Lista de exercícios 1

Questão 1. Uma das maneiras de se calcular o valor do número de Euler é por meio de:

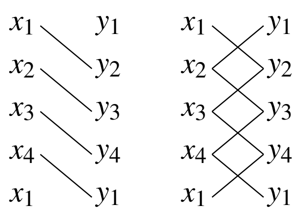
$$e = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n.$$

Implemente uma função chamada `euler` em Python que calcule um valor aproximado para e.

Questão 2. Elabore uma função em Python que calcula a derivada de um polinômio P de grau $< n$. Sua função deve se chamar `derivada` e ter como argumento de entrada um arranjo da NumPy contendo os coeficientes de P, armazenados segundo a ordem decrescente dos graus dos monômios correspondentes. Como resultado, ela deve retornar um arranjo contendo os coeficientes de P' segundo a mesma ordem descrita anteriormente. Verifique sua implementação com o polinômio:

$$P(x) = x^{20} - 210x^{19} + 20615x^{18} - 1256850x^{17} + 53327946x^{16}.$$

Questão 3 (Hill, 2016)**.** O método de Gauss para o cálculo da área de um polígono simples funciona do seguinte modo. Considere um polígono simples com n vértices (x_1, y_1), (x_2, y_2), …, (x_N, y_N), $N \geq 3$, ordenados no sentido anti-horário. Armazene as coordenadas desses pontos em um arranjo $(N+1)\times 2$, repetindo na última linha as coordenadas do primeiro vértice. Agora, (a) multiplique cada coordenada x das primeiras N linhas pela coordenada y da linha imediatamente abaixo e tome a soma $S_1 = x_1y_2 + x_2y_3 + \cdots + x_Ny_1$. Depois, (b) multiplique cada coordenada y das N primeiras linhas pela coordenada x da linha imediatamente abaixo e tome a soma $S_2 = y_1x_2 + y_2x_3 + \cdots + y_Nx_1$. Então, a área do polígono será $\frac{1}{2}|S_1 - S_2|$. Este algoritmo está ilustrado na figura a seguir.



(a) (b)

Implemente esse algoritmo como uma função em Python que recebe dois arranjos `x` e `y` da NumPy como entrada e retorna a área do polígono.

Verifique sua implementação no polígono de coordenadas:

```
# Abcissas
x = np.array([0.5, 8.0, 19.0, 26.0, 15.0, 4.0])

# Ordenadas
y = np.array([0.5, 4.0, 11.0, 25.0, 27.0, 9.0])
```

Questão 4 (Adaptado de Maratona de Programação da SBC 2013)**.** Todos devem conhecer o jogo Zerinho ou Um (em algumas regiões também conhecido como Dois ou Um), utilizado para determinar um ganhador entre três ou mais jogadores. Para quem não conhece, o jogo funciona da seguinte maneira. Cada jogador escolhe um valor entre zero ou um; a um comando (geralmente um dos competidores anuncia em voz alta "Zerinho ou... Um!"), todos os participantes mostram o valor escolhido, utilizando uma das mãos: se o valor escolhido foi um, o competidor mostra o dedo indicador estendido; se o valor escolhido foi zero, mostra a mão com todos os dedos fechados. O ganhador é aquele que tiver escolhido um valor diferente de todos os outros; se não há um jogador com valor diferente de todos os outros (por exemplo todos os jogadores escolhem zero, ou um grupo de jogadores escolhe zero e outro grupo escolhe um), não há ganhador. Você deve escrever uma função em Python que determina se há um ganhador, e nesse caso determina qual o número correspondente ao ganhador.

A entrada de sua função deve consistir de um arranjo `x` da `numpy` contendo as escolhas de $n > 1$ jogadores. Sua função deve retornar o par `(True, i)` para indicar quando o i-ésimo jogador venceu a partida, ou `None`, caso contrário. Você deve utilizar o seguinte cabeçalho:

```
def zeroouum(x):
```

Verifique sua implementação nas entradas:

```
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
[1, 0, 1, 0, 1]
[0, 0, 1, 0, 0, 0]
[1, 1, 1, 1, 0, 1]
```

Questão 5. A sequência de Fibonacci é uma velha conhecidade na matemática. Originalmente usada para modelar a dinâmica populacional de coelhos em 1202, ela hoje encontra aplicações nos mais diversos campos do conhecimento, como a computação, botânica, arquitetura, etc.

Podemos definir a sequência de Fibonacci $(F_n)_{n=0}^{\infty}$ usando recursividade. Com efeito, definimos $F_0 = 0$ e $F_1 = 1$, os quais chamamos de casos base. Agora, estabelecemos a seguinte relação recursiva:

$$F_n = F_{n-1} + F_{n-2}, \text{ para } n = 2, 3, 4, \ldots$$

Um algoritmo matricial bastante elegante que nos permite usar os arranjos da NumPy para calcular os termos dessa sequência se baseia na seguinte equação:

$$\begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} F_{n-1} + F_{n-2} \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_{n-1} \\ F_{n-2} \end{bmatrix}.$$

Utilize a equação acima para desenvolver e implementar em Python o algoritmo matricial em uma função denominada `fibmat`. Esta função deve retornar o valor de F_n. Utilize-a para construir uma tabela semelhante à publicada em `https://planetmath.org/listoffibonaccinumbers`.