



LET'S CODE

# Lista de exercícios - Python

## Questão 1.

Faça um programa que peça ao usuário um número e imprima todos os números de um até o número que o usuário informar.

💡 Exemplo:

Se o usuário informar o número 5, seu programa deverá imprimir: 1 2 3 4 5.

## Questão 2.

Crie um programa que leia um valor qualquer e apresente uma mensagem dizendo em qual dos seguintes intervalos  $([0,25], (25,50], (50,75], (75,100])$  este valor se encontra. Caso o valor não esteja em nenhum destes intervalos, deverá ser impressa a mensagem "Fora de intervalo". Veja alguns exemplo abaixo:

💡 Entrada: 25.01 | Saída: (25,50]  
Entrada: 25.00 | Saída: [0,25]  
Entrada: 100.00 | Saída: (75,100]  
Entrada: -25.02 | Saída: Fora de intervalo

📌 Lembrando que o `[` ou `]` representa que o valor está contido no intervalo, enquanto o `(` ou `)` representa que o valor associado não está contido no intervalo. Em outras palavras, `(75, 100]` representa o intervalo que vai de maior que 75 (não incluindo o 75) até menor ou igual 100.

## Questão 3.

Crie uma função que recebe o valor do raio de um círculo como parâmetro e retorna o valor da área desse círculo. Lembrando que a área de círculo é dada pela equação:  $A = \pi r^2$ .

💡 Dica: Para utilizar um valor mais preciso do pi ( $\pi$ ), você pode importar a biblioteca `math`, e utilizar o `math.pi`, como ilustrado no código abaixo:

```
import math  
  
print(math.pi)
```

#### Questão 4.

Faça um programa que peça 2 números inteiros e um número real, calcule e mostre:

- a) o produto entre o dobro do primeiro e a metade do segundo.
- b) a soma entre o triplo do primeiro e o terceiro.
- c) o terceiro elevado ao cubo.

#### Questão 5.

Vamos fazer um programa para verificar quem é o assassino de um crime. Para descobrir o assassino, a polícia faz um pequeno questionário com 5 perguntas onde a resposta só pode ser sim ou não:

1. Mora perto da vítima?
2. Já trabalhou com a vítima?
3. Telefonou para a vítima?
4. Esteve no local do crime?
5. Devia para a vítima?

Cada resposta sim dá um ponto para o suspeito. A polícia considera que os suspeitos com 5 pontos são os assassinos, com 4 a 3 pontos são cúmplices e 2 pontos são apenas suspeitos, necessitando de outras investigações. Valores abaixo de 2 são liberados.

No seu programa, você deve fazer essas perguntas e, de acordo com as respostas do usuário, você vai informar como a polícia o considera.

#### Questão 6.

Faça uma função que recebe uma lista de números e retorna a soma dos elementos dessa lista.

#### Questão 7.

Faça um programa que leia as coordenadas de 2 (dois) pontos em um plano cartesiano 2D: a coordenada x do primeiro ponto ( $x_1$ ), a coordenada y do primeiro ponto ( $y_1$ ), a coordenada x do segundo ponto ( $x_2$ ) e a coordenada y do segundo ponto ( $y_2$ ). Em seguida, calcule a distância euclidiana entre os pontos, utilizando a equação abaixo:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

#### Questão 8.

Calcule a soma de mil termos dos inversos dos fatoriais:  $1/(1!) + 1/(2!) + 1/(3!) + 1/(4!) + \dots$

Dica: Use três variáveis:

- um contador;
- uma variável para soma;
- e uma variável para os termos.

Lembre-se de que  $4! = 4 * 3 * 2 * 1$ , que também é igual a  $4 * 3!$ .

### Questão 9.

Crie (manualmente ou sorteando os números) uma lista de 10 números e imprima:








1. uma lista com os 4 primeiros números;
2. uma lista com os 5 últimos números;
3. uma lista contendo apenas os elementos das posições pares;
4. uma lista contendo apenas os elementos das posições ímpares;
5. a lista inversa da lista sorteada (isto é, uma lista que começa com o último elemento da lista sorteada e termina com o primeiro);
6. uma lista inversa dos 5 primeiros números;
7. uma lista inversa dos 5 últimos números.

### Questão 10.

Na música, uma nota tem um tom (sua frequência, resultando em quão grave ou agudo é o som) e uma duração (por quanto tempo a nota soa). Neste problema, estamos interessados apenas na duração das notas.


Marcos está dando os primeiros passos para ser um compositor de jingles. Ele está tendo alguns problemas, mas ao menos ele está criando melodias agradáveis e ritmos atrativos. Um jingle é dividido em uma sequência de compassos, e um compasso é formado de uma série de notas.

A duração de uma nota é indicada pela sua forma. Neste problema, iremos utilizar letras maiúsculas para indicar a duração de uma nota. A seguinte tabela lista todas as notas disponíveis:

Notas							
Identificador	W	H	Q	E	S	T	X
Duração	1	1/2	1/4	1/8	1/16	1/32	1/64

A duração de um compasso é a soma da duração de suas notas. Nos jingles de Marcos, cada compasso tem a mesma duração. Como Marcos é apenas um iniciante, seu famoso professor Johann Sebastian III o ensinou que a duração de um compasso deve ser sempre 1.

Por exemplo, Marcos escreveu uma composição contendo cinco compassos, dentre os quais quatro possuem a duração correta e um está errado. No exemplo abaixo, cada compasso é delimitado com barras e cada nota é representada como na tabela acima.

 /HH/QQQQ/XXXTXTEQH/W/HW/

Marcos gosta de computadores assim como de música. Ele quer que você escreva um programa que determine, para cada uma de suas composições, quantos compassos possuem a duração correta e quais são os compassos com duração incorreta.

Exemplo de Entrada	Exemplo de Saída
/HH/QQQQ/XXXTXTEQH/W/HW/	Qtd. de Corretos: 4 Incorretos: HW
/W/W/SQHES/	Qtd. de Corretos: 3
/WE/TEX/THES/	Qtd. de Corretos: 0 Incorretos: [WE, TEX, THES]

## Desafio final 🚀 - Python

Fala, pessoal! Sejam muito bem-vindos ao Desafio de Python! Se você chegou aqui, então já deve ter acompanhado as aulas do nosso curso, certo? Bom, agora, é hora de colocar seus novos conhecimentos em prática! Vamos nessa?! 🚀 😊

Antes de tudo, é válido destacar que por se tratar de um desafio, é essencial que você troque uma ideia com outras pessoas, caso sinta dificuldades. Para isso, conte com a [comunidade](#)! Ah, e claro, você também pode ficar de olho nas dúvidas de outras pessoas para ajudá-las por lá. Tudo isso vai te fazer avançar cada vez mais rápido rumo ao seu objetivo! 🎯

### Conhecendo o seu desafio

Você já deve ter jogado o Jogo da Forca, certo? O que você acha de desenvolver o seu próprio Jogo da Forca em Python? Esse será o seu desafio!

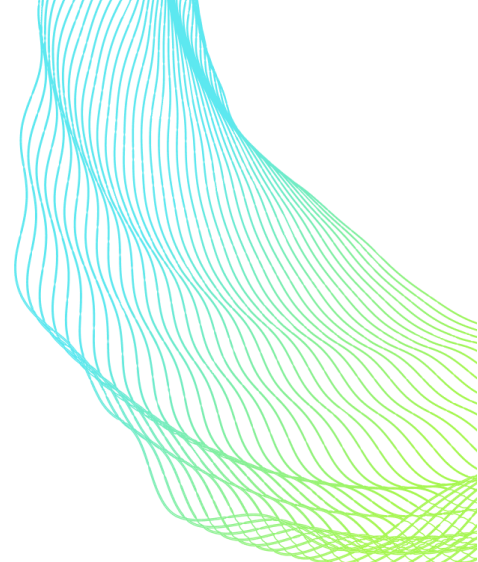
Em resumo, ao iniciar o jogo, o jogador receberá uma palavra "codificada", de forma que ele saiba apenas o número de letras que a compõem. Com base nisso, o jogador deve escolher uma letra que acredita fazer parte daquela palavra. O jogo acaba quando o jogador erra a letra pela sexta vez ou quando acerta todas as letras da palavra.

### Como desenvolver o seu Jogo da Forca?


Para te ajudar com isso, vamos te passar algumas diretrizes para que você tenha uma noção clara de como o jogo deve funcionar e de quais etapas você deve seguir para atingir esse resultado. Vamos lá!

1. No início do código, você pode solicitar o nome do jogador. Assim, você pode imprimir uma mensagem de boas-vindas. Outra opção, caso queira incluir, é imprimir uma mensagem explicando como o jogo funciona.
2. Você precisa definir uma palavra para o jogador descobrir, certo? Para isso, você pode ter uma lista com várias palavras e, no início do programa, sortear uma delas.
3. Escolhida uma palavra para o jogador descobrir, você já pode mostrar para ele quantas letras a palavra tem. Vai ser importante ter uma outra variável que consiste na palavra que o jogador está tentando acertar. Você pode utilizar uma lista, que inicia com vários `_`, por exemplo; sendo o número de `_` igual ao número de caracteres da palavra que ele precisa descobrir.



- 
4. A partir daí, o jogo começa. Ou seja, você irá pedir que o usuário informe uma letra repetidas vezes, até que ele erre 6 vezes ou acerte todas as letras da palavra (já consegue imaginar como vai ser essa estrutura de repetição?).
  5. Lembre-se que, ao término dessa repetição, você deve mostrar que o usuário perdeu, caso ele tenha errado 6 vezes, ou que ele acertou a palavra, caso ele a tenha completado. Além disso, é importante que você informe qual era a palavra a ser descoberta.

Pronto! Essas são algumas diretrizes para que você possa se guiar no desenvolvimento do seu desafio.

 **Importante:** apesar das diretrizes elencadas acima, você pode utilizar sua criatividade para deixar esse jogo com a sua cara. Fique à vontade para personalizar o funcionamento dele, de acordo com o que você acredita que vai te ajudar a extrair o máximo de conhecimento dessa experiência.

## Para inspirar

Para te inspirar de maneira você possa ter uma referência ao longo do desenvolvimento desta atividade,, assista a este vídeo: [https://youtu.be/Oc\\_HCXAYdxq](https://youtu.be/Oc_HCXAYdxq)

## Dicas

Você deve ter notado no vídeo acima que existem alguns recursos do Python que estão sendo utilizados, mas que você pode não ter visto durante o curso. Não se preocupe; no primeiro momento, você não precisa se preocupar em utilizá-los ou até mesmo em fazer da mesma forma que está no vídeo. Lembre-se que é apenas uma referência.

Porém, tendo certeza que você vai querer ir além, já preparamos algumas dicas que podem te ajudar:

### Qual comando eu poderia utilizar no Python para limpar o console?

Para isso, você deve utilizar um comando do sistema, o que é possível de ser feito por meio do módulo `os` do Python. Nesse módulo, existe uma função chamada `system` que te permite utilizar comandos do sistemas (comandos que você utilizaria no cmd, prompt de comando ou Terminal). Sendo assim, veja o exemplo abaixo:

```
import os
```

```
# o comando abaixo vai limpar o seu console, caso você esteja utilizando o Windows os.system('cls')
```

```
# o comando abaixo vai limpar o seu console, caso você esteja utilizando o Linux/Mac os.system('clear')
```

## Como eu posso fazer para "congelar" a execução do meu programa por um tempo específico?

Você pode fazer isso utilizando a função `sleep` do módulo `time`. Esse módulo já vem instalado com o Python, portanto, basta você fazer a importação desse módulo ou apenas da função `sleep` (como no exemplo abaixo).

```
from time import sleep
```

```
sleep(3)
```

```
print('Essa mensagem só aparece após 3 segundos...')
```

Observando o exemplo acima, você pode notar que a função `sleep` deve receber um parâmetro, que é o tempo em segundos que o programa irá "dormir". Ou seja, "segurar" o seu fluxo de execução. Portanto, o `print` que vem logo abaixo da função `sleep` será executado apenas 3 segundos após a chamada dessa função `sleep`.