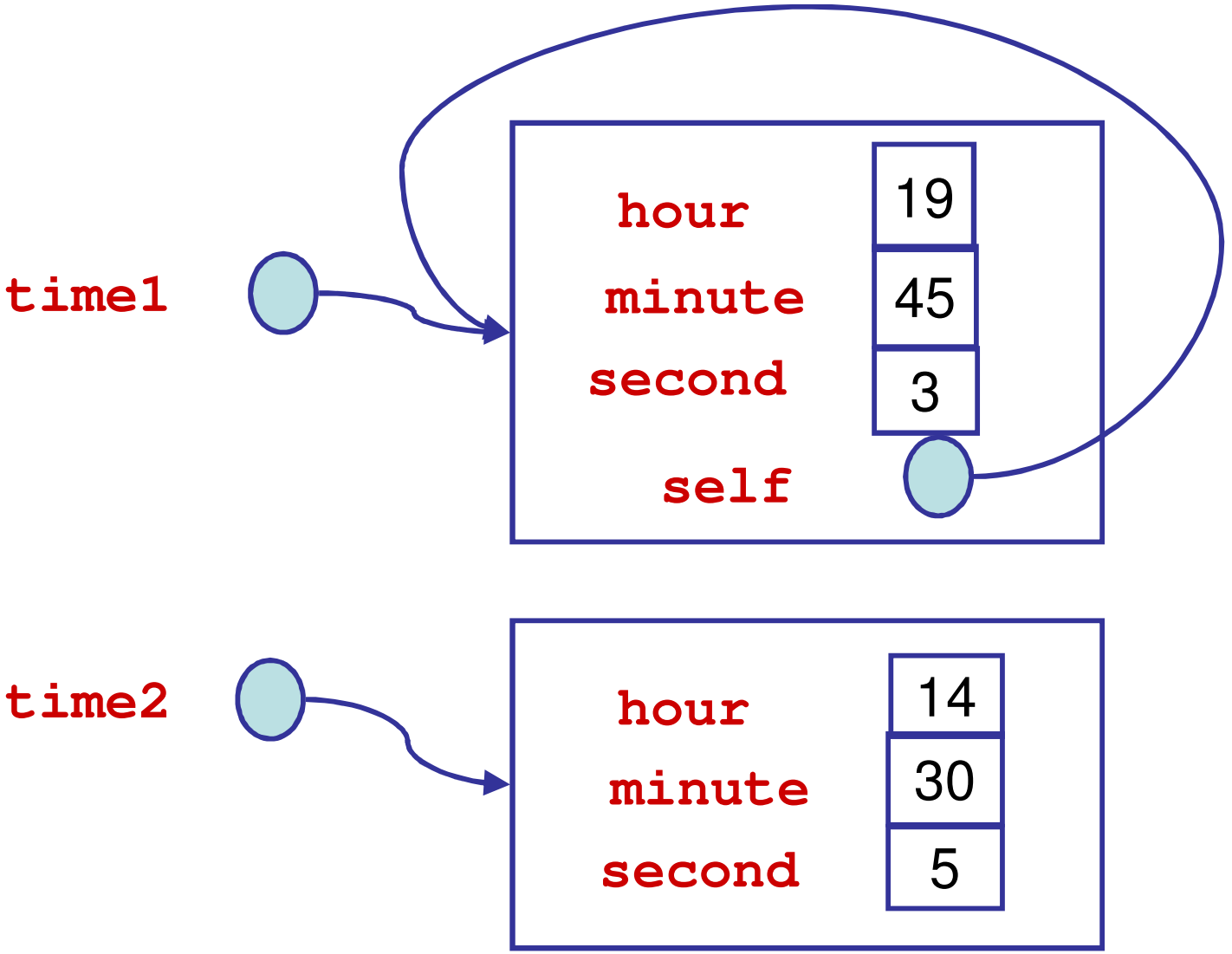
```
>>> time1.hour = 13   #direct access
>>> time1.minute = 20

# add in the class Time
 def setTime(self, hh=12, mm=0, s=0):
    '''(Time)-> None'''
    self.hour = hh
    self.minute = mm
    self.second = s

>>> time1.setTemps(19, 45, 3)
```

The key-word "self"

- Objets time1 and time2 use the same code in the class Time to change their own copies of hour, minute, and second.
- When we want to refer to the object on which the method has been invoked we use self.
- We use self when we descibe the class. Each method takes it as the first parameter by default.

self

```
time1.setTime(19, 45, 3)
```



Display

```
def display_hour(self):
 print("{0}:{1}:{2}".format(self.hour,self.minute,self.second))
>>> time1.display()
19:45:3
>>> print(time1)
<__main__.Time object at 0x02A1BA50>
```

We add a mthod that transforms a object in a chaine of characters that allow the display of a Time object according to the format hour:minute:second in a print

```
def __repr__(self):
 return (str(self.hour) +":" +str(self.minute) +":"
         +str(self.second))
>>> print(time1)
19:45:3
```

Compare Time values: Equality

```
>>> t1 = Time(10,25)
>>> t2 = Time(10,25)  #two similar objects
>>> t1 == t2
False

# if we add in the class:
def __eq__(self, other):
    '''(Time)-> bool'''
    return self.hour == other.hour and
           self.minute == other.minute and
           self.second == other.second

>>> t1 == t2
True
```

Exercise 1: Change the method setTime

- Change the code of the method setTime to make sure that:
  - **hour**: must be in the interval 0 ≤ **hour** ≤ 23
  - **minute**: must be in the interval 0 ≤ **minute** ≤ 59
  - **seconde**: must be in the interval 0 ≤ **second** ≤ 59

```
>>> time1.setTime(25,10,63)
>>> time1
1:11:3
```

- Call this method in __init__ also to correct the values if they are incorrect

Exercise 1 (suite)

Add the folowing methods for the class **Time**:

- **isBefore(t2)**

returns **True** if the time represented by **self** is before the time of **t2**, otherwise **False**.

- **Duration(t2)**

returns a new object **Time** with the number of hours, minutes and seconds between **self** and **t2**.

- Test your methods in the main program or interpretor

```
>> t1 = Time(12,4,0)
>>> t2 = Time(10,2,1)
>>> t1.isBefore(t2)
False
>>> t2.isBefore(t1)
True
```

```
>>> t2.setTime(12,45,2)
>>> t1.isBefore(t2)
True
>>> t1.duration(t2)
0:41:2
```

Exercise 2 – The class Car

Define a class Car() that helps instanciate objects reproducing car behavior.
The constructor of this class will initialize the following instance's attributs with values by default indicated : brand= 'Ford', color = 'red', pilote = 'person', speed = 0.
When we instanciate a new object Car(), we could choose its brand and color but not its speed nor the name of the driver.

The following methods are defined:

- choice_driver(name) will pick (or change) the name of the driver.

Exercise 2: The class Car (suite)

- accelerate(flow, duration) will vary the car speed. The speed variation obtained is equal to the produit: flow × duration. For instance, if the car accelerate at the flow of 1.3 m/s during 20 seconds, its speed gain must be equal to 26 m/s. Negative flows will be accepted (which will allow slowing down). The speed variation will not be allowed if the driver is « person ».
- display_all() will allow to display the actual car properties: its brand, color, its driver name and speed.
- Add the methods __repr__ and __eq__

Examples don how to use them :

```
>>> a1 = Car('Peugeot', 'blue')
>>> a2 = Car(color = ‘green')
>>> a3 = Car('Mercedes')
>>> a1.choice_driver('Roméo')
>>> a2.choice_driver('Juliette')
>>> a2.accelerate(1.8, 12)
>>> a3.accelerate(1.9, 11)
This car does not have a driver !
>>> a2.display_all()
Green Ford driven by Juliette, speed = 21.6 m/s.
>>> a3.display_all()
Red Mercedes driven by person, vitesse = 0 m/s.
```

Exercise 3: The class Bank

- Define a class BankAccount(), that can instanciate objects such as account1, account2, etc. The constructor of the class initializes two instance attributs name and solde, with values by default 'Dupont' and 1000.
- Three methods are defined:
- deposit(sum) allows to add some money to the account ;
- Withdraw(sum() allows to pull some money from the accouunt;
- display() display the name of the account holderand the sold of his account.

Optionnal: add mthods __eq__ and __repr__

Examples on how to use the class :

```
>>> account1 = BankAccount('Duchmol', 800)
>>> account1.depositt(350)
>>> accoint1.withdraw(200)
>>> account1.display()
The solde of the Bank account of Duchmol is 950
dollars.
>>> account2 = BankAccount()
>>> account2.depot(25)
>>> account2.display()
The solde of the Bank account of Dupont is de 1025
dollars.
```

Exercise 3: The class Bank (suite)

- Definie a new class **AccountSaving()**, deriving from class **BankAccount()**, that helps create saving accounts that allow interest to grow with time. Assume that those interests are calculated every month.
- The constructor of the new class will initialize them with a monthly interest rate equal, by default, to **0.3%**. A method **changeRate(value) can** modify that rate as it wishes.
- A method **capitalisation(numberMonth)** should: display the number of months and the interest rate taken into account; calculate the solde reached during the capitalisation, the composed interest, for the rate and number of months chosen.

Examples don to proceed with this class :

```
>>> c1 = AccountSaving('Duvivier', 600)
>>> c1.depositt(350)
>>> c1.display()
The solde of the Bank account for Duvivier is 950 dollars.
>>> c1.capitalisation(12)
Capitalisation on 12 months at the monthly rate of 0.3 %.
>>> c1.display()
The solde of the bank account for Duvivier is 984.769981274
dollars.
>>> c1.changerate(.5)
>>> c1.capitalisation(12)
Capitalisation on 12 months at the monthly rate of 0.5 %.
>>> c1.display()
The solde of the bank account for Duvivier is Duvivier is
1045.50843891 dollars.
```