Instituto Federal de Brasília
Campus Taguatinga

Lista de Exercícios 1 (09/14/2025)

Ciência de Dados - 2025/1

Dr. Prof. Raimundo C. S. Vasconcelos
Dr. Prof. Fabiano C. Fernandes

Tales Lima de Oliveira
tales.oliveira@estudante.ifb.edu.br

Disponibilidade do Código

Os `códigos-fonte` e `scripts` desenvolvidos para esta atividade estão disponíveis publicamente e podem ser acessados através do GitHub e do Google Colab.

1. Desenvolva um programa que selecione um número aleatório entre 0 e 100. O usuário deverá tentar adivinhar esse número. Ao final, o programa exibirá a sequência de palpites do usuário e o total de tentativas feitas até acertar o número correto.

Código 1:

```python
import random

def guessing_game():
    secret_number = random.randint(0, 100)
    guesses = []
    print("Tente adivinhar o número entre 0 e 100!")

    while True:
        try:
            guess = int(input("Digite seu palpite: "))
            guesses.append(guess)

            if guess < secret_number:
                print("Muito baixo! Tente novamente.")
            elif guess > secret_number:
                print("Muito alto! Tente novamente.")
            else:
                print("Parabéns! Você acertou!")
                break
        except ValueError:
            print("Por favor, digite um número inteiro válido.")

    print("\n--- Resultado ---")
    print("Número de tentativas:", len(guesses))
    print("Seus palpites foram:", guesses)

guessing_game()
```

2. Você está organizando uma lista de alunos por turmas em uma escola. Cada turma contém vários alunos, e cada aluno é representado por um dicionário com informações sobre seu nome e idade.

2.1. Criar as turmas: Crie duas turmas, cada uma contendo uma lista de dicionários representando os alunos. Cada dicionário deve ter as chaves `nome` e `idade`.

Código 2.1: a)

```
class1 = [
    {"name": "Ana", "age": 14},
    {"name": "Bruno", "age": 15},
    {"name": "Carla", "age": 14}
]

class2 = [
    {"name": "Daniel", "age": 13},
    {"name": "Eduarda", "age": 14},
    {"name": "Felipe", "age": 13}
]
```

2.2. Adicionar um novo aluno: Adicione um novo aluno em uma das turmas.

Código 2.2: b)

```
new_student = {"name": "Gabriel", "age": 15}
class1.append(new_student)
```

2.3. Remover um aluno: Remova um aluno de uma das turmas.

Código 2.3: c)

```
student_to_remove = "Carla"
class1 = [student for student in class1 if student["name"] != student_to_remove]
```

2.4. Exibir os alunos de uma turma.

Código 2.4: d)

```
print("Alunos da turma 1:")
for student in class1:
    print(f"Nome: {student['name']}, Idade: {student['age']}")
```

3. **Crie uma função que receba uma lista de palavras e retorne o número de palavras que tenham um tamanho maior do que 5.**

Código 3:

```python
def count_long_words(word_list):
    return len([word for word in word_list if len(word) > 5])

words = ["banana", "sol", "computador", "livro", "água", "amizade"]
print(count_long_words(words))
```

4. **Você deve construir uma pirâmide ao estilo Mario. O tamanho da pirâmide será decidida pelo usuário.**

Código 4:

```python
def mario_pyramid(height):
    for i in range(1, height + 1):
        spaces = ' ' * (height - i)
        blocks = '#' * i
        print(spaces + blocks)

user_height = int(input("Tamanho: "))
mario_pyramid(user_height)
```

5. **Usando a biblioteca Numpy: Faça uma função que receba um array como entrada, remova o primeiro e o último elementos do array e retorne um novo array contendo os elementos restantes.**

Código 5:

```python
import numpy as np

def remove_first_and_last(arr):
    if arr.size <= 2:
        return np.array([])  # Retorna array vazio se tiver 2 ou menos
        ↪ elementos
    return arr[1:-1]

array = np.array([10, 20, 30, 40, 50])
new_array = remove_first_and_last(array)
print("Novo array:", new_array)
```

6. Utilizando a biblioteca Numpy: Gere um array A contendo todos os números ímpares entre 0 e 100. A partir de A, crie outro array que contenha a soma cumulativa dos elementos de A que são divisíveis por 3 e 5 simultaneamente.

Código 6:

```python
import numpy as np

# Passo 1: Gerar array A com números ímpares de 0 a 100
A = np.arange(1, 100, 2)

# Passo 2: Filtrar elementos divisíveis por 3 e 5 (ou seja, por 15)
divisible_by_15 = A[(A % 15 == 0)]

# Passo 3: Calcular a soma cumulativa
cumulative_sum = np.cumsum(divisible_by_15)

# Exibir os resultados
print("Array A (ímpares):", A)
print("Divisíveis por 3 e 5:", divisible_by_15)
print("Soma cumulativa:", cumulative_sum)
```

7. Faça uma função que dados dois vetores (valores e pesos), calcule a média ponderada de valores com seus respectivos pesos. Observação: não é permitido utilizar a função np.average do NumPy

---

Código 7:

```python
import numpy as np

def weighted_average(values, weights):
    values = np.array(values)
    weights = np.array(weights)

    if values.size != weights.size:
        raise ValueError("Os vetores de valores e pesos devem ter o mesmo
        ↪  tamanho.")

    total_weight = np.sum(weights)
    weighted_sum = np.sum(values * weights)

    return weighted_sum / total_weight

valores = [7.5, 8.0, 6.0]
pesos = [2, 3, 1]

media = weighted_average(valores, pesos)
print(f"Média ponderada: {media:.2f}")
```

8. Utilizando a biblioteca Numpy: Você é um professor e está organizando as notas de seus alunos. Para tanto, você precisa realizar algumas tarefas, como:

8.1. Determinar a média das notas

Código 8.1: a)

```python
import numpy as np

# Lista de nomes dos alunos
students = np.array(["Ana", "Bruno", "Carla", "Daniel", "Eduarda", "Felipe",
↪  "Gabriel"])
# Notas correspondentes
grades = np.array([7.5, 5.0, 8.0, 6.0, 4.5, 9.0, 3.5])

average_grade = np.mean(grades)
print(f"Média da turma: {average_grade:.2f}")
```

8.2. Determinar a maior e a menor notas e os alunos que as tiraram

Código 8.2: b)

```python
max_grade = np.max(grades)
min_grade = np.min(grades)

students_max = students[grades == max_grade]
students_min = students[grades == min_grade]

print(f"Maior nota: {max_grade} - Aluno(s): {', '.join(students_max)}")
print(f"Menor nota: {min_grade} - Aluno(s): {', '.join(students_min)}")
```

8.3. Determinar as notas dos alunos aprovados e reprovados

Código 8.3: c)

```python
approved = students[grades >= 5.0]
approved_grades = grades[grades >= 5.0]

reproved = students[grades < 5.0]
reproved_grades = grades[grades < 5.0]

print("Alunos aprovados:")
for name, grade in zip(approved, approved_grades):
    print(f"{name}: {grade}")

print("\nAlunos reprovados:")
for name, grade in zip(reproved, reproved_grades):
    print(f"{name}: {grade}")
```

9. Sem utilizar a biblioteca Numpy: Faça uma função que receba um array como entrada, remova o primeiro e o último elementos do array e retorne um novo array contendo os elementos restantes.

Código 9:

```python
def remove_first_and_last(arr):
    if len(arr) <= 2:
        return []  # Retorna lista vazia se tiver 2 ou menos elementos
    return arr[1:-1]

lista = [10, 20, 30, 40, 50]
nova_lista = remove_first_and_last(lista)
print("Nova lista:", nova_lista)
```

10. Considere um array bidimensional no qual cada vetor de uma dimensão possui 3 pontuações que um jogador recebeu em 3 provas distintas. Faça uma função que dado esse array, retorne a prova com maior soma de pontuações.

Código 10:

```python
def best_exam(scores):
    if not scores or not scores[0]:
        return None  # Lista vazia ou mal formatada

    # Supondo que todas as linhas têm o mesmo tamanho
    num_exams = len(scores[0])
    total_by_exam = [0] * num_exams

    # Soma das colunas
    for row in scores:
        for i in range(num_exams):
            total_by_exam[i] += row[i]

    # Encontrar o índice da prova com maior soma
    max_index = total_by_exam.index(max(total_by_exam))
    return max_index  # ou retornar "Prova 1", "Prova 2", etc.

# Cada linha: jogador -> [prova1, prova2, prova3]
pontuacoes = [
    [10, 8, 6],
    [7, 9, 5],
    [8, 6, 9]
]

print(f"A prova com maior soma de pontuações é a Prova {best_exam(pontuacoes) + 1}")
```