Entrada de valores **2**
Conversão de valores 2
Tipos de Conversão 2
Dividir uma entrada com vários valores 2
Com valores inteiros 2
Com valores decimais 3
Listas (vetores ou arrays) **3**
Declaração de vetor simples 3
Declaração de matriz 3
Adicionar valores 4
No vetor 4
Na variável 4
Na Matriz 4
Em uma string 4
Pegar intervalos 4
De uma posição até a última 4
De uma posição a outra 5
A partir de uma posição final 5
Deletar valores 5
Verificar se um valor já existe 5
Saber o tamanho do vetor 6
Strings **6**
Trocar texto ou caractere específico 6
Pegar um intervalo de uma string 6
Remover caracteres de uma string 6
Condicionais **7**
Loopings **8**
while 8
for 8
Organizar valores **8**
Ordem Crescente 8
Ordem decrescente 8
Exemplos: **9**
Corrida de Kart: 9
Números Primos: 10
O Bruxo e os Frascos 11
Funções extras usadas na última maratona **12**
Funções Matemáticas **17**
Logaritmo 17
Logaritmo natural 17
Logaritmo natural de número negativo 17

Calcular o logaritmo natural de uma lista de números 17
Raiz Quadrada 18
Método de aproximação: 18
Utilizando a função math.sqrt(): 18
Saída **18**
Delimitar casas decimais: 18
Normal 18
Percentual 18
Formato Brasileiro 18*

Entrada de valores

```
nome = input("Digite o seu nome: ")
print("Olá, " + nome + "!")

idade = input("Digite a sua idade: ")
idade = int(idade)  # Converte a entrada para um número inteiro
print("No próximo ano, você terá", idade + 1, "anos.")
```

Conversão de valores

```
entrada = int(input())
```

Tipos de Conversão

Conversão de Tipo Explícita:

int(x): Converte x para um inteiro.
float(x): Converte x para um número de ponto flutuante.
str(x): Converte x para uma string.
list(x): Converte x para uma lista.
tuple(x): Converte x para uma tupla.
set(x): Converte x para um conjunto.
dict(x): Converte x para um dicionário.

Dividir uma entrada com vários valores

Com valores inteiros

```
entrada = input()
# dividindo a entrada em uma lista de strings (a cada espaço terá uma nova posição)
numeros_str = entrada.split()
# convertendo a lista de stings para uma lista de inteiros
```

```
numeros = [int(num) for num in numeros_str]
```

Com valores decimais

```
entrada = "10 20 30 3.14 5.5"

# Dividir a entrada em partes com base nos espaços em branco
partes = entrada.split()

# Converter as partes em números inteiros ou decimais
valores = []
for parte in partes:
    if "." in parte:
        valor_decimal = float(parte)
        valores.append(valor_decimal)
    else:
        valor_inteiro = int(parte)
        valores.append(valor_inteiro)

print(valores)
```

Listas (vetores ou arrays)

Declaração de vetor simples

```
participantes = []
voltas = []
```

Declaração de matriz

```
matriz = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

matriz_vazia = []

linha1 = [1, 2, 3]
linha2 = [4, 5, 6]
linha3 = [7, 8, 9]

matriz_vazia.append(linha1)
matriz_vazia.append(linha2)
matriz_vazia.append(linha3)
```

Adicionar valores

No vetor

```
vetor = [1, 2, 3]
vetor.append(4)
print(vetor)  # Isso imprimirá [1, 2, 3, 4]

vetor1 = [1, 2, 3]
vetor2 = [4, 5]
vetor1 = vetor1 + vetor2
print(vetor1)  # Isso imprimirá [1, 2, 3, 4, 5]

vetor1 = [1, 2, 3]
vetor2 = [4, 5]
vetor1.extend(vetor2)
print(vetor1)  # Isso imprimirá [1, 2, 3, 4, 5]

vetor = [1, 2, 3]
novo_elemento = 4
vetor = [x for x in vetor] + [novo_elemento]
print(vetor)  # Isso imprimirá [1, 2, 3, 4]
```

Na variável

```
qtd_pilotos = numeros[0]
qtd_voltas = numeros[1]
qtd_voltas_invalidas = numeros[2]
```

Na Matriz

```
melhor_volta.append(volta_nome)
melhor_volta[0].append(volta_tempo)
melhor_volta[0][1] = volta[1]
```

Em uma string

```
nome = input()
sobrenome = input()
nome_completo = nome + " " + sobrenome
print(nome_completo)
```

Pegar intervalos

De uma posição até a última

```
voltas[0:]
```

De uma posição a outra

voltas[1:9] # pega os valores da posição 1 até a posição (9-1) = 8

A partir de uma posição final

```
entrada = [24, 79, 10, 50]
print(entrada[-3:-1])
```

Deletar valores

```
vetor = [1, 2, 3, 4, 5]
del vetor[2]  # Remove o elemento de índice 2 (o valor 3)
print(vetor)  # Isso imprimirá [1, 2, 4, 5]

vetor = [1, 2, 3, 4, 3]
vetor.remove(3)  # Remove a primeira ocorrência do valor 3
print(vetor)  # Isso imprimirá [1, 2, 4, 3]

vetor = [1, 2, 3, 4, 5]
valor_removido = vetor.pop(2)  # Remove o elemento de índice 2 (o valor 3)
print(valor_removido)  # Isso imprimirá 3
print(vetor)  # Isso imprimirá [1, 2, 4, 5]

vetor = [1, 2, 3, 4, 5]
elemento_a_remover = 3
vetor = [x for x in vetor if x != elemento_a_remover]
print(vetor)  # Isso imprimirá [1, 2, 4, 5]
```

Verificar se um valor já existe

```
vetor = [1, 2, 3, 4, 5]
elemento = 3

if elemento in vetor:
    print(f"{elemento} existe na lista.")
else:
    print(f"{elemento} não existe na lista.")

 vetor = [1, 2, 3, 4, 3]
elemento = 3

if vetor.count(elemento) > 0: #count() retorna o número de elementos existentes no vetor
    print(f"{elemento} existe na lista.")
else:
    print(f"{elemento} não existe na lista.")
```

Saber o tamanho do vetor

```
vetor = [1, 2, 3, 4, 5]
tamanho = len(vetor)
print("O tamanho do vetor é:", tamanho)
```

Strings

Trocar texto ou caractere específico

```
texto = "Olá, mundo!"
novo_texto = texto.replace("m", "M")
print(novo_texto)  # Isso imprimirá "Olá, Mundo!"

import re

texto = "Olá, mundo!"
novo_texto = re.sub(r"m", "M", texto)
print(novo_texto)  # Isso imprimirá "Olá, Mundo!"
```

Pegar um intervalo de uma string

```
texto = "Esta é uma string de exemplo"
intervalo = texto[5:12]  # Pega os caracteres do índice 5 (inclusive) ao 11 (exclusivo)
print(intervalo)  # Isso imprimirá "é uma s"

texto = "Esta é uma string de exemplo"
inicio = texto[:4]  # Pega os caracteres do início até o índice 3
fim = texto[16:]  # Pega os caracteres a partir do índice 16 até o final
print(inicio)  # Isso imprimirá "Esta"
print(fim)  # Isso imprimirá "exemplo"

texto = "Esta é uma string de exemplo"
intervalo = texto[-6:-1]  # Pega os caracteres do sexto caractere a partir do final até o último
print(intervalo)  # Isso imprimirá "exemp"

texto = "Esta é uma string de exemplo"
intervalo = texto[::2]  # Pega todos os caracteres com um passo de 2
print(intervalo)  # Isso imprimirá "Esa m  tigd xepo"
```

Remover caracteres de uma string

```
texto = "Esta é uma string com espaços"
texto_sem_espacos = "".join(texto.split())
print(texto_sem_espacos)  # Isso imprimirá "Estaéumastringcomespaços"
```

```
texto = "Esta é uma string com números: 12345"
texto_sem_numeros = "".join(caracter for caracter in texto if not caracter.isdigit())
print(texto_sem_numeros)  # Isso imprimirá "Esta é uma string com números: "

texto = "12.345.67"
texto_sem_pontos = texto.replace(".", "")
print(texto_sem_pontos)  # Isso imprimirá "1234567"

import re

texto = "Esta é uma string com números: 12345"
texto_sem_numeros = re.sub(r'\d', '', texto)
print(texto_sem_numeros)  # Isso imprimirá "Esta é uma string com números: "
```

Condicionais

```
idade = 18

if idade < 18:
    print("Você é menor de idade.")
elif idade == 18:
    print("Você acabou de atingir a maioridade.")
else:
    print("Você é maior de idade.")

idade = 25
tem_carteira_de_motorista = True

if idade >= 18:
    if tem_carteira_de_motorista:
        print("Você pode dirigir legalmente.")
    else:
        print("Você tem idade para dirigir, mas não tem carteira de motorista.")
else:
    print("Você é menor de idade e não pode dirigir.")

tem_carteira_de_motorista = True
tem_carro = False

if tem_carteira_de_motorista and not tem_carro:
    print("Você pode dirigir, mas não tem um carro para dirigir.")
```

Loopings

while

```
contador = 0

while contador < 5:
    print("Contador é", contador)
    contador += 1

print("Loop concluído")
```

for

```
frutas = ["maçã", "banana", "laranja", "uva"]

# Itera sobre a lista de frutas usando um loop for
for fruta in frutas:
    print(fruta)
```

Organizar valores

Ordem Crescente

```
numeros = [5, 2, 9, 1, 5, 6]
numeros_ordenados = sorted(numeros)
print(numeros_ordenados)

numeros = [5, 2, 9, 1, 5, 6]
numeros.sort()
print(numeros)
```

Ordem decrescente

```
numeros = [5, 2, 9, 1, 5, 6]
numeros_decrescente = sorted(numeros, reverse=True)
print(numeros_decrescente)

numeros = [5, 2, 9, 1, 5, 6]
numeros.sort(reverse=True)
print(numeros)
```

Exemplos:

Corrida de Kart:

```
# entrada do usuário para os números
entrada = input()
# dividindo a entrada em uma lista de strings (a cada espaço terá uma nova posição
numeros_str = entrada.split()
# convertendo a lista de stings para uma lista de inteiros
numeros = [int(num) for num in numeros_str]
# posição 0 corresponde ao número de pilotos
qtd_pilotos = numeros[0]
# posição 1 corresponde ao número de voltas
qtd_voltas = numeros[1]
# posição 2 corresponde ao número de voltas inválidas
qtd_voltas_invalidas = numeros[2]
# declaração de listas (vetores)
participantes = []
voltas = []
melhor_volta = []
# Pegando os nomes dos corredores e colocando-os no vetor participantes
controle = 0
while controle < qtd_pilotos:
    participantes.append(input())
    controle += 1
# Pegando as informações das voltas e colocando-as no vetor voltas
controle = 0
while controle < qtd_voltas:
    voltas.append(input())
    controle += 1
# Retirando as voltas inválidas
voltas.pop(qtd_voltas_invalidas)
controle = 0
while controle < len(voltas):
    # Colocando as primeiras voltas de cada corredor
    if controle < qtd_pilotos:
        melhor_volta.append(voltas[controle])
    # Percorrendo a lista em que está armazenadas as melhores voltas
    else:
        controle2 = 0
        while controle2 < len(melhor_volta):
            # Se a sigla da volta for igual a da melhor volta
            if voltas[controle][0:3] == melhor_volta[controle2][0:3]:
                # Comparando os minutos da volta para ver se eles são menores do que a da
melhor volta
                if int(voltas[controle][4]) < int(melhor_volta[controle2][4]):
```

```
                # Deleto primeiro o valor da volta_melhor correspondente a condição acima, e
na linha seguinte
                # adiciono o valor da volta atual, já que ela foi melhor do que a armazenada em
melhor_volta
                del melhor_volta[controle2]
                melhor_volta.append(voltas[controle])
            # Comparando os segundos e os milissegundos, sendo estes casas decimais dos
segundos
            elif voltas[controle][4] == melhor_volta[controle2][4] and not float(
                voltas[controle][6:]) >= float(melhor_volta[controle2][6:]):
                # Deleto primeiro o valor da volta_melhor correspondente a condição acima, e
na linha seguinte
                # adiciono o valor da volta atual, já que ela foi melhor do que a armazenada em
melhor_volta
                del melhor_volta[controle2]
                melhor_volta.append(voltas[controle])
        controle2 += 1
    controle += 1
controle = 0
# Adiciono a melhor_volta a seu participante correspondente
while controle < len(participantes):
    controle2 = 0
    while controle2 < len(melhor_volta):
        if participantes[controle][0] + participantes[controle][1:3].upper() ==
melhor_volta[controle2][0:3]:
            participantes[controle] += " " + melhor_volta[controle2][4:]
        controle2 += 1
    controle += 1
controle = 1
# Reorganizando a lista em ordem crescente numérica
participantes = sorted(participantes, key=lambda x: int(''.join(filter(str.isdigit, x))))
# Imprimindo os valores
while controle <= qtd_pilotos:
    print(str(controle) + ' ' + participantes[controle - 1])
    controle += 1
```

Números Primos:

```
def primo(numerous):
    if numerous == 1:
        return False
    i = 2
    while i * i <= numerous:
        if numerous % i == 0:
            return False
        i += 1
    return True
```

```
x = int(input())  # transforma o valor digitado no console em inteiro e armazena na variável x
primos = []
for n in range(1, x + 1):  # para n entre 1 e x + 1
    if primo(n):
        primos.append(str(n))  # Transforma o inteiro n em string e adiciona ela ao vetor primos
print(' '.join(primos))  # join percorre todos os valores do vetor primos
```

O Bruxo e os Frascos

```
s = input()

cont = 0

vogais = ['A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u']

for c in s:
    if c in vogais:
        cont += 1

print('frasco', cont % 3)
```

PRIMOS usando algumas funções extras:

```
def primo(n):  # Define uma função chamada 'primo' que verifica se 'n' é primo.
    if n == 1:  # Se 'n' for igual a 1, retorna False, pois 1 não é primo.
        return False

    i = 2  # Inicializa uma variável 'i' com 2. Vamos começar a verificar divisores a partir de 2.

    while i * i <= n:  # Enquanto o quadrado de 'i' for menor ou igual a 'n', continue a
verificação de divisores.
        if n % i == 0:  # Se 'n' for divisível por 'i', retorna False, pois encontramos um divisor.
            return False
        i += 1  # Incrementa 'i' para verificar o próximo número como divisor.

    return True  # Se nenhum divisor for encontrado, retorna True, indicando que 'n' é primo.

x = int(input())  # Solicita ao usuário que insira um número inteiro e o armazena em 'x'.

primos = []  # Inicializa uma lista vazia chamada 'primos' para armazenar os números primos.

for n in range(1, x + 1):  # Para cada número 'n' no intervalo de 1 até 'x' (inclusive).
    if primo(n):  # Chama a função 'primo' para verificar se 'n' é primo.
        primos.append(str(n))  # Se 'n' for primo, adiciona a representação em string de 'n' à
lista 'primos'.
```

print(**' '.join(primos)**) # Imprime a lista de números primos como uma string com espaços entre eles.

Funções extras usadas na última maratona

```
# Utiliza .upper() para transformar o texto em maiúsculas
texto = "Olá, Mundo!"
texto_maiusculo = texto.upper()
print(texto_maiusculo)  # Saída: "OLÁ, MUNDO!"

# Utiliza .strip() para remover espaços em branco do início e do final
texto = "   Python é divertido!   "
texto_limpo = texto.strip()
print(texto_limpo)  # Saída: "Python é divertido!"


# Utiliza .split() para dividir uma frase em palavras com base no espaço em branco
frase = "Python é uma linguagem de programação"
palavras = frase.split(" ")
print(palavras)  # Saída: ['Python', 'é', 'uma', 'linguagem', 'de', 'programação']


# Utiliza enumerate() para obter índice e valor dos elementos em uma lista
lista = ["a", "b", "c", "d"]
for indice, valor in enumerate(lista):
    print(f"Índice: {indice}, Valor: {valor}")
# Saída:
# Índice: 0, Valor: a
# Índice: 1, Valor: b
# Índice: 2, Valor: c
# Índice: 3, Valor: d


# Utiliza .pop() para remover e retornar um elemento de uma lista com base no índice
lista = [1, 2, 3, 4, 5]
elemento_removido = lista.pop(2)
print(lista)  # Saída: [1, 2, 4, 5]
print(elemento_removido)  # Saída: 3


# Utiliza math.floor() para arredondar um número decimal para o maior número inteiro não maior que ele
import math
numero_decimal = 3.7
numero_arredondado = math.floor(numero_decimal)
print(numero_arredondado)  # Saída: 3
```

```
# Diferença entre sort() e sorted()
lista = [3, 1, 2, 4]

# Usando sort() para ordenar a lista original
lista.sort()
print(lista)  # Saída: [1, 2, 3, 4]

# Usando sorted() para criar uma nova lista ordenada
nova_lista = sorted(lista)
print(nova_lista)  # Saída: [1, 2, 3, 4]


# Utiliza itemgetter para extrair o segundo elemento de cada tupla em uma lista de tuplas
from operator import itemgetter

lista_de_tuplas = [(1, "a"), (3, "c"), (2, "b")]
extrair_segundo_elemento = itemgetter(1)
for tupla in lista_de_tuplas:
    print(extrair_segundo_elemento(tupla))
# Saída:
# a
# c
# b

# Utiliza filter() para filtrar números pares de uma lista
numeros = [1, 2, 3, 4, 5, 6, 7, 8]
pares = list(filter(lambda x: x % 2 == 0, numeros))
print(pares)  # Saída: [2, 4, 6, 8]

# Função recursiva para calcular o fatorial de um número
def fatorial(n):
    if n == 0:
        return 1
    else:
        return n * fatorial(n - 1)

resultado = fatorial(5)
print(resultado)  # Saída: 120

# Exemplo de programação dinâmica para calcular o n-ésimo número de Fibonacci
def fibonacci(n):
    fib = [0, 1]
    for i in range(2, n + 1):
        fib.append(fib[i - 1] + fib[i - 2])
    return fib[n]

resultado = fibonacci(5)
```

```
print(resultado)  # Saída: 5

from collections import Counter

# Usando Counter para contar elementos em uma lista
lista = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
contagem = Counter(lista)
print(contagem)  # Saída: Counter({4: 4, 3: 3, 2: 2, 1: 1})

# Ordenando uma lista de dicionários com base na idade
pessoas = [{'nome': 'Alice', 'idade': 30},
           {'nome': 'Bob', 'idade': 25},
           {'nome': 'Charlie', 'idade': 35}]
ordenado = sorted(pessoas, key=lambda x: x['idade'])
print(ordenado)  # Saída: [{'nome': 'Bob', 'idade': 25}, {'nome': 'Alice', 'idade': 30}, {'nome':
'Charlie', 'idade': 35}]

import math

# Usando math.gcd() para encontrar o maior divisor comum
a = 12
b = 18
mdc = math.gcd(a, b)
print(mdc)  # Saída: 6

# Implementação de busca binária para encontrar um elemento em uma lista ordenada
def busca_binaria(lista, alvo):
    esquerda, direita = 0, len(lista) - 1
    while esquerda <= direita:
        meio = (esquerda + direita) // 2
        if lista[meio] == alvo:
            return meio
        elif lista[meio] < alvo:
            esquerda = meio + 1
        else:
            direita = meio - 1
    return -1

lista_ordenada = [1, 2, 3, 4, 5, 6, 7]
indice = busca_binaria(lista_ordenada, 4)
print(indice)  # Saída: 3

# Implementação do algoritmo de ordenação rápida (Quicksort)
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
```

```
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quicksort(left) + middle + quicksort(right)

lista = [3, 6, 8, 10, 1, 2, 1]
ordenada = quicksort(lista)
print(ordenada)  # Saída: [1, 1, 2, 3, 6, 8, 10]
```

MAP()

```
# Utiliza map() para aplicar .upper() a cada elemento de uma lista de strings
palavras = ["python", "é", "divertido"]
palavras_maiusculas = list(map(str.upper, palavras))
print(palavras_maiusculas)  # Saída: ['PYTHON', 'É', 'DIVERTIDO']

# Utiliza map() para aplicar int() a cada elemento de uma lista de strings
strings_numericas = ["1", "2", "3", "4"]
numeros = list(map(int, strings_numericas))
print(numeros)  # Saída: [1, 2, 3, 4]

# Utiliza map() para elevar cada elemento de uma lista ao quadrado
numeros = [1, 2, 3, 4, 5]
quadrados = list(map(lambda x: x**2, numeros))
print(quadrados)  # Saída: [1, 4, 9, 16, 25]
```

QUESTÃO GRÁFICA

apenas para conhecimento do código

```
# Adjacency Matrix representation in Python

class Graph(object):
    # Inicializa a matriz de adjacência e o número de vértices
    def __init__(self, size):
        self.v = size  # Armazena o número de vértices no grafo
        self.adjMatrix = []  # Inicializa a matriz de adjacência como uma lista vazia
        for i in range(size):
            self.adjMatrix.append([0 for i in range(size)])  # Inicializa a matriz com zeros
        self.size = size

    # Adiciona arestas
    def add_edge(self, v1, v2):
        if v1 == v2:
            print("Mesmo vértice %d e %d" % (v1, v2))
        self.adjMatrix[v1][v2] = 1  # Define uma aresta entre o vértice v1 e v2

    # Restauro das arestas
    def remove_edge(self, v1, v2):
```

```
        if self.adjMatrix[v1][v2] == 0:
            print("Nenhuma aresta entre %d e %d" % (v1, v2))
            return
        self.adjMatrix[v1][v2] = 0  # Remove a aresta entre o vértice v1 e v2

    def __len__(self):
        return self.size

    # Função para realizar uma busca em profundidade (DFS) no grafo
    def dfs(self, start, visited, nodes):
        # Imprime o nó atual
        nodes.append(start)

        # Marca o nó atual como visitado
        visited[start] = True

        # Para cada nó do grafo
        for i in range(self.v):  # Utiliza o número de vértices (.v) para iterar sobre os vértices
            # Se algum nó for adjacente ao nó atual e ainda não foi visitado
            if (self.adjMatrix[start][i] == 1 and (not visited[i])):
                self.dfs(i, visited, nodes)

# Função principal que lê a entrada, cria o grafo e executa as operações de DFS
def main():
    nodes, relations, tests = input().split()
    nodes, relations, tests = [int(nodes), int(relations), int(tests)]
    names_list = []
    relations_list = []

    # Lê as relações e constrói uma lista de nomes

    g = Graph(nodes)  # Cria uma instância da classe Graph com o número de vértices
especificado

    for i in range(relations):
        # Adiciona arestas ao grafo com base nas relações
        g.add_edge(names_list.index(relations_list[i][2]),
                   names_list.index(relations_list[i][0]))
        g.add_edge(names_list.index(relations_list[i][2]),
                   names_list.index(relations_list[i][1]))

    for i in range(tests):
        a, b = input().split()

        # Realiza DFS para 'a'
        visited = [False] * nodes
        nodes_a = []
        g.dfs(names_list.index(a), visited, nodes_a)
```

```
        # Realiza DFS para 'b'
        visited = [False] * nodes
        nodes_b = []
        g.dfs(names_list.index(b), visited, nodes_b)

        # Verifica se há interseção entre os conjuntos de nós alcançados por 'a' e 'b'
        result = len(set(nodes_a).intersection(set(nodes_b))) > 0
        if result:
            print('verdadeiro')
        else:
            print('falso')

# Executa a função principal se este script for o ponto de entrada
if __name__ == '__main__':
    main()
```

Funções Matemáticas

Logaritmo

Logaritmo natural

```
import math

numero = 10
logaritmo_natural = math.log(numero)

print("O logaritmo natural de", numero, "é:", logaritmo_natural)
```

Logaritmo natural de número negativo

```
import math

numero = -5
logaritmo_natural = math.log(abs(numero))

print("O logaritmo natural de", numero, "é:", logaritmo_natural)
```

Calcular o logaritmo natural de uma lista de números

```
import math

numeros = [1, 2, 3, 4, 5]
logaritmos_naturais = [math.log(numero) for numero in numeros]
```

```
print("Os logaritmos naturais dos números", numeros, "são:", logaritmos_naturais)
```

Raiz Quadrada

Método de aproximação:

```
def calcular_raiz_aproximada(numero, margem_erro):
    aproximacao = numero
    while abs(aproximacao * aproximacao - numero) > margem_erro:
        aproximacao = (aproximacao + numero / aproximacao) / 2
    return aproximacao
```

Utilizando a função math.sqrt():

```
import math

numero = 16
raiz_quadrada = math.sqrt(numero) #ou raiz_quadrada = numero ** (1/2)
print(raiz_quadrada)
```

Saída

Delimitar casas decimais:

Normal

```
texto_lucro = f'R${lucro:.2f}'
print(f'O lucro foi de {texto_lucro}')
```

Percentual

```
margem = lucro / faturamento
print(f'A margem foi de {margem:.2%}')
```

Formato Brasileiro

```
texto_lucro = f'R${lucro:_.2f}'
texto_lucro = texto_lucro.replace('.',',').replace('_','.')
print(f'O lucro foi de {texto_lucro}')
```