Ficha de trabalho #2

Classes
Objetos
Métodos

1. Uma escola pretende informatizar o seu sistema, que guarda informação sobre colaboradores e alunos. Para todas as pessoas é pretendido guardar: nome (primeiro nome e último nome), data de nascimento e telefone. Deve ser possível obter a idade das pessoas. Para os colaboradores é ainda pretendido guardar o salário. Há dois tipos de colaboradores: professores e funcionários. Para os professores é também necessário guardar a disciplina que lecionam. Deve ser possível atribuir um aumento percentual a qualquer colaborador. Para os restantes funcionários é necessário guardar ainda o bloco de trabalho. Para os alunos é necessário guardar ainda o nº de aluno (gerado sequencialmente), o curso que frequentam e o ano em que se encontram. Deve ser possível, a qualquer altura, visualizar a descrição textual de qualquer uma das pessoas guardadas no registo. Pretende-se informatizar este sistema recorrendo a Programação orientada a objetos em Python. Devem ser criadas as classes necessárias, com os componentes necessários para obedecer às funcionalidades acima descritas, bem como dois objetos por cada classe, para testar as funcionalidades referidas.

   1.1. Classe Pessoa:

      1.1.1. Criar o método `_ _init_ _` com os atributos primeiro nome, último nome, data de nascimento e telefone
      1.1.2. Criar o método `idade`, que retorna a idade da pessoa, de acordo com a sua data de nascimento e a data atual[1,2]. Deve ser possível utilizar este método como se fosse uma variável
      1.1.3. Criar getter e setter para nome. O setter recebe um nome completo (separado por espaço) e preenche o primeiro nome e o último nome
      1.1.4. Criar o método `_ _str_ _` para devolver a descrição textual de uma Pessoa no seguinte formato (o nome deve usar o getter):
      `Nome (idade) [telefone]`

   1.2. Classe Colaborador:

      1.2.1. A classe Colaborador herda da classe Pessoa
      1.2.2. Criar o método `_ _init_ _` que chama o construtor da superclasse e para além disso define o atributo `salário`
      1.2.3. Criar o método que devolve a descrição textual de um colaborador, recorrendo à descrição textual da superclasse e lidando com as variáveis específicas da subclasse, no seguinte formato
      `Colaborador <descrição textual de Pessoa>: salário €`
      1.2.4. Criar o método atribuirAumento(pc), que atribui um aumento percentual ao colaborador.

[1] Pode usar o package datetime (date): https://docs.python.org/3/library/datetime.html
[2] As idades apresentadas nos exemplos de output foram calculadas a 20/12/2022

1.3. Classe Professor

1.3.1. A classe herda da classe Colaborador

1.3.2. Criar o método `_ _init_ _` que chama o construtor da superclasse e para além disso define o atributo `disciplina`

1.3.3. Criar dois professores:

	nome	Data nascimento	telefone	Salário	disciplina
pr1	Gustavo Gomes	date(1977,1,1)	987654321	800	Programação
pr2	Hugo Humberto	date(1987,3,8)	912584684	800	Redes

1.3.4. Criar o método que devolve a descrição textual de um professor, recorrendo à descrição textual da superclasse e lidando com as variáveis específicas da subclasse e testar para os dois professores existentes. Exemplo:

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

1.3.5. Atribuir um aumento de 5% ao professor p1 (o salário deve passar a ser 840)

1.3.6. Alterar o nome do professor pr2 para Hugo Horta

1.4. Classe Funcionário

1.4.1. A classe herda da classe Colaborador

1.4.2. Criar o método `_ _init_ _` que chama o construtor da superclasse e para além disso define o atributo `bloco`

1.4.3. Criar dois funcionários:

	nome	Data nascimento	telefone	salário	bloco
f1	Igor Ílhavo	date(1990,12,12)	954785632	700	A
f2	Joana Jacinto	date(1992,2,12)	974521365	700	B

1.4.4. Criar o método que devolve a descrição textual de um funcionário, recorrendo à descrição textual da superclasse e lidando com as variáveis específicas da subclasse e testar para os dois funcionários existentes. Exemplo:

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

1.4.5. Atribuir um aumento de 7.5% ao funcionário f1 (o salário deve passar a ser 752.5)

1.4.6. Alterar o nome do funcionário f2 para Joana Jardim

1.5. Classe Aluno

1.5.1. A classe herda da classe Pessoa

1.5.2. A classe contém uma variável de classe nrAlunos, que inicia a 0

1.5.3. Criar o método _ _init_ _ que chama o construtor da superclasse e para além disso define os atributos `nrAluno` (gerado sequencialmente), `curso` e ano

1.5.4. Criar dois alunos:

	nome	Data nascimento	telefone	curso	ano
a1	Laura Lis	date(2000,1,1)	957432658	Programador	1
a2	Miguel Moreira	date(2000,5,21)	916845239	Adm. Redes	3

1.5.5. Criar o método que devolve a descrição textual de um aluno, recorrendo à descrição textual da superclasse e lidando com as variáveis específicas da subclasse e testar para os dois alunos existentes. Exemplo:

```
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. Alterar o nome do aluno a2 para "Mário Moreira".

No final, deve ser possível executar o seguinte código, obtendo o output apresentado de seguida.
Código:

```
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. Um banco pretende informatizar as suas contas. No geral, para cada conta, é pretendido guardar o número de conta (gerado sequencialmente), o nome do titular, o saldo (valor inicial: 0), os movimentos (lista de tuplos (tipo, valor, saldo)) e o número de movimentos (calculado a partir dos movimentos). Há dois tipos de conta: contas à ordem e contas a prazo. Para as contas à ordem é também necessário guardar o limite do valor do levantamento. Estas contas permitem efetuar depósitos (sem limite) e levantamentos (com valor máximo definido). Todas as contas a prazo têm um imposto sobre juros de 28%. Para este tipo de conta é ainda necessário guardar a taxa de juro. Estas contas apenas permitem depósitos. Elaborar o código Python necessário para que seja possível executar o seguinte código, obtendo o output apresentado de seguida.

   Código:

```
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
```