"Politics is the skilled use of blunt objects."
-- L.B. Pearson

ITI 1120
Module 10: Conception object oriented

General concepts:

1. Classes and objects; 2. UML diagrams and conversion in Python. 3. Constructors; 4. Objects equivalency. 5. Classes with complex fields; 6.Class and instance variables; 7. Heritage. 8. Classes conception

Click to edit Master subtitle style

General objective: Implementation of classes and use of objects

Theme 1. Classes and objets

Sub-theme: Classes and objects

- An object is an entity that we build by instantiating from within a class.
- A class is a « category » or a «type » defined by users or predefined in a library.
- For example, we can find in the library turtle, a class Turtle() from which we can create several turtles.

Sub-theme: Encapsulation

```
from turtle import *


alex = Turtle()
marie = Turtle()
alex.color("blue")
alex.forward(150)
alex.left(90)
alex.forward(75)
marie.color("red")
marie.forward(-150)
marie.right(90)
marie.forward(75)
```

Each object (turtue) has its propreties well separeted:

Each obcjet can run functions/methodes indépendantly from each other (alex versus marie).

Sub-theme: Grouping information

- How to store information on a student for a course?
  - Student number (integer)
  - Midterm (float)
  - Final exanm (float)
  - Registration for credits (boolean)

What is wrong with the following solutions?

- Each value is a separated variable.
- Create 4 lists.
- Put all the values in a list. Each element is a list of 4 values.

Sub-theme: « Records »

- A « record » enables a variable to store several values of different types.
  - Another way to look at it, is like a group of variables of different types.
- The records and the lists are differently important:
  - Each field of a record has a **name**. A value is accessed by indicating the name of the filed.
- Example (a simple record with 4 fields):

StudNum	1234567
Midterm	60.0
Final Exam	80.0
ForCredit	VRAI

Use of records

- Assume that the previous record was stored in a variable E.
- To access the midterm result:

```
E.Midterm
```

- It is referencing a field in the record E. A field can be used similarly to a variable of the same type. For example:

```
T  =  E.Midterm + E.FinalExam

E.ForCredit =  False
```

- The entire record can be used in another affectation or be passed as a parameter.

```
X  =  E
```

Sub-theme: Records and classes

- Few languages allow the creation of records, without the possibility og defining operations on those records. For example:
  - Pascal has « **`record`** »
  - C has « **`struct`** » (structures)

- Other languages allow you also to define operations on those types.
  - Record types are usually called « classes », and the specific records (*instances*) « objects ».
  - Examples:
    - Classes in C++, Java and Python
    - Classes contain attributes (fields)as well as methods to execute operations on the attributes.

Sub-theme: Attributes and methods

- An object can be considered as a record because there is a set of attributes (values with names (variables) stored in the object).

- Each object is created from a class. An object is referenced from a variable of reference (like a table).

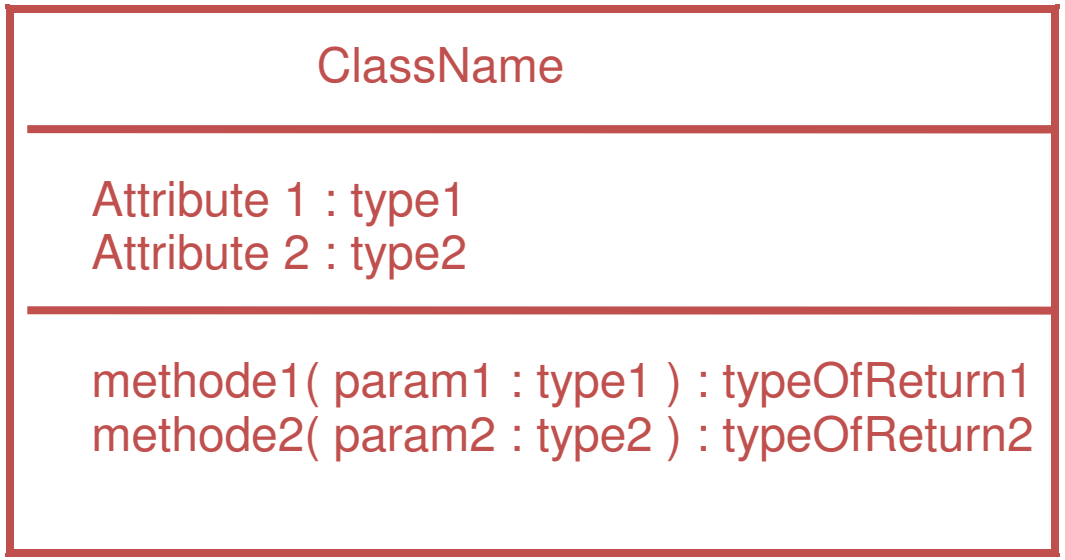
- A “class” can be seen as a «template» that allows creation of objects with sets of identical attributes.
  - The class can also contain methods (algorithms) to perform calculations/transformations on attributes of the object (and/or on external data).

- A **method** is invoked on an object by using the point operator (**.**), like for the records fields.

result ← anObject**.a**Methode(aParameter)

Theme 2. UML Diagrams UML and conversion to Python.

Sub-thème: Class Diagrams



"attributes" are
Like field variables
Of a record

- This form of diagram comes from a normalized notation called "Unified Modeling Language" (or UML).

Sub-theme: Classes and objects in Python

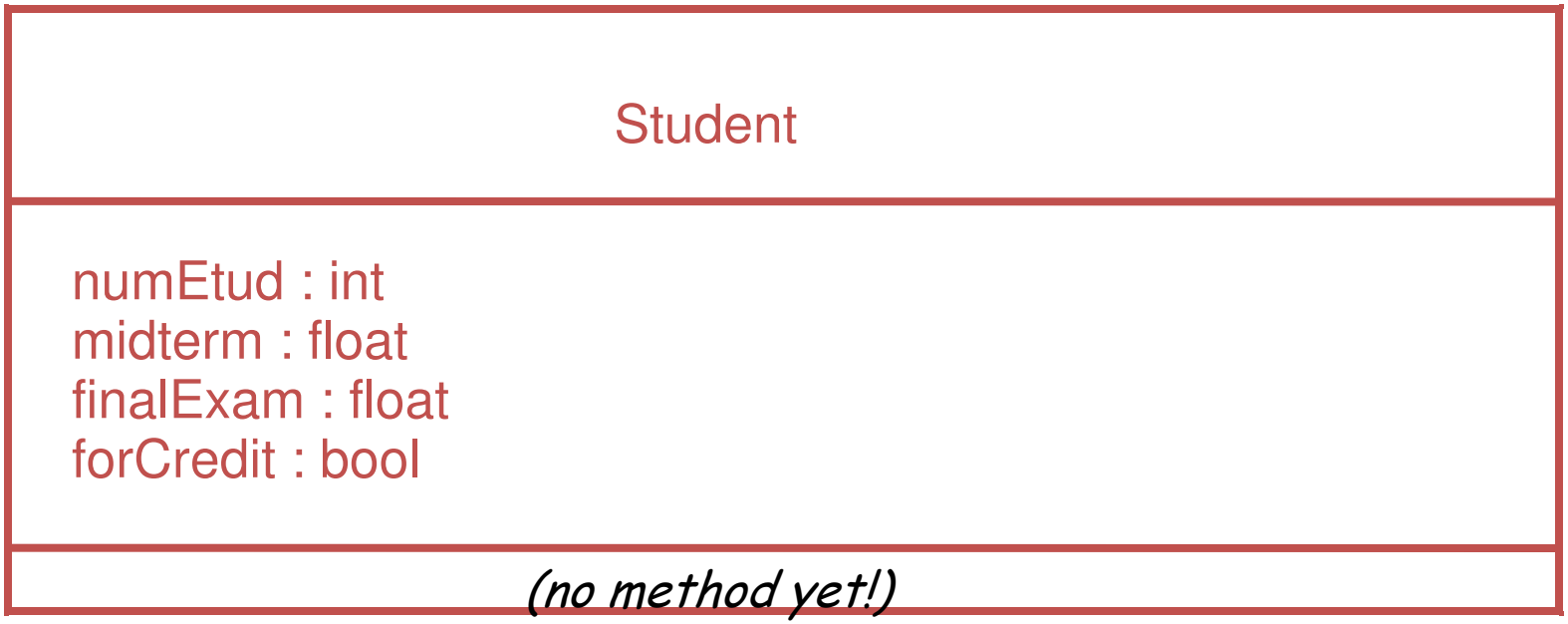
```
class <Class name>:
    ''' Class description
    '''
    # Attributes
    # Methods
```

- This is only the definition of the structure of a record to build objects.
  - an object is referenced by a variable of reference.

```
# Object creation: var = <Name of class>()
```

Sub-theme:First version of the Student class

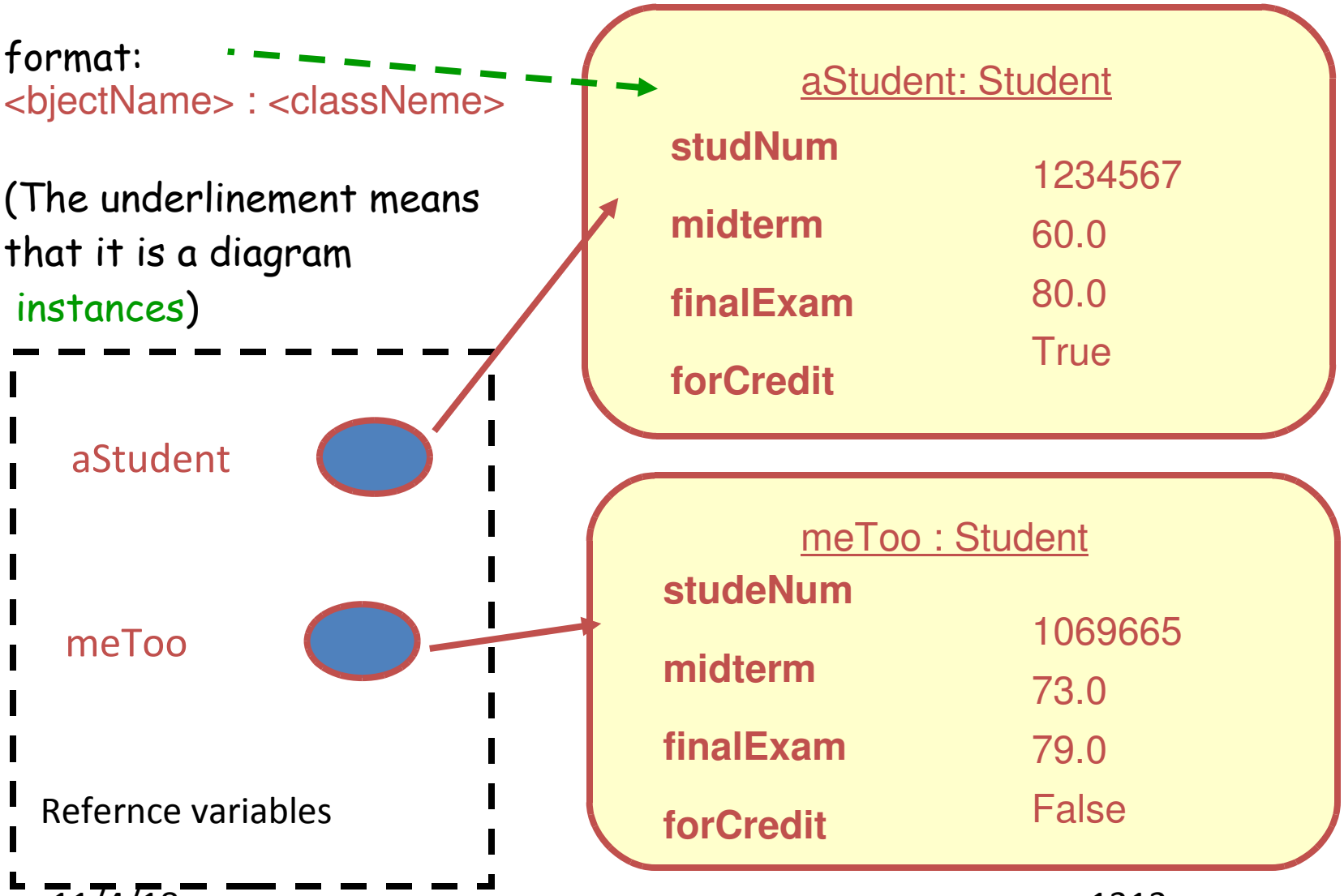
- For each student, we want to store his/her student#, both the midterm and final exams result and confirm that the course is taken for credit (or not). We will bring up his final mark later.



Conversion in Python

```
class Student:

   ''' A student has a student number a midterm and final exam mark as
weel as a Boolean value, true if the course is taken for credit '''

aStudent = Student()  # Object creation, instance etc.

aStudent.stuNum = 1234567

aStudent.midterm= 60.0

aStudent.finalExam = 80.0

aStudent.forCredit = True


meToo = Student()  # Another object creation

meToo.studeNum = 1069665

meToo.midterm = 73.0

meToo.finalExam = 79.0

meToo.forCrédit = False
```

Sub-theme: Use of objects in Python



Sub-theme: The famous `self`

- We distinguish the same field in two different records with the the variable name and th point operator:
  - `aStudent.forCredit` versus `meToo.forCredit`

- Similarly, when a method in the class want to use « the value of the object field on which the method is being invoked », then `self` makes reference to the invoked object.

- During the call `aStudent.getForCredit()`, `self` is referencing `aStudent`
  - … and thus `self.foCrédit` is `aStudent.forCredit`, `that is True`.

- During the call `meToo.getforCrdit()`, `self` is referencing to `meToo`
  - … and thus `self.forCrdit` is `meToo.forCredit`, `that is False`.

Sub-theme: Dissimulating information (*Information Hiding*)

- If we want to modify our objects Student to compute the final mark that is worth 20% of the midterm and 80% of the final exam:
  - We add a method computeFinalmark()

```
0.2*midterm + 0.8*finalExam
```

We could add a field finalMark to our class to store the result after computing it.

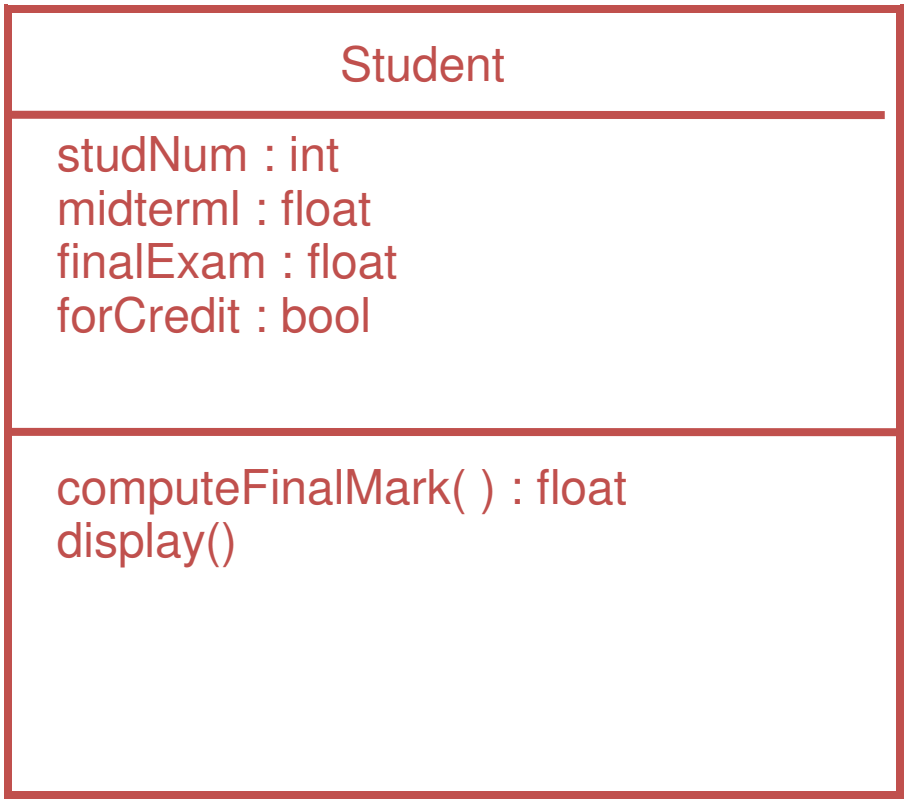
But it would be better to compute it each time because the final mark changes every time when we modify the value **`midterm`** or **`finalExam`**, to be coherent.

- We can also change the formula, without changing the method name

```
0.25*midterm + 0.75*finalExam
```

- Hiding the algorithm or other details is called dissimulating information.

Sub-theme: Student class with mthods and dissimulating information



Conversion to Python

```
class Student:
  # we add methods to the class

  def computeeFinalMark(self):
      return 0.2 * self.midterm + 0.8 * self.finalExam

  def display(self):
      print("Student number", self.studeNum)
      print("Midterm", self.midterm)
      print("Dinal exam", self.finalExam)
      print("For Credit", self.forCredit)

# In the main program we call methods
aStudent.display()
print("Finale mark:", aStudent.computeFinalMark())
meTooAussi.display()
print(" Finale mark:", meToo.computeFinalMark())
```

Sub-theme: Other example: the class Point

```
class Point(object):
    ''' Geometric Point'''

    def dtsplay(self):

print("X",self.x,"Y",self.y,
"Coleur", self.color)
```