Worksheet #2

Classes
Objects
Methods

1. A school intends to computerize its system, which stores information about employees and students. For all people it is intended to save: name (first name and last name), date of birth and telephone number. It should be possible to get people's ages. For employees, it is also intended to save the salary. There are two types of contributors: faculty and staff. For teachers, it is also necessary to keep the subject they teach. It should be possible to assign a percentage raise to any employee. For the remaining employees, it is still necessary to keep the workbook. For students, it is also necessary to save the student number (generated sequentially), the course they attend and the year they are in. It must be possible, at any time, to view the textual description of any of the persons stored in the register. It is intended to computerize this system using object-oriented programming in Python. The necessary classes must be created, with the necessary components to comply with the functionalities described above, as well as two objects for each class, to test the referred functionalities.

   1.1. Person Class:

      1.1.1. Create method `_ _ init _ _` with attributes first name, last name, date of birth and phone
      1.1.2. Create the method `age` , which returns the person's age, according to their date of birth and the current date [1, 2]. It should be possible to use this method as if it were a variable
      1.1.3. Create getter and setter for name. The setter takes a full name (separated by space) and fills in the first name and last name
      1.1.4. `_ _ str _ _` method to return a Person's textual description in the following format (the name must use the getter):
      `Name (age) [phone]`

   1.2. Collaborator Class:

      1.2.1. The Collaborator class inherits from the Person class.
      1.2.2. `_ _ init _ _` method that calls the superclass constructor and also sets the `salary` attribute
      1.2.3. Create the method that returns a collaborator's textual description, using the superclass's textual description and dealing with the subclass's specific variables , in the following format
      `Employee <textual description of Person>: salary €`
      1.2.4. Create the method assignIncrease (pc), which assigns a percentage increase to the employee.

   1.3. Class Teacher

[1] You can use datetime (date) package : https://docs.python.org/3/library/datetime.html
[2] The ages shown in the output examples were calculated on 12/20/2022

1.3.1. The class inherits from the Collaborator class

1.3.2. `_ _ init _ _` method that calls the superclass constructor and also sets the `discipline attribute`

1.3.3. Create two teachers:

	name	Birth date	telephone	Wage	discipline
pr1	Gustavo Gomes	date(1977,1,1)	987654321	800	Schedule
pr2	Hugo Humberto	date(1987,3,8)	912584684	800	Networks

1.3.4. Create the method that returns the textual description of a teacher, resorting to the textual description of the superclass and dealing with the subclass specific variables and test for the two existing teachers. Example:

```
Professor Colaborador Gustavo Gomes (45) [987654321]: 800€ => Programação
Professor Colaborador Hugo Humberto (35) [912584684]: 800€ => Redes
```

1.3.5. Assign a 5% raise to teacher p1 (salary should now be 840)

1.3.6. Change teacher name pr2 to Hugo Horta

1.4. Employee Class

1.4.1. The class inherits from the Collaborator class

1.4.2. `_ _ init _ _` method that calls the superclass constructor and also sets the `block attribute`

1.4.3. Create two employees:

	name	Birth date	telephone	wage	block
f1	Igor Ílhavo	date(1990,12,12)	954785632	700	A
f2	Joana Jacinto	date(1992,2,12)	974521365	700	B

1.4.4. Create the method that returns the textual description of an employee, calling the superclass's textual description and handling the subclass-specific variables, and testing for the two existing employees. Example:

```
Funcionário Colaborador Igor Ilhavo (32) [954785632]: 700€ => Bloco: A
Funcionário Colaborador Joana Jacinto (30) [974521365]: 700€ => Bloco: B
```

1.4.5. Assign a 7.5% raise to employee f1 (salary should now be 752.5)

1.4.6. Change the f2 employee name to Joana Jardim

1.5. Student Class

1.5.1. The class inherits from the Person class

1.5.2. nrStudents class variable , which starts at 0

1.5.3. `_ _ init _ _` method that calls the superclass constructor and also defines the attributes `nrStudent` (sequentially generated), `course` and `year`

1.5.4. Create two students:

	name	Birth date	telephone	course	year
to 1	laura lis	date(2000,1,1)	957432658	Programmer	1
a2	Miguel Moreira	date(2000,5,21)	916845239	Adm. Networks	3

1.5.5. Create the method that returns a student's textual description, calling the superclass's textual description and dealing with subclass-specific variables, and testing for the two existing students. Example:

```
Aluno nº 1 - Laura Lis (22) [957432658] => 1º ano do curso de Programador
Aluno nº 2 - Miguel Moreira (22) [916845239] => 3º ano do curso de Adm. Redes
```

1.5.6. Change the name of student a2 to “Mário Moreira”.

In the end, it should be possible to run the following code, obtaining the output shown below.

Code:

```
pr1 = Professor("Gustavo", "Gomes", date(1977, 1, 1), 987654321, 800, "Programação")
pr2 = Professor("Hugo", "Humberto", date(1987, 3, 8), 912584684, 800, "Redes")
print(pr1)
print(pr2)
pr1.atribuirAumento(5)
print(pr1)
pr2.nome = "Hugo Horta"
print(pr2)
f1 = Funcionario("Igor", "Ilhavo", date(1990, 12, 12), 954785632, 700, "A")
f2 = Funcionario("Joana", "Jacinto", date(1992, 2, 12), 974521365, 700, "B")
print(f1)
print(f2)
f1.atribuirAumento(7.5)
print(f1)
f1.nome = "Joana Jacinto"
a1 = Aluno("Laura", "Lis", date(2000, 1, 1), 957432658, "Programador", 1)
a2 = Aluno("Miguel", "Moreira", date(2000, 5, 21), 916845239, "Adm. Redes", 3)
print(a1)
print(a2)
a2.nome = "Mário Moreira"
print(a2)
```

Output:

```
Professor Colaborador Gustavo Gomes (45) [987654321]: 800€ => Programação
Professor Colaborador Hugo Humberto (35) [912584684]: 800€ => Redes
Professor Colaborador Gustavo Gomes (45) [987654321]: 840.0€ => Programação
Professor Colaborador Hugo Horta (35) [912584684]: 800€ => Redes
Funcionário Colaborador Igor Ilhavo (32) [954785632]: 700€ => Bloco: A
Funcionário Colaborador Joana Jacinto (30) [974521365]: 700€ => Bloco: B
Funcionário Colaborador Igor Ilhavo (32) [954785632]: 752.5€ => Bloco: A
Aluno nº 1 - Laura Lis (22) [957432658] => 1º ano do curso de Programador
Aluno nº 2 - Miguel Moreira (22) [916845239] => 3º ano do curso de Adm. Redes
Aluno nº 2 - Mário Moreira (22) [916845239] => 3º ano do curso de Adm. Redes
```

2. A bank wants to computerize its accounts. In general, for each account, it is intended to save the account number (generated sequentially), the name of the holder, the balance (initial value: 0), the movements (list of tuples (type, value, balance)) and the number of movements (calculated from the movements). There are two types of account: current accounts and term accounts. For current accounts, it is also necessary to keep the withdrawal amount limit. These accounts allow you to make deposits (no limit) and withdrawals (with a defined maximum amount). All term accounts have an interest tax of 28%. For this type of account it is still necessary to save the interest rate. These accounts only allow deposits. Create the necessary Python code to run the following code, obtaining the output shown below.

Code:

```
c1 = Ordem("Zé", 200)
print(c1)
c1.deposito(300)
c1.levantamento(10)
c1.levantamento(250)
print(c1)
print("=====================================================")
c2=Prazo("Ana",0.1)
print(c2)
c2.deposito(100)
c2.deposito(200)
print(c2)
```

Output:

```
Conta à ordem: limite levantamento = 200€ - Conta nº 1 | Titular: Zé | Saldo: 0€ | 1 movimentos:
    - Ini   0   0
Depóstito de 300€ efetuado. Saldo atual = 300€
Levantamento de 10€ autorizado. Saldo atual = 290€
Levantamento de 250€ não autorizado.
Conta à ordem: limite levantamento = 200€ - Conta nº 1 | Titular: Zé | Saldo: 290€ | 3 movimentos:
    - Ini   0   0
    - Dep   300 300
    - Lev   10  290
=====================================================
Conta a prazo: taxa = 0.1% (imposto = 28%) - Conta nº 2 | Titular: Ana | Saldo: 0€ | 1 movimentos:
    - Ini   0   0
Depósito: 100€ | Juro bruto anual: 0.1€ | Juro líquido anual: 0.072€ | saldo atual: 100€
Depósito: 200€ | Juro bruto anual: 0.3€ | Juro líquido anual: 0.216€ | saldo atual: 300€
Conta a prazo: taxa = 0.1% (imposto = 28%) - Conta nº 2 | Titular: Ana | Saldo: 300€ | 3 movimentos:
    - Ini   0   0
    - Dep   100 100
    - Dep   200 300
```