Ficha de trabalho #3

Módulos
Recursividade

1. Criar um *package* `classesEscola` e lá dentro adaptar o exercício 1 da ficha sobre "Classes, objetos e métodos" para utilizar a organização por módulos. No final, deve o *package* deve ter os seguintes ficheiros:

- **`pessoa.py`**: contém todo o código necessário para a definição da classe Pessoa
- **`colaborador.py`**: contém todo o código necessário para a definição da classe Colaborador, que herda de Pessoa (*import* do módulo pessoa)
- **`professor.py`**: contém todo o código necessário para a definição da classe Professor, que herda de Colaborador (*import* do colaborador)
- **`funcionario.py`**: contém todo o código necessário para a definição da classe Funcionario, que herda de Colaborador (*import* do módulo colaborador)
- **`aluno.py`**: contém todo o código necessário para a definição da classe Aluno, que herda de Pessoa (*import* do módulo pessoa)
- **`main.py`**: contém os *imports* e o código necessários para mostrar o mesmo output que no final do exercício da ficha anterior:

```
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. Criar um *package* `classesBanco` e lá dentro adaptar o exercício 2 da mesma ficha para utilizar a organização por módulos. No final, ao executar o ficheiro `main.py`, deve ser obtido o mesmo 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
```

3. Criar as funções recursivas que implementem as seguintes funcionalidades:

   3.1. Imprimir valores de n até 0. Para chamar a função, o código deve pedir o valor n ao utilizador e garantir que este não é menor que 0.

   Exemplo:

Input	Output
3	3 2 1 0

   3.2. Soma consecutivamente valores de 1 a n. Para chamar a função, o código deve pedir o valor n ao utilizador e garantir que este não é menor que 0.

   Exemplo:

Input	Output
3	`6 # (3+2+1)`

   3.3. Executa uma multiplicação fazendo somas sucessivas. Para chamar a função, os valores devem ser pedidos ao utilizador e deve ser garantido que nenhum deles é inferior a 1.

Input	Output
3 4	`12 # (4+4+4)`

1.1. Colocar as funções recursivas desenvolvidas num ficheiro (módulo) `recursivas.py`. Criar um ficheiro `main.py`, que contém código para mostrar utilizador um menu que lhe permite escolher que função recursiva pretende utilizar a partir desse módulo.

1.2. Criar um *package* chamado `funcoes` e colocar o módulo `recursivas.py` dentro desse *package*. O ficheiro `main.py` deve ficar fora do package, mas manter as funcionalidades.