Instituto Superior de Contabilidade e Administração de Coimbra
Instituto Politécnico de Coimbra

Caderno de Exercícios

Programação

(UC do curso em Ciência de Dados para a Gestão)

António Trigo

2023/2024

Conteúdo

Capítulo 1

Exercícios introdutórios de Python

1.1 Primeiros exercícios

Para a realização destes exercícios recomenda-se a utilização do jupyter notebook através do Google Colab ou do Visual Studio Code da Microsoft com uma instalação local do Python.

Cada exercício deverá ser realizado numa ou mais células de código do jupyter notebook.

1.1.1. Escreva numa célula de código do jupyter notebook uma instrução para mostrar o seu nome no ecrã.

1.1.2. Escreva uma instrução que mostre no ecrã o resultado da operação de 3 + 4.

1.1.3. Realize a instrução anterior mas questionando o utilizador pelos valores de entrada (sugestão: utilize a instrução input).

1.1.4. Escreva um programa que questione o utilizador pela sua idade e mostre no ecrã a idade do utilizador daqui a 30 anos (sugestão: utilize uma variável para guardar a idade).

1.1.5. Escreva uma instrução que mostre o tipo de dados de uma variável utilizada para guardar um valor inteiro (ex. x = 5, sugestão: type(x)).

1.1.6. Crie um algoritmo para converter um valor de milhas para kms.

1.1.7. Crie um algoritmo para calcular a área de um quadrado.

1.1.8. Crie um algoritmo para calcular a área de uma circunferência.

1.1.9. Crie um algoritmo que converta temperaturas em graus Celsius para graus Fahrenheit

1.1.10. Crie um algoritmo que calcule a distância Euclidiana entre dois pontos.

1.1.11. Crie um algoritmo que troque o valor de duas variáveis.

1.1.12. Escreva uma instrução para verificar a versão de Python que está a utilizar.

Capítulo 2

Exercícios sobre Estruturas de Controlo de Fluxo

2.1 Estruturas de seleção

2.1.1. Crie um algoritmo que verifique se um número introduzido pelo utilizador é positivo, negativo ou zero.

2.1.2. Crie um algoritmo que verifique se um número introduzido pelo utilizador é par.

2.1.3. Crie um algoritmo que apresente o maior de dois números inseridos pelo utilizador.

2.1.4. Crie um algoritmo que Verifique se uma pessoa pode votar com base na idade.

2.1.5. Crie um algoritmo que verifique se um determinado ano inserido pelo utilizador é bissexto.

2.1.6. Crie um algoritmo que verifique se um carater introduzido pelo utilizador é uma vogal.

2.1.7. Crie um algoritmo que com base nos ângulos de um triângulo determine se os mesmos correspondem a um triângulo válido.

2.1.8. Crie um algoritmo que verifique, com base nos lados do triângulo, se o mesmo é equilátero, isósceles ou escaleno.

2.1.9. Crie um algoritmo que calcule o índice de massa corporal (IMC) do utilizador e apresente se o mesmo é magro (<18.5), normal (entre 18.5 e 24.9), excesso de peso (25 a 29.9) ou obeso (>29.9).

2.1.10. Crie um algoritmo que determine o maior de três números.

2.1.11. Crie um algoritmo que verifique se uma letra é vogal ou consoante.

2.1.12. Crie um algoritmo que classifique três números em ordem crescente.

2.1.13. Crie um algoritmo que determine o dia da semana com base em um número (1 para Domingo, 2 para Segunda, etc.).

2.1.14. Crie um algoritmo que verifique se um número está dentro de um intervalo específico.

2.1.15. Crie um algoritmo que determine a estação do ano com base no mês.

2.1.16. Crie um algoritmo que determine o valor absoluto de um número.

2.2 Estruturas de repetição

2.2.1. Crie um algoritmo que apresente no ecrã os números de 1 a 100.

2.2.2. Crie um algoritmo que imprima os números pares de 2 a 20.

2.2.3. Crie um algoritmo que imprima os múltiplos de 5 de 10 a 50.

2.2.4. Crie um algoritmo que apresente no ecrã os números de 100 a 1.

2.2.5. Crie um algoritmo que apresnete no ecrã a tabuada do 5.

2.2.6. Crie um algoritmo que apresente no ecrã a tabuada de um número introduzido pelo utilizador.

2.2.7. Crie um algoritmo que some todos os números de 1 a 100.

2.2.8. Crie um algoritmo que some todos os números pares de 1 a 100.

2.2.9. Escreva um algoritmo que receba dois números inteiros e gere os números inteiros que estão no intervalo compreendido por eles.

2.2.10. Crie um algoritmo que calcule o fatorial de um número.

2.2.11. Escreva um algoritmo que mostre os n termos da seguinte série: $S = 1/1 + 2/3 + 3/5 + 4/7 + 5/9 + ... + n/m$. Imprima no final a soma da série.

2.2.12. Crie um algoritmo que devolva os divisores de um número inserido pelo utilizador.

2.2.13. Crie um algoritmo para verificar se um número é primo.

2.2.14. Escreva um algoritmo que leia uma sequência de números inteiros a partir do teclado e apresente o máximo e o mínimo. O algoritmo termina quando for lido o 0.

2.2.15. Crie um algoritmo que verifique se um número é uma capicua.

2.2.16. Implemente o jogo do adivinho em que o utilizador tenta adivinhar um número gerado automaticamente pelo computador (sugestão: *random.randint(a, b)*)

2.2.17. Supondo que a população do país A é de 80000 habitantes com uma taxa anual de crescimento de 3% e que a população do país B é 200000 habitantes com uma taxa de crescimento de 1.5%. Escreva um algoritmo que calcule e escreva o número de anos necessários para que a população do país A ultrapasse ou igual a população do país B, mantidas as taxas de crescimento.

2.2.18. Qual é o menor número inteiro positivo, tal que, se retirarmos o algarismo das unidades e o colocarmos do lado esquerdo, obtemos um número 5 vezes maior. Escreva um algoritmo que resolva o problema proposto.

2.2.19. Escreva um programa que leia 20 números entre 10 e 30 (os números lidos devem ser validados) e apresente o produto dos que pertencerem ao intervalo [10,20].

2.2.20. Crie um algoritmo que calcule a média de uma sequência de números fornecida pelo utilizador. Termina quando for introduzido um 0.

2.2.21. Crie um algoritmo que que conte e imprima a quantidade de números pares e ímpares em uma sequência de números fornecida pelo utilizador. Termina quando for introduzido um 0.

2.2.22. Crie um algoritmo que determine o número de número da maior sequência crescente de um conjunto de números introduzidos pelo utilizador. ermina quando for introduzido um 0.

2.2.23. Crie um algoritmo que apresente um triângulo de asteriscos:
*
**
***
****

2.2.24. Crie um algoritmo que desenhe um triângulo de números:
1
12
123
1234

2.2.25. Crie um algoritmo que desenhe uma pirâmide de números:
__1
_121
12321

2.2.26. Crie um algoritmo que gere uma palavra-passe aleatória de um comprimento especificado, incluindo uma mistura de letras maiúsculas, letras minúsculas, dígitos e caracteres especiais.

2.2.27. Crie um algoritmo que simule um jogo básico de "pedra, papel e tesoura"contra o computador. A escolha do computador deve ser aleatória.

2.2.28. Escreva um algoritmo que converta a numeração Romana em Árabe.

2.2.29. Escreva um algoritmo que converta a numeração Árabe em Romana.

2.2.30. Um número perfeito é um número inteiro positivo para o qual a soma de todos os seus divisores inteiros positivos próprios (excluindo ele mesmo) é igual ao próprio número. Escreva um algoritmo que apresente no ecrã todos os números perfeitos até 10000.

2.2.31. Um número primo de Mersenne é um número de Mersenne (número da forma $M(n) = 2^(n) - 1$, com n um número inteiro positivo) que também é um número primo. Escreva um algoritmo que determine todos os números de Mersenne até n = 61.

2.2.32. Escreva um algoritmo que avalie se um NIF introduzido pelo utilizador é válido.

2.2.33. Escreva um algoritmo que apresente no ecrã todas as capicuas até 10000.

2.2.34. Qual é o menor múltiplo de 182 composto apenas pelo dígito 4?

2.2.35. Qual é o menor múltiplo de 416 composto apenas pelos dígitos 1 e 2?

2.2.36. Escreva um algoritmo que determine o maior número ímpar e o menor par introduzidos pelo utilizador. A introdução de números pelo utilizador termina quando for introduzido um número negativo.

2.2.37. Escreva um algoritmo que leia 15 números entre 15 e 50 (os números lidos devem ser validados) e apresente a soma dos números primos lidos.

Capítulo 3

Exercícios sobre Funções

3.1 Funções

3.1.1. Escreva e teste uma função que calcule a soma de dois números.

3.1.2. Escreva e teste uma função que devolva a área e o perímetro de um quadrado.

3.1.3. Escreva e teste uma função que simule as operações de uma máquina de calcular.

3.1.4. Escreva uma função que determine se um número é primo e depois utilize essa função para gerar os números primos existentes entre 10 e 90.

3.1.5. Escreva uma função que simule o lançamento de um dado (valores entre 1 e 6) e depois invoque-a 100 vezes apresentando as estatísticas dos lados que saíram.

3.1.6. Implemente uma função que receba um número e calcule a soma de todos os dígitos desse número.

3.1.7. Desenvolva uma função que receba uma string e retorne o número de vogais nela.

3.1.8. Escreva uma função que receba um número inteiro e determine se ele é um número perfeito, retornando True ou False.

3.1.9. Implemente uma função que calcule e retorne o valor de PI usando a fórmula de Leibniz, com um número específico de termos.

3.1.10. Escreva uma função que gere e retorne uma sequência de números primos até um número "n"especificado como argumento.

3.1.11. Crie uma função que receba dois números inteiros e determine se um é múltiplo do outro, retornando True ou False.

3.1.12. Desenvolva uma função que calcule e retorne o máximo divisor comum (MDC) de dois números inteiros fornecidos como argumentos.

3.1.13. Crie uma função que determine e retorne o número de dígitos em um número inteiro fornecido como argumento.

3.1.14. Escreva e teste uma função que devolva o fatorial de um número.

3.1.15. Escreva e teste uma função recursiva que devolva o fatorial de um número.

3.1.16. Escreva uma função recursiva que calcule a série de Fibonacci.

3.2 Funções lambda

Funções lambda em Python são funções anónimas, ou seja, são funções que não têm um identificador/nome associado a elas. Elas são úteis quando há necessidade de utilizar funções simples e rápidas, geralmente usadas como argumentos em outras funções ou em operações de mapeamento e filtragem (mais quando chegarmos às listas).

Sintaxe: [lambda argumentos: expressão]

3.2.1. Crie uma função *lambda* que adicione 10 a um número introduzido pelo utilizador.

3.2.2. Crie uma função *lambda* que multiplique dois números introduzidos pelo utilizador.

3.2.3. Escreva uma função lambda que eleve um número ao quadrado.

3.2.4. Escreva uma função lambda que calcule a média de dois números.

Capítulo 4

Exercícios sobre Estruturas de Dados em Python

Este capítulo aborda as estruturas de dados existentes no Python, nomeadamente, as listas, tuplos, conjuntos, dicionários e *strings* (texto).

4.1 Listas

4.1. Crie uma lista com cinco elementos de vários tipos de dados e apresente-a no ecrã.

4.2. Mostre o elemento que está na posição 3 no ecrã.

4.3. Altere o valor do elemento da posição 2 da lista e mostre novamente a lista no ecrã.

Crie uma lista vazia chamada frutas e execute as seguintes operações:

- Adicione os elementos "maçã", "banana"e "laranja".
- Altere "maçã"para "uva".
- Remova a palavra banana da lista de frutas.
- Apresente cada um dos itens da lista.

4.4. Apresente no ecrã o tamanho de uma lista definida por si.

4.5. Crie uma lista com cinco nomes, adicione um novo nome à lista e apresente a lista no ecrã.

4.6. Crie uma lista com 7 inteiros e apresente no ecrã a soma dos mesmos de duas formas: 1) utilizando a função *sum()* disponível no Python e 2) utilizando um ciclo.

4.7. Apresente a soma dos números pares existentes numa lista definida por si.

4.8. Apresente a soma dos números que estão nas posições pares da lista.

4.9. Apresente todos os elementos de uma lista de frutas [’ananas’,’banana’,’laranja’] utilizando dois ciclos diferentes, um com índice e outro sem índice.

4.10. Defina uma lista com 7 inteiros e utilizando o *slicing*:

- Apresente o primeiro elemento.
- Apresente o 3,4 e 5 elementos.
- Apresente o último elemento.
- Apresente os elementos que estão nas posições múltiplas de 3.

4.11. Crie uma lista chamada notas contendo números de 1 a 20 e faça os seguintes procedimentos:

- Calcule a média das notas na lista notas.
- Encontre o valor máximo na lista notas.
- Encontre o valor mínimo na lista notas.
- Conte quantas vezes o número 5 aparece na lista notas.

4.12. Crie uma função que receba uma lista de inteiros e devolva um lista com os números primos existentes nessa lista.

4.13. Crie uma função que receba uma lista de inteiros e devolva o elemento que aparece mais vezes na lista e o respetivo número de vezes.

4.14. Crie uma função que receba uma lista de inteiros e devolva o máximo da lista e o número de vezes que ele ocorre.

4.15. Aplique as funções *built-in* do Python (*type(obj), len(obj), sorted(obj), sum(obj), min(obj), max(obj) e abs(numero)*) à seguinte lista de valores reais [1.2,1.5,3.6,0.9,0.8] e comente o resultado das operações.

4.16. Utilizando as *list comprehension* crie uma nova lista a partir de uma lista de inteiros em que os novos inteiros são o triplo da lista original.

4.17. Utilizando as *list comprehension* crie uma nova lista a partir de uma lista de inteiros que contenha os inteiros da lista original que são múltiplos de 3.

4.18. Utilizando as *list comprehension* crie uma nova lista a partir de uma lista de frutas ['ananas', 'banana', 'coco', 'figo'] que contenha as frutas que possuem a vogal 'a'.

4.19. Ordene uma lista de inteiros utilizando a função *sort()*.

4.20. Imprima uma lista de inteiros pela ordem inversa utilizando a função *reverse()*.

4.21. Utilize a função *filter* para filtrar de uma lista de carateres as vogais. Exemplo: Entrada: ['a','b','c','d'] -> Saída: ['a'].

4.22. Utilize a função *filter* para encontrar frutas que começam com a letra "m".

4.23. Utilize a Função *filter* com lambda para encontrar números ímpares.

4.24. Utilize a função *filter* em conjunto com uma função *lambda* para filtrar de uma lista de inteiros os elementos/valores que estão acima da média dos elementos/valores da lista de inteiros.

4.25. Utilize a função *map* para atribuir a todos os elementos de uma lista de inteiros o seu quadrado.

4.2 Tuplos

4.1. Crie um tuplo com algumas cores e imprima cada cor.

4.2. Crie um tuplo de coordenadas (x, y) e, em seguida, desempacote-a para mostrar os valores de x e y.

4.3. Crie um tuplo de números e encontre o maior e o menor valor.

4.4. Conte quantas vezes um elemento específico ocorre em um tuplo.

4.5. Una dois tuplos em um único tuplo.

4.6. Crie uma lista de números e converta-a em um tuplo.

4.7. Crie um tuplo com os números pares de 1 a 10.

4.3 Conjuntos

4.1. Crie um conjunto vazio.

4.2. Crie um conjunto com os elementos: "maçã", "banana", "laranja".

4.3. Adicionar elementos a um conjunto: ex. "uva".

4.4. Remova um elemento do conjunto

4.5. Una dois conjuntos.

4.6. Intersete dois ocnjuntos

4.4 Dicionários

4.1. Crie um dicionário vazio.

4.2. Adicione elementos a um dicionário.

4.3. Aceda valores de um dicionário.

4.4. Modifique um valor de um dicionário.

4.5. Remova um elemento de um dicionário.

4.6. Verifique a existência de uma chave num dicionário.

4.7. Percorra um dicionário utilizabdo um ciclo.

4.8. Obtenha uma lista de todas as chaves e valores existentes num dicionário.

4.9. Crie um dicionário com um dicionário dentro.

4.10. Verifique se um dicionário está vazio.

4.5 Miscelânea

4.1. Crie uma lista de estudantes, em que cada estudante, é um dicionário como os seguintes campos: nome, idade e notas, sendo as notas por sua vez uma lista com três notas.

4.2. Calcule a média das notas.

4.3. Apresente os nomes dos estudantes aprovados, ou seja, que tem uma média de notas superior a 9.5.

Capítulo 5

Exercícios sobre Tratamento de Exceções

5.1. Escreva um algoritmo que peça ao utilizador dois números e realize a divisão do primeiro número pelo segundo número. Certifique-se de tratar a exceção da divisão por zero.

5.2. Dada uma lista, solicite ao utilizador que insira um índice e, em seguida, imprima o elemento da lista nesse índice. Certifique-se de tratar exceções de índices fora dos limites.

5.3. Peça ao utilizador para inserir um número inteiro, trate exceções caso a entrada não seja um número inteiro.

5.4. Realize uma divisão por zero e use um bloco *finally* para garantir que uma parte do código seja executada, independentemente de ocorrer ou não uma exceção.

Capítulo 6

Exercícios sobre Ficheiros e Diretórios

6.1. Ler um ficheiro de texto. Crie um algoritmo que leia um ficheiro de texto chamado "exemplo.txt"e imprima o seu conteúdo.

6.2. Escrever num ficheiro de texto. Crie um algoritmo que permita ao utilizador inserir linhas de texto em um ficheiro chamado "notas.txt".

6.3. Listar os ficheiros existentes numa Pasta/Diretório. Liste todos os ficheiros existentes num diretório específico.

6.4. Copiar um ficheiro. Copie o conteúdo de um ficheiro chamado "origem.txt"para um novo ficheiro chamado "destino.txt".

6.5. Contar palavras de um ficheiro. Leia um ficheiro de texto e conte quantas palavras ele contém.

6.6. Apagar um ficheiro. Apague ficheiro chamado "ficheiro_a_apagar.txt".

6.7. Renomear um ficheiro. Renomeie um ficheiro chamado "antigo.txt"para "novo.txt".

6.8. Criar uma Pasta/Diretório. Crie um diretório chamado "novo_diretorio".

6.9. Remover uma Pasta/Diretório. Remova um diretório chamado "diretorio_a_apagar"e todo o seu conteúdo.

6.10. Verificar se uma Paste/Diretório existe. Verifique se um diretório chamado "meu_diretorio"existe.

6.11. Para guardar uma lista em um ficheiro binário em Python, podemos usar a biblioteca pickle. O módulo pickle permite serializar (converter em um formato binário) objetos Python, como listas, dicionários e objetos personalizados, para que possam ser armazenados em ficheiros e posteriormente recuperados. Aqui está como você pode fazer isso:

```
import pickle
# Sua lista de exemplo
minha_lista = [1, 2, 3, 4, 5]
# Nome do ficheiro para guardar a lista
nome_ficheiro = "minha_lista.bin"
# Abrir o ficheiro para escrita binaria (wb)
with open(nome_ficheiro, "wb") as ficheiro:
    # Use pickle.dump() para serializar a lista e grava-la no ficheiro
    pickle.dump(minha_lista, ficheiro)
print("Lista salva com sucesso em", nome_ficheiro)
```

Neste exemplo, a lista minha_lista é serializada e salva no ficheiro binário "minha_lista.bin". Agora, podemos ler e recuperar a lista a partir desse ficheiro binário da seguinte maneira:

```
import pickle
# Nome do ficheiro onde a lista esta armazenada
nome_ficheiro = "minha_lista.bin"
# Abrir o ficheiro para leitura binaria (rb)
with open(nome_ficheiro, "rb") as ficheiro:
    # Use pickle.load() para desserializar a lista do ficheiro
    lista_recuperada = pickle.load(ficheiro)
print("Lista recuperada:", lista_recuperada)
```

Certifique-se de que o ficheiro binário existe antes de tentar recuperar a lista. O módulo pickle permite salvar e recuperar objetos Python complexos, tornando-o útil para a persistência de dados. Certifique-se de que o ficheiro não seja manipulado por fontes não confiáveis, pois o pickle pode executar código malicioso incorporado em objetos serializados.

Capítulo 7

Exercícios sobre Programação Orientada a Objetos

7.1. Defina uma classe chamada Carro com atributos como marca, modelo, ano, cor e um método info que imprime as informações do carro.

7.2. Instanciação de Objetos. Crie dois objetos do tipo Carro e imprima suas informações.

7.3. Herança. Crie uma classe derivada chamada CarroDesportivo que herda de Carro e inclui um atributo adicional velocidade_maxima. Adicione um método que imprime a informação do carro desportivo, incluindo a velocidade máxima.

7.4. Polimorfismo. Crie uma função chamada exibir_informacoes que aceita um objeto Carro e imprime suas informações usando o método info.

7.5. Composição. Defina uma classe Motor que represente um motor de um carro com atributos como cavalos, cilindradas e um método ligar.

7.6. Composição de Carro. Modifique a classe Carro para incluir um objeto Motor. Adicione um método iniciar que liga o motor.

7.7. Polimorfismo e Composição. Crie uma função ligar_carro que aceita um objeto Carro (composição) e chama o método iniciar do carro.