1 - Variáveis e tipos de dados em Python

1.1 - Variáveis

```
'''
Em Python, variáveis são utilizadas para armazenar valores na memória do
computador. Para criar uma variável, você precisa usar um nome e um valor.
O nome da variável deve seguir as regras de nomenclatura do Python:

Deve começar com uma letra ou sublinhado (_).
Pode conter letras, números e sublinhados.
Não pode ser uma palavra-chave reservada do Python.
Para atribuir um valor a uma variável, utilize o operador de atribuição (=).
Exemplo:
'''
nome = 'Douglas'
print(nome)
print(type(nome))
idade = 48
print(idade)
print(type(idade))
altura = 1.85
print(altura)
print(type(altura))
```

```
Douglas
<class 'str'>
48
<class 'int'>
1.85
<class 'float'>
```

1.2 - Tipos de dados

```
'''
Tipos de Dados:

Cada variável em Python possui um tipo de dado, que define o tipo de valor que
ela pode armazenar. Os tipos de dados mais comuns em Python são:

Inteiro (int): Números inteiros, como 1, 2, 3, etc.
Ponto flutuante (float): Números com casas decimais, como 1.5, 2.75, 3.14, etc.
String (str): Textos, como "Olá, mundo!", "Python é legal", etc.
Booleano (bool): Valores True ou False.
Conversão de Tipos:

É possível converter um tipo de dado para outro usando funções específicas.
Exemplo:
'''
nome = 'Douglas'
print(type(nome))
numero_inteiro = 10
print(type(numero_inteiro))
numero_float = float(numero_inteiro)  # Converte um inteiro para um float
print(type(numero_float))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
```

2 - Operadores Python

2.1 - Operadores Matemáticos

```
'''
Os operadores aritméticos em Python permitem realizar operações matemáticas
básicas, como adição, subtração, multiplicação, divisão e exponenciação. São
ferramentas essenciais para qualquer programador Python, desde iniciantes até
experientes.
'''
```

```
# Soma (+)
soma = 5 + 2
print(soma)
```

```
7
```

```
# Subtração (-)
menos = 5 - 2
print(menos)
```

```
3
```

```
# Multiplicação (*)
vezes = 5 * 2
print(vezes)
```

```
10
```

```
# Divisão (/)
dividir = 5 / 2
print(dividir)
```

```
2.5
```

```
# Parte Inteira da divisão (//)
inteiro = 5 // 2
print(inteiro)
```

```
2
```

```
# Resto da divisão (%)
resto = 5 % 2
print(resto)
```

```
1
```

```
# Exponenciação (**)
elevado = 5 ** 2
print(elevado)
```

```
25
```

2.2 - Operadores de Comparação

```
'''
Os operadores de comparação em Python permitem verificar se duas expressões são
iguais, diferentes, maiores, menores, etc. São ferramentas essenciais para tomar
decisões e realizar comparações em seus programas. Retornam com resultado
verdadeiro (True) ou Falso (False) (tipo bool)
'''
```

```
# Operador de Igualdade (==)
igualdade = 3 == 4
print(igualdade)
```

```
False
```

```
# Operador de Diferença (!=)
diferente = 3 != 4
print(diferente)
```

```
True
```

```
# Operador maior que (>)
maior = 3 > 4
print(maior)
```

```
False
```

```
# Operador menor que (<)
menor = 3 < 4
print(menor)
```

```
True
```

```
# Operador maior ou igual a (>=)
maior_igual = 3 >= 4
print(maior_igual)
```

```
False
```

```
# Operador menor ou igual a (<=)
menor_igual = 3 <= 4
print(menor_igual)
```

```
True
```

```
# Operador de identidade (is)
a = 7
b = 7.0
print(a is b)
print(a == b)
```

```
False
True
```

```
# Operador de identidade (is not)
a = 7
b = 7.0
print(a is not b)
print(a != b)
```

2.3 Operadores de Associação

```
'''
Os operadores de associação in e not in em Python permitem verificar se um
elemento está presente ou não em uma coleção, como listas, tuplas, dicionários e
strings. São ferramentas úteis para filtrar dados e realizar comparações em seus
programas. Assim como os aoperadores de comparação eles retornam
Verdadeiro (True) ou Falso (False)
'''
```

```
# Operador in
nome = 'Douglas'
print('D' in nome)
```

```
True
```

```
# Operador not in
nome = 'Douglas'
print('D' not in nome)
```

```
False
```

2.4 - Operadores Lógicos (AND, OR e NOT)

```
'''
Os operadores lógicos em Python permitem combinar expressões booleanas
(True ou False) para formar novas expressões booleanas. São ferramentas
essenciais para tomar decisões e realizar comparações complexas em seus
programas. São eles:

E (and): Retorna True se ambas as expressões forem True.
Exemplo: x > 0 and y < 10.

Ou (or): Retorna True se pelo menos uma das expressões for True.
Exemplo: x == 0 or y == 10.

Não (not): Inverte o valor da expressão.
Exemplo: not (x > 0).
'''
```

```
# Operador AND retorna verdadeiro se todas as condições forem verdadeiras
a = 3
b = 4
c = 5
print(a < b and b < c)
print(a < b and b > c)
```

```
True
False
```

```
# Operador OR retorna verdadeiro se pelo menos uma das condições for verdadeira
a = 3
b = 4
c = 5
print(a < b or b < c) # verdadeiro
print(a < b or b > c) # verdadiro
print(a > b or a > c) # Falso
```

```
True
True
False
```

```
# Operador not (inverte o resulta de verdadeiro para falso e de falso para verdadeiro)
a = 3
b = 4
c = 5
print(not(a > b))
print(not(a < b))
```

```
True
False
```

3 - Strings

3.1 - Formatação de Strings (f-strings)

```
'''
As f-strings em Python são uma maneira concisa e elegante de formatar strings.
Elas permitem incorporar expressões Python diretamente dentro de strings,
tornando o código mais legível e fácil de manter.

Como usar f-strings:

Para usar f-strings, basta prefixar uma string literal com a letra "f".
Dentro da string, você pode usar expressões Python entre chaves "{}":
'''
```

```
cliente = 'Douglas'
idade = 48
print(f'O nome do cliente é {cliente}, e ele tem {idade} anos de idade.')
```

```
O nome do cliente é Douglas, e ele tem 48 anos de idade.
```

```
taxa = 2/3
print(taxa)
print(f'O resultado da taxa é aproximadamente {taxa:.2f}')
```

```
0.6666666666666666
O resultado da taxa é aproximadamente 0.67
```

3.2 - Entrada definida pelo usuário (função input)

```
'''
A função input() em Python permite que você obtenha entrada do usuário durante a
execução do programa. É uma ferramenta essencial para interagir com o usuário e
coletar dados para processamento.

Como usar a função input():

A função input() é simples de usar. Basta fornecer uma mensagem entre parênteses
para instruir o usuário sobre o que digitar:
'''
```

```
nome = input('Informe seu primeiro nome: ')
print(f'Olá, {nome}!')
```

```
Informe seu primeiro nome: Douglas
Olá, Douglas!
```

```
nome = input('Informe o seu primeiro nome: ')
massa = float(input('Informe seu peso em kg: '))
altura = float(input('Informe sua altura em metros na forma decimal: '))
imc = massa / (altura ** 2)
print(f'Olá {nome}! Seu IMC é {imc:.2f}')
```

```
Informe o seu primeiro nome: Douglas
Informe seu peso em kg: 80
Informe sua altura em metros na forma decimal: 1.85
Olá Douglas! Seu IMC é 23.37
```

3.3 - Posições de caracteres nas strings

```
'''
Este bloco traz possível soluções para quando queremos saber qual caracter temos
em uma posição (índice) ou sequência de caracteres em um intervalo
'''
# 0    1   2   3   4   5  6  7  8  9 10 11 12 13 (índice)
# -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 (índice reverso)
# 1º 2º 3º 4º 5º 6º 7º 8º 9º 10º 11º 12º 13º 14º (ordinal)
# d  o  u  g  @  g  m  a  i   l   .   c   o   m
email = 'doug@gmail.com'
print(email[4])
print(email[:]) # imprimir um intervalo de todos os caracteres da string email
print(email[0:4]) # selecionando o intervalo a partir do índice 0 até o índice 3
print(email[2:]) # seleciona o intervalo a partir do índice 2 até o final
print(email[:5]) # seleciona o intervalo de todos os caracteres até o de índice 4
print(email[0:5:2]) # seleciona último caracter de uma sequência
'''
resumo do fatiamento de strings
nome_variavel[indice de começo : índice de parada : de quanto em quanto]
nome_variavel[início : fim : passo]
'''
```

```
@
doug@gmail.com
doug
ug@gmail.com
doug@
du@
```

```
# Contar a quantidade de carcteres em uma sequência
email = 'doug@gmail.com'
print(len(email))
qtde = len(email)
print(qtde)
```

```
14
14
```

```
cpf = 12345678910
print(type(cpf))
cpf = str(cpf)
print(type(cpf))
```

```
<class 'int'>
<class 'str'>
```

```
# Operador IN

email = 'doug@gmail.com'

print('u' in email)
```

```
True
```

```
# Operador NOT IN
email = 'doug@gmail.com'

print('u' not in email)
```

```
False
```

```
# Método capitalize converte em maiúscula apenas a primeira letra de uma string
nome = 'douglas'
print(nome)
nome = nome.capitalize()
print(nome)
```

```
douglas
Douglas
```

```
# Método Title, serve para deixar a primeira letra de cada palavra maiúscula
nome = 'douglas almeida ribeiro'
print(nome)
nome = nome.capitalize()
print(nome)
nome = nome.title()
print(nome)
```

```
douglas almeida ribeiro
Douglas almeida ribeiro
Douglas Almeida Ribeiro
```

```
# Método upper deixa todas as letras em maiúscula
nome = 'Douglas de almeida ribeiro'
print(nome)
nome = nome.upper()
print(nome)
```

```
Douglas de almeida ribeiro
DOUGLAS DE ALMEIDA RIBEIRO
```

```
# Método para contar quantas vezes um determinado caracter aparece em uma string
email = 'doug@gmail.com'
email.count('@')
```

```
1
```

```
# Método que mostra a posição de um determinado caracter (caso apareça mais de uma vez, ele por padrão mostra
email = 'doug@gmail.com'
email.find('o')
```

```
1
```

```
# Método startswith verifica se uma string começa com um determinado caracter ou sequência de caracteres
email = 'doug@gmail.com'
email.startswith('doug')
```

```
True
```

```
# Método endswith verifica se uma string termina com um determinado caracter ou sequência de caracteres
email = 'doug@gmail.com'
email.endswith('.com')
```

```
True
```

```
# Método isnumeric() verifica se o conteúdo de uma string é ou não numérico
cpf = '12345678910'
cpf.isnumeric()
```

```
False
```

```
# Método isalpha() verifica se o conteúdo de uma string é ou não letra
nome = 'Douglas Ribeiro'
nome.isalpha()
```

```
False
```

```
# Método isalnum verifica se o conteúdo de uma string é ou não combinação de letra e número
nome = 'Dougla5'
nome.isalnum()
```

```
True
```

```
# Método strip remove espaços indesejados no começo e/ou final da string
nome = ' Douglas '
print(len(nome))
nome = nome.strip()
print(len(nome))
```

```
9
7
```

```
# Método replace substitui um caracter por outro
# nome_da_string.replace('caracter_indesejado', 'caeacter_desejado')
preco = 'R$ 10.50'
preco = preco.replace('.',',')
print(preco)
```

```
R$ 10,50
```

```
# Método split (divide uma string em duas ou mais partes por meio de um separador)
nome = 'Douglas Almeida Ribeiro'
nomes = nome.split()
print(nomes)
```

```
['Douglas', 'Almeida', 'Ribeiro']
```

```
# Método split (divide uma string em duas ou mais partes por meio de um separador)
date = '12/05/2023'
dia_mes_ano = date.split('/')
print(dia_mes_ano)
```

```
['12', '05', '2023']
```

```
'''
Crie um programa que receba do usuário dois ou mais nomes separados
por espaços e no mesmo input e os armazene em uma variável chamada nomes
'''
```

```
nomes = input('Informe dois ou mais nomes separados por espaços: ')
print(nomes)
'''
Imagine que você deseja guardar os nomes informados em um estrutura única
porém separando os nomes
'''
nomes = nomes.split()
print(nomes)
```

```
Informe dois ou mais nomes separados por espaços: Douglas Daniela Pedro Maria
Douglas Daniela Pedro Maria
['Douglas', 'Daniela', 'Pedro', 'Maria']
```

```
nomes = input('Informe dois ou mais nomes separados por espaços: ')
print(nomes)
'''
Imagine que você deseja guardar os nomes informados em um estrutura única
porém separando os nomes
'''
nomes = nomes.split()
print(nomes)
print(type(nomes))
```

```
Informe dois ou mais nomes separados por espaços: Douglas Pedro
Douglas Pedro
['Douglas', 'Pedro']
<class 'list'>
```

4 - Estruturas de dados

4.1 - Listas

```
# Uma lista é um objeto que pode ser vazio, ou ter um ou mais itens
lista = ['dado1', 'dado2','dado3','dado4', 'dado_n']
#índices    0         1       2       3       4       ou:
#          -5        -4      -3      -2      -1
#           1º        2º      3º      4º      5º
'''
Qual é a informação presente na lista cuja posição é o
índice 3?
'''
print(f'A informação presente na lista na posição de índice 3 é: "{lista[3]}"')

'''
Quais dados estão presentes no intervalo que vai do 1º ao 3º elementos da lista?
'''
print(f'Os valores no intervalo do 1º ao 3º elementos são: "{lista[0:3]}"')
```

```
A informação presente na lista na posição de índice 3 é: "dado4"
Os valores no intervalo do 1º ao 3º elementos são: "['dado1', 'dado2', 'dado3']"
```

```
produtos = ['monitor', 'celular', 'notebook', 'desktop', 'impressora', 'teclado']
precos = [600, 2000, 5000, 3000, 1500, 200]
vendas = [300, 500, 100, 700, 500, 1000]
estoques = [5000] * 6 #  Este comando cria uma lista que tem 6 elementos com o mesmo valor (5000)
```

```
print(f'a lista produtos tem {len(produtos)} elementos')
print(f'a lista precos tem {len(precos)} elementos')
print(f'a lista vendas tem {len(vendas)} elementos')
print(f'a lista estoques tem {len(estoques)} elementos')
```

```
a lista produtos tem 6 elementos
a lista precos tem 6 elementos
a lista vendas tem 6 elementos
a lista estoques tem 6 elementos
```

```
'''
Elabore um programa que receba do usuário um número inteiro e positivo
de no máximo 5, e automaticamente, exiba o nome do produto correspondente,
seu preço, sua quantidade vendida, o valor faturado em R$, e quantas unidades
ainda restam em estoque desse produto.
```

```
'''
produtos = ['monitor', 'celular', 'notebook', 'desktop', 'impressora', 'teclado']
precos = [600, 2000, 5000, 3000, 1500, 200]
vendas = [300, 500, 100, 700, 500, 1000]
estoques = [5000] * 6 #  Este comando cria uma lista que tem 6 elementos com o mesmo valor (5000)
```

```
i = int(input('Informe um número inteiro e positivos de 0 a 5: ')) # recebe do usuário um número inteiro e posi
print(f'O produto escolhido foi: {produtos[i]}') # Exibe o nome do produto referente ao número informado pelo u
print(f'seu preço é de: R$ {precos[i]:,.2f}')
print(f'sua quantidade vendida fou de {vendas[i]} unidades')
print(f'o seu faturamento foi de R$ {(vendas[i]*precos[i]):,.2f} ')
print(f'e ainda restam {estoques[i]-vendas[i]} unidades em estoque')
```

```
Informe um número inteiro e positivos de 0 a 5: 2
O produto escolhido foi: notebook
seu preço é de: R$ 5,000.00
sua quantidade vendida fou de 100 unidades
o seu faturamento foi de R$ 500,000.00
e ainda restam 4900 unidades em estoque
```

```
i = int(input('Informe um número inteiro e positivos de 0 a 5: '))
print(f'''
O produto escolhido foi: {produtos[i]}, seu preço é de: R$ {precos[i]:,.2f},
sua quantidade vendida fou de {vendas[i]} unidades,
o seu faturamento foi de R$ {(vendas[i]*precos[i]):,.2f},
e ainda restam {estoques[i]-vendas[i]} unidades em estoque''')
```

```
Informe um número inteiro e positivos de 0 a 5: 2
O produto escolhido foi: notebook, seu preço
é de: R$ 5,000.00, sua quantidade vendida fou de 100 unidades,
 o seu faturamento foi de R$ 500,000.00, o seu faturamento
 foi de R$ 500,000.00
```

```
'''
Inserindo valores em uma lista
'''
produtos = ['monitor', 'celular', 'notebook', 'desktop', 'impressora', 'teclado']
precos = [600, 2000, 5000, 3000, 1500, 200]
vendas = [300, 500, 100, 700, 500, 1000]
estoques = [5000] * 6 #  Este comando cria uma lista que tem 6 elementos com o mesmo valor (5000)

'''
Inserindo um novo produto na lista produtos
'''
produtos.append('mouse')
print(produtos)
precos.append(10)
print(precos)
vendas.append(1000)
print(vendas)
estoques.append(5000)
print(estoques)
```

```
['monitor', 'celular', 'notebook', 'desktop', 'impressora', 'teclado', 'mouse']
[600, 2000, 5000, 3000, 1500, 200, 10]
[300, 500, 100, 700, 500, 1000, 1000]
[5000, 5000, 5000, 5000, 5000, 5000, 5000]
```

```
'''
Inserindo valores em um índice específico de uma lista
'''
produtos = ['monitor', 'celular', 'notebook', 'desktop', 'impressora', 'teclado']
precos = [600, 2000, 5000, 3000, 1500, 200]
vendas = [300, 500, 100, 700, 500, 1000]
estoques = [5000] * 6 #  Este comando cria uma lista que tem 6 elementos com o mesmo valor (5000)
'''
método insert
Imagine que você deseja inserir o produto microfone na primeira posição da lista
(índice 0), sem excluir nenhum produto previamente existente
'''
produtos.insert(0,'microfone')
print(produtos)
precos.insert(0,700)
print(precos)
vendas.insert(0,50)
print(vendas)
estoques.insert(0,5000)
print(estoques)
```

```
['microfone', 'monitor', 'celular', 'notebook', 'desktop', 'impressora', 'teclado']
[700, 600, 2000, 5000, 3000, 1500, 200]
[50, 300, 500, 100, 700, 500, 1000]
[5000, 5000, 5000, 5000, 5000, 5000, 5000]
```

```
'''
A principal diferença entre append e insert é a posição que o elemento é
inserido, no append por padrão o elemento é inserido na última posição e no
insert é possível escolher a posição onde vamos inserir o novo elemento
'''
```

```
'''
Excluindo elementos de uma lista
1º método: (pop) ele exclui com a referência do índice
'''
produtos = ['monitor', 'celular', 'notebook', 'desktop', 'impressora', 'teclado']
precos = [600, 2000, 5000, 3000, 1500, 200]
vendas = [300, 500, 100, 700, 500, 1000]
estoques = [5000] * 6 #  Este comando cria uma lista que tem 6 elementos com o mesmo valor (5000)

'''
Suponha que você deseja excluir o produto teclado da lista de produtos
'''
produtos.pop()
print(produtos)
precos.pop()
print(precos)
vendas.pop()
print(vendas)
estoques.pop()
print(estoques)

```

```
['monitor', 'celular', 'notebook', 'desktop', 'impressora']
[600, 2000, 5000, 3000, 1500]
[300, 500, 100, 700, 500]
[5000, 5000, 5000, 5000, 5000]
```

```
'''
Utilizando o método remove para excluir um dado pelo seu valor
'''
produtos = ['monitor', 'celular', 'notebook', 'desktop', 'impressora', 'teclado']
precos = [600, 2000, 5000, 3000, 1500, 200]
vendas = [300, 500, 100, 700, 500, 1000]
estoques = [5000] * 6 #  Este comando cria uma lista que tem 6 elementos com o mesmo valor (5000)

'''
Suponha que você deseja remover o teclado da lista produtos e não saiba qual é
o seu índice
'''
produtos.remove('teclado')
print(produtos)
precos.remove(200)
print(precos)
```

```
['monitor', 'notebook', 'desktop', 'impressora', 'teclado']
[600, 2000, 5000, 3000, 1500]
```

```

'''
Suponha que você deseja saber se na lista produtos existe o termo celular
'''
produtos = ['monitor', 'celular', 'notebook', 'desktop', 'impressora', 'teclado']
precos = [600, 2000, 5000, 3000, 1500, 200]
vendas = [300, 500, 100, 700, 500, 1000]
estoques = [5000] * 6 #  Este comando cria uma lista que tem 6 elementos com o mesmo valor (5000)

print('celular' in produtos)


```

```
True
```

```
produtos = ['monitor', 'celular', 'notebook', 'desktop', 'impressora', 'teclado']
precos = [600, 2000, 5000, 3000, 1500, 200]
vendas = [300, 500, 100, 700, 500, 1000]
estoques = [5000] * 6 #  Este comando cria uma lista que tem 6 elementos com o mesmo valor (5000)


'''
Suponha que você deseja saber se na lista produtos existe o termo celular e caso
exista qual é o seu índice - método index
'''
i = produtos.index('celular')


print(f'''
O produto escolhido foi: {produtos[i]}, seu preço é de: R$ {precos[i]:,.2f},
sua quantidade vendida fou de {vendas[i]} unidades,
o seu faturamento foi de R$ {(vendas[i]*precos[i]):,.2f}, o seu estoque
é de {(estoques[i] - vendas[i])} unidades''')
```

```
O produto escolhido foi: celular, seu preço é de: R$ 2,000.00,
sua quantidade vendida fou de 500 unidades,
o seu faturamento foi de R$ 1,000,000.00, o seu estoque
é de 4500 unidades
```

```
produtos = ['monitor', 'celular', 'notebook', 'desktop', 'impressora', 'teclado']
precos = [600, 2000, 5000, 3000, 1500, 200]
vendas = [300, 500, 100, 700, 500, 1000]
estoques = [5000] * 6 #  Este comando cria uma lista que tem 6 elementos com o mesmo valor (5000)
'''
Imagine que você deseja permitir que o usuário consulte a existência de um
produto pelo nome e caso esse produto exista, o programa retorna o nome do
produto, seu preço unitário de venda, sua quantidade vendida, seu faturamento e
quantas unidades ainda restam em estoique desse produto.
'''

i = produtos.index(input('Informe nome do produto que deseja consultar: ').lower())

print(f'''
O produto escolhido foi: {produtos[i]}, seu preço é de: R$ {precos[i]:,.2f},
sua quantidade vendida fou de {vendas[i]} unidades,
o seu faturamento foi de R$ {(vendas[i]*precos[i]):,.2f}, o seu estoque
é de {(estoques[i] - vendas[i])} unidades''')
```

```
Informe nome do produto que deseja consultar: TECLADO

O produto escolhido foi: teclado, seu preço é de: R$ 200.00,
sua quantidade vendida fou de 1000 unidades,
o seu faturamento foi de R$ 200,000.00, o seu estoque
é de 4000 unidades
```

4.2 - Tuplas

```
'''
Tuplas são estruturas de dados imutáveis
'''
# Criando tuplas
nomes = ('Douglas', 'Daniela', 'Pedro', 'Maria')
idades = (47, 8, 68, 67)
alturas = (1.85, 1.32, 1.75, 1.69)
tupla_vazia = ()
print(type(nomes))
print(nomes)
```

```
<class 'tuple'>
('Douglas', 'Daniela', 'Pedro', 'Maria')
```

```
# Fatiando tuplas
nomes = ('Douglas', 'Daniela', 'Pedro', 'Maria')
idades = (47, 8, 68, 67)
alturas = (1.85, 1.32, 1.75, 1.69)
print(nomes[0])
print(idades[1:3])
print(alturas[-1])
```

```
Douglas
(8, 68)
1.69
```

```
# convertendo objetos em tuplas e vice-versa
lista = [1,2,3]
tupla = tuple(lista)
nova_lista = list(tupla)
print(lista)
print(tupla)
print(nova_lista)
```

```
[1, 2, 3]
(1, 2, 3)
[1, 2, 3]
```

```
# verificando a existência de um elemento em uma tupla
lista = [1,2,3]
tupla = tuple(lista)
nova_lista = list(tupla)
print(4 in tupla)
print(4 not in tupla)
```

```
False
True
```

4.3 - Dicionários

```
carro = {'marca':'vw','modelo':'gol','ano':'2014','motor':'1.0'}
type(carro)
```

```
dict
```

```
# Inserindo uma chave um valor em um dicionário existente
carro = {'marca':'vw','modelo':'gol','ano':'2014','motor':'1.0'}
carro['cor'] = 'azul'
print(carro)
```

```
{'marca': 'vw', 'modelo': 'gol', 'ano': '2014', 'motor': '1.0', 'cor': 'azul'}
```

```
carro['km'] = 280000
print(carro)
```

```
{'marca': 'vw', 'modelo': 'gol', 'ano': '2014', 'motor': '1.0', 'cor': 'azul', 'km': 280000}
```

```
carro = {'marca':'vw','modelo':'gol','ano':'2014','motor':'1.0'}
'''
buscar um valor em um dicionário diretamente pela sua chave pode ser
um problema se por exemplo essa chave não existir no dicionário original
'''
carro['marca']
```

```
'vw'
```

```
carro = {'marca':'vw','modelo':'gol','ano':'2014','motor':'1.0'}
'''
buscar um valor em um dicionário diretamente pela sua chave pode ser
um problema se por exemplo essa chave não existir no dicionário original
'''
carro['doc']
```

```
---------------------------------------------------------------------------
------
KeyError                                  Traceback (most recent call
last)
<ipython-input-30-7c0be97081e7> in <cell line: 6>()
      4 um problema se por exemplo essa chave não existir no
dicionário original
      5 '''
----> 6 carro['doc']
```

```
carro = {'marca':'vw','modelo':'gol','ano':'2014','motor':'1.0'}
'''
Alternativa
'''
carro.get('marca')
```

```
'vw'
```

```
carro = {'marca':'vw','modelo':'gol','ano':'2014','motor':'1.0'}
'''
Alternativa
'''
carro.get('doc')
'''
Resumindo: para os dicionários o método get é bem semelhante à busca
diretamente pela chave, mas com uma importante diferença: pelo método
get, caso a chave não exista no dicionário original, ao onvés de travar o
programa, o método retorna vazio
'''
```

```
# Removendo o último para de chave e valor de um dicionário
carro = {'marca':'vw','modelo':'gol','ano':'2014','motor':'1.0'}
carro.popitem()
```

```
('motor', '1.0')
```

```
# Excluindo um item de um dicionário
carro = {'marca':'vw','modelo':'gol','ano':'2014','motor':'1.0'}
carro.pop('marca')
print(carro)
```

```
{'modelo': 'gol', 'ano': '2014', 'motor': '1.0'}
```

```
# deixando um dicionário sem elemento algum
carro = {'marca':'vw','modelo':'gol','ano':'2014','motor':'1.0'}
print(carro)
carro.clear()
print(carro)
```

```
{'marca': 'vw', 'modelo': 'gol', 'ano': '2014', 'motor': '1.0'}
{}
```

4.4 - Sets (conjuntos)

```
# Criando conjuntos
nomes = {'Douglas', 'Daniela', 'Pedro', 'Maria'}
pesos = {78, 32, 86, 74}
print(nomes)
print(pesos)
print(type(nomes))
```

```
{'Pedro', 'Douglas', 'Maria', 'Daniela'}
{32, 78, 74, 86}
<class 'set'>
```

```
# impossibilidade de receber valores duplicados
alturas = {1.85, 1.34, 1.78, 1.69, 1.34}
print(alturas)
```

```
{1.69, 1.78, 1.85, 1.34}
```

```
# Adição de elemento único em um conjunto
loja_a = {'garfo', 'faca', 'caneca', 'copo'}
loja_b = {'copo', 'caneca', 'taça', 'tigela'}
print(loja_a)
print(loja_b)
loja_a.add('prato')
loja_b.add('pote')
print(loja_a)
print(loja_b)
# Adição de mais de um elemento em um conjunto
loja_a.update(['bule', 'chaleira'])
print(loja_a)
# União de conjuntos
loja_c = loja_a.union(loja_b)
print(loja_c)
```

```
{'garfo', 'faca', 'copo', 'caneca'}
{'copo', 'taça', 'tigela', 'caneca'}
{'garfo', 'copo', 'caneca', 'faca', 'prato'}
{'copo', 'tigela', 'taça', 'pote', 'caneca'}
{'garfo', 'copo', 'bule', 'caneca', 'chaleira', 'faca', 'prato'}
{'garfo', 'copo', 'tigela', 'bule', 'chaleira', 'caneca', 'taça', 'faca', 'pote', 'prato'}
```

```
# Intersecção de conjuntos
loja_a = {'garfo', 'faca', 'caneca', 'copo'}
loja_b = {'copo', 'caneca', 'taça', 'tigela'}
interseccao = loja_a.intersection(loja_b)
print(interseccao)
```

```
{'copo', 'caneca'}
```

```
# Verificando os elementos exclusivos na comparação de conjuntos
loja_a = {'garfo', 'faca', 'caneca', 'copo'}
loja_b = {'copo', 'caneca', 'taça', 'tigela'}
only_a = loja_a.difference(loja_b)
only_b = loja_b.difference(loja_a)
print(only_a)
print(only_b)
```

```
{'garfo', 'faca'}
{'taça', 'tigela'}
```

```
# Excluíndo elementos de um conjunto pelo método remove
loja_a = {'garfo', 'faca', 'caneca', 'copo'}
loja_b = {'copo', 'caneca', 'taça', 'tigela'}
loja_a.remove('faca')
print(loja_a)
'''
esse método remove o elemento caso ele exista, caso não existe ele 'crasha' o
código
'''
```

```
---------------------------------------------------------------------
------
KeyError                                  Traceback (most recent call
last)
<ipython-input-50-695fa26e86da> in <cell line: 4>()
      2 loja_a = {'garfo', 'faca', 'caneca', 'copo'}
      3 loja_b = {'copo', 'caneca', 'taça', 'tigela'}
----> 4 loja_a.remove('carro')
      5 print(loja_a)

KeyError: 'carro'
```

```
# Excluíndo elementos de um conjunto pelo método dicard
loja_a = {'garfo', 'faca', 'caneca', 'copo'}
loja_b = {'copo', 'caneca', 'taça', 'tigela'}
loja_a.discard('carro')
print(loja_a)
```

```
{'garfo', 'faca', 'copo', 'caneca'}
```

```
# Excluindo todos os elementos de um conjunto
loja_a = {'garfo', 'faca', 'caneca', 'copo'}
loja_a.clear()
print(loja_a)
```

```
set()
```

5 - Estruturas de controle de fluxo

5.1 - IF ELIF ELSE

```
'''
if comparação for verdadeira:
  o que deve ser feito caso o resultado da comparação do if for True
else:
  o que deve ser feito caso o resultado do if seja False
'''
```

```
'''
if comparação for verdadeira:
  o que deve ser feito caso o resultado da comparação do if for True
elif segunda comparação for verdadeira:
  o que deve ser feito caso o resultado da 2ª comparação do if for True
else:
  o que deve ser feito caso o resultado do if seja False e do elif também seja False
'''
```

```
'''
Imagine que você deseja pedir ao usuário que digite uma nota de 0 a 10
e caso a nota seja maior ou igual 6 seja exibida a mensagem: Parabéns você foi
aprovado!, caso contrário exiba: Infelizmente você não foi aprovado dessa vez!
continue tentando.
'''
nota = float(input('Informe uma nota de 0 a 10: '))
if nota >= 6:
  print('Parabéns você foi aprovado!')
else:
  print('Infelizmente você não foi aprovado dessa vez! continue tentando.')
```

```
Informe uma nota de 0 a 10: 5.99999
Infelizmente você não foi aprovado dessa vez! continue tentando.
```

```
'''
Agora imagine que você tem três possibilidades:
1ª Nota maior ou igual a 6: 'Parabéns você foi aprovado!'
2ª Nota menor que 6, mas maior ou igual a 4: 'Você deverá fazer uma prova de recuperação'
3ª Nota menor que 4:'Infelizmente você não foi aprovado dessa vez! continue tentando.'
'''
nota = float(input('Informe uma nota de 0 a 10: '))
if nota >= 6:
  print('Parabéns você foi aprovado!')
elif nota >= 4:
  print('Você deverá fazer uma prova de recuperação')
else:
  print('Infelizmente você não foi aprovado dessa vez! continue tentando.')
```

```
Informe uma nota de 0 a 10: 3.999
Infelizmente você não foi aprovado dessa vez! continue tentando.
```

```
'''
Agora imagine que você tem quatro possibilidades:
1ª Nota maior ou igual a 6: 'Parabéns você foi aprovado!'
2ª Nota menor que 6, mas maior ou igual a 4: 'Você deverá fazer uma prova de recuperação online'
3ª Nota menor 4, mas maio ou igual a 2: 'Você deverá fazer uma prova de recuperação presencial'
4ª Nota menor que 2:'Infelizmente você não foi aprovado dessa vez! continue tentando.'
'''
nota = float(input('Informe uma nota de 0 a 10: '))
if nota >= 6:
  print('Parabéns você foi aprovado!')
elif nota >= 4:
  print('Você deverá fazer uma prova de recuperação online')
elif nota >= 2:
  print('Você deverá fazer uma prova de recuperação presencial')
else:
  print('Infelizmente você não foi aprovado dessa vez! continue tentando.')
```

```
Informe uma nota de 0 a 10: 1.99
Infelizmente você não foi aprovado dessa vez! continue tentando.
```

```
'''
Igine que no exemplo das notas, você deseja diferenciar a mensagem para o aluno
que tirar nota máxima, como por exemplo escrever:
Parabéns, você foi aprovado com nota máxima!
'''
nota = float(input('Informe uma nota de 0 a 10: '))
if nota == 10:
  print('Parabéns, você foi aprovado com nota máxima!')
else:
  if nota >= 6:
    print('Parabéns você foi aprovado!')
  elif nota >= 4:
    print('Você deverá fazer uma prova de recuperação online')
  elif nota >= 2:
    print('Você deverá fazer uma prova de recuperação presencial')
  else:
    print('Infelizmente você não foi aprovado dessa vez! continue tentando.')
```

```
Informe uma nota de 0 a 10: 1.9
Infelizmente você não foi aprovado dessa vez! continue tentando.
```

```
'''
Igine que no exemplo das notas, você deseja diferenciar a mensagem para o aluno
que tirar nota máxima, como por exemplo escrever:
Parabéns, você foi aprovado com nota máxima!
'''
nota = float(input('Informe uma nota de 0 a 10: '))
if nota != 10:
  if nota >= 6:
    print('Parabéns você foi aprovado!')
  elif nota >= 4:
    print('Você deverá fazer uma prova de recuperação online')
  elif nota >= 2:
    print('Você deverá fazer uma prova de recuperação presencial')
  else:
    print('Infelizmente você não foi aprovado dessa vez! continue tentando.')
else:
  print('Parabéns, você foi aprovado com nota máxima!')
```

```
Informe uma nota de 0 a 10: 10
Parabéns, você foi aprovado com nota máxima!
```

```
'''
Imagine que você deseja que um usuário digite dois números quaisquer, e na
sequência o programa exibe o maior deles
'''
a = float(input('Informe um número qualquer: '))
b = float(input('Informe outro número qualquer: '))
if a == b:
  print('Você informou dois números iguais!')
else:
  if a > b:
    print(f'O maior número informado é chamado de "a" e é igual a {a}')
  else:
    print(f'O maior número informado é chamado de "b" e é igual a {b}')
```

```
Informe um número qualquer: 5
Informe outro número qualquer: 5
Você informou dois números iguais!
```

5.2 - Estrutura de repetição - FOR

```
palavra = 'love' # cria a variável palvra que recebe o texto love
for letra in palavra: # declara o seguinte: para cada letra em palavra faça algo:
  print(letra) # define o "algo" da linha anterior como exiba cada letra
```

```
l
o
v
e
```

```
palavra = 'love' # cria a variável palvra que recebe o texto love
for a in palavra: # declara o seguinte: para cada letra em palavra faça algo:
  print(a) # declara o seguinte: para cada letra em palavra faça algo:
```

```
l
o
v
e
```

```
lista = ['a','b','c','d'] # cria uma lista com as letras a,b,c,d
for item in lista: # declara que para cada item encontrado na lista faça algo
  print(item) # difine o que deve ser feito com cada item encontrado (exibí-lo)
```

```
a
b
```

```
c
d
```

```
tupla = (1,2,3,4) # Cria uma tupla com os números 1,2,3,4
for numero in tupla: # declara que para cada numero encontrado seja feito algo
  print(numero) # Define o que deve ser feito com cada número encontrado
```

```
1
2
3
4
```

```
for i in range(4):# declara que para cada i (índice) encontrado em uma sequencia de qutro números (começando
  print(i) # exiba cada número gerado na sequência declarada anteriormente
```

```
0
1
2
3
```

```
'''
crie uma estrutura que percorra uma sequência e imprima os índices dessa sequência
'''
palavra = 'love'# cria a variável palvra que recebe o texto love
for i in range(4):# declara que para cada i (índice) encontrado em uma sequencia de qutro números (começando
  print(palavra[i]) # define que seja exibida cada letra da palavra associada a cada índice gerando na declar
```

```
l
o
v
e
```

```
palavra = 'love'# cria a variável palvra que recebe o texto love
for i in range(len(palavra)):# declara que para cada i (índice) encontrado em uma sequencia de n números (com
  print(palavra[i]) # define que seja exibida cada letra da palavra associada a cada índice gerando na declar
```

```
l
o
v
e
```

```
palavra = 'love' # cria a variável palvra que recebe o texto love
for i, v in enumerate(palavra):# declara que para cada i (índice) encontrado em uma sequencia de n números (c
  print(i,v) # define que seja exibida cada letra da palavra associada a cada índice gerando na declaração fo
```

```
0 l
1 o
2 v
3 e
```

```
cidade = 'botucatu' # cria a variável cidade que recebe o texto botucatu
for i in range(1,8,2):# declara que para cada i (índice) encontrado em uma sequencia de n números (começando
  print(cidade[i]) # define que seja exibida cada letra da palavra associada a cada índice gerando na declara
```

```
o
u
a
u
```

5.2.1 - Método sum

```
# método sum
'''
soma os vamlores de uma sequência, caso ela seja numérica
exemplo:
'''
numeros = [1,2,3,4]
print(sum(numeros))
```

```
10
```

```
'''
Crie um programa que receba do usuário n notas (1 ou mais notas), calcule a
média aritmética delas e caso a média seja maior ou igual a 6 exiba a mensagem
Parabéns, você foi aprovado, caso a média seja menor que 6 mas maior ou igual a
4 exiba a mensagem você está em recuperação, caso a média seja menor que 4 exiba
a mensagem  Infelizmente você não foi aprovado e caso a média seja igual a
10 exiba a mensagem Parabéns você foi aprovado com nota máxima.
 Obd.: as n notas digitadas pelo usuário deverão ser informadas em um único input
 separadas por espaços.
'''
notas = input('Informe as notas de 0 a 10 separadas por espaços: ')
notas = notas.split()
for i, v in enumerate(notas):
  notas[i] = float(v)
media = sum(notas)/len(notas)
print(media)
if media == 10:
  print('Parabéns você foi aprovado com nota máxima!')
else:
  if media >= 6:
    print('Parabéns você foi aprovado!')
  elif media >= 4:
    print('Você ficou de recuperação!')
  else:
    print('Infelizmente você não foi aprovado!')
```

```
Informe as notas de 0 a 10 separadas por espaços: 6 6 6 6
6.0
Parabéns você foi aprovado!
```

```
'''
Alternativa usando o range
'''
notas = input('Informe as notas de 0 a 10 separadas por espaços: ')
notas = notas.split()
for i in range(len(notas)):
  notas[i] = float(notas[i])
media = sum(notas)/len(notas)
print(media)
if media == 10:
  print('Parabéns você foi aprovado com nota máxima!')
else:
  if media >= 6:
    print('Parabéns você foi aprovado!')
  elif media >= 4:
    print('Você ficou de recuperação!')
  else:
    print('Infelizmente você não foi aprovado!')
```

```
'''
Crie uma lista com o nome de idades com os seguintes números 48, 55, 62 77 78 80
'''
idades = [48,55,62,77,78,80]

'''
Agora converta cada valor da lista em uma string
'''

for i in range(len(idades)):
  idades[i] = str(idades[i])
print(idades)

'''
alternativa utilizando o enumerate
'''
for i, v in enumerate(idades):
  idades[i] = str(v)
print(idades)
```

```
['48', '55', '62', '77', '78', '80']
['48', '55', '62', '77', '78', '80']
```

```
'''
Crie uma entrada de dados numéricos (notas de 0 a 10) em um único input onde os
valores são informados e separados por espaços, e armazene isso tudo
em uma variável chamada nota, a seguir verifique qual tipo de dados
está armazenado nessa variável.
'''
nota = input('Informe as notas de 0 a 10 separadas por espaços: ')
print(type(nota))

'''
Agora separe os valores armazenados em nota utilizando os espaços como
separadores e armazene tudo em um objeto chamado notas.
'''
notas = nota.split()

'''
Agora converta cada valor do objeto notas em número do tipo float,
e na sequência exiba o objeto notas e a soma de todos os valores deste
objeto
'''

for i in range(len(notas)):
  notas[i] = float(notas[i])
print(sum(notas))

# Alternativa usando o enumerate

for i, v in enumerate(notas):
  notas[i] = float(v)
print(sum(notas))

'''
Agora elabore um código que a partir dos valores da lista notas calcule a
média aritmética e exiba o valor dessa média
'''

media = sum(notas)/len(notas)
print(media)

'''
Agora crie um código que caso a média seja maior ou igual a 6 exiba a mensagem
Parabéns, você foi aprovado, caso a média seja menor que 6 mas maior ou igual a
4 exiba a mensagem você está em recuperação, caso a média seja menor que 4 exiba
a mensagem  Infelizmente você não foi aprovado e caso a média seja igual a
10 exiba a mensagem Parabéns você foi aprovado com nota máxima.
'''

if media == 10:
  print('Parabéns você foi aprovado com nota máxima!')
else:
  if media >= 6:
    print('Parabéns você foi aprovado!')
  elif media >= 4:
    print('Você ficou de recuperação!')
  else:
    print('Infelizmente você não foi aprovado!')
```

```
Informe as notas de 0 a 10 separadas por espaços: 10 10 10
<class 'str'>
30.0
30.0
10.0
Parabéns você foi aprovado com nota máxima!
```

5.3 - Laço While

```
'''
Imagine que você deseja criar um programa que inicialmente solicite ao usuário
uma quantidade de notas que ele deseja informar armazenando-a em uma variável
chamada n, e na sequência o programa execute um loop while solicitando nota por
nota e armazendando-as em uma lista chamada notas, até que a quantidade de notas
nessa lista seja igual a quantidade informada pelo usuário na variável n.
'''
n = int(input('''Informe um número inteiro que representa a
quantidade de notas que deseja informar: ''' ))
c = 0
notas = []
while c < n:
  notas.append(float(input(f'Informe a {c+1}ª nota de 0 a 10: ')))
  c += 1
print(notas)
```

```
dadosx = input('Informe os valores de x separados por espaços: ')
dadosy = input('''Informe os valores de y separados por espaços
(informe a mesma quantidade de dados informados em x): ''')
x = dadosx.split()
y = dadosy.split()
xq = []
yq = []
xy = []
for i in range(len(x)):
  x[i] = float(x[i])
  y[i] = float(y[i])
  xq.append(x[i]**2)
  yq.append(y[i]**2)
  xy.append(x[i]*y[i])
n = len(x)
sx = sum(x)
sy = sum(y)
sxq = sum(xq)
syq = sum(yq)
sxy = sum(xy)
r = (sxy - (sx * sy / n)) / ((sxq - sx**2/n) * (syq - sy**2/n))**(1/2)
rq = r**2
if rq > 0.5:
  if r > 0:
    print(f'O coeficiente de correlação é {r:.4f} e indica forte correlação positiva')
  else:
    print(f'O coeficiente de correlação é {r:.4f} e indica que temos forte correlação negativa')
else:
  print(f'O coeficiente de correlação é {r:.4f} e indica baixa correlação')
```

```
Informe os valores de x separados por espaços: 1 2 3 4 5
Informe os valores de y separados por espaços
(informe a mesma quantidade de dados informados em x): 8 5 4 2 1
O coeficiente de correlação é -0.9815 e indica que temos forte correlação negativa
```

5.4 - Instrução Break

```
'''
Imagine que você deseja utilizar uma estrutura for combinada com a função range
para percorrer e imprimir uma sequência de tamanho 10, ou seja, que como saída
tenhamos 10 prints de 10 números diferentes
'''
for i in range(10):
  print(f'O {i+1}º valor é {i}')
```

```
O 1º valor é 0
O 2º valor é 1
O 3º valor é 2
O 4º valor é 3
O 5º valor é 4
```

```
O 6º valor é 5
O 7º valor é 6
O 8º valor é 7
O 9º valor é 8
O 10º valor é 9
```

```
'''
Imagine que na mesma situação do exercício anterior você
deseja interromper o ciclo de impressões após o programa ter
impresso 5 valores diferentes
'''
for i in range(10):
  if i == 5:
    break
  print(f'O {i+1}º valor é {i}')
```

```
O 1º valor é 0
O 2º valor é 1
O 3º valor é 2
O 4º valor é 3
O 5º valor é 4
```

5.5 - Instrução continue

```
'''
Ainda imaginando a situação original, suponha que você deseja imprimir
toda a sequencia exceto um determinado valor, por exemplo o 5
'''
for i in range(10):
  if i == 5:
    continue
  print(f'O {i+1}º valor é {i}')
```

```
O 1º valor é 0
O 2º valor é 1
O 3º valor é 2
O 4º valor é 3
O 5º valor é 4
O 7º valor é 6
O 8º valor é 7
O 9º valor é 8
O 10º valor é 9
```

5.6 - Declaração pass

```
'''
Imagine que você deseja que o programa ignore a parada estabelecida pelo if
'''
for i in range(10):
  if i == 5:
    pass
  print(f'O {i+1}º valor é {i}')
```

```
O 1º valor é 0
O 2º valor é 1
O 3º valor é 2
O 4º valor é 3
O 5º valor é 4
O 6º valor é 5
O 7º valor é 6
O 8º valor é 7
O 9º valor é 8
O 10º valor é 9
```

5.7 - while True

```
'''
Imagine que você deseja criar um programa que receba do usuário um número
inteiro que defina quantas vezes será impressa a expressão "olá mundo!" e depois
imprima essa expressão exatamente na quantidade informada pelo usuário
'''
```

```
# Solução 01
n = int(input('Informe quantas vezes você deseja ver a mensagem olá mundo: '))
c = 0
while c < n:
  print('Olá Mundo!')
  c += 1
```

```
Informe quantas vezes você deseja ver a mensagem olá mundo: 4
Olá Mundo!
Olá Mundo!
Olá Mundo!
Olá Mundo!
```

```
# Solução 02 usando while True
n = int(input('Informe quantas vezes você deseja ver a mensagem olá mundo: '))
c = 0
while True:
  if c < n:
    print('Olá Mundo!')
    c += 1
  else:
    break
```

```
Informe quantas vezes você deseja ver a mensagem olá mundo: 3
Olá Mundo!
Olá Mundo!
Olá Mundo!
```

```
'''
O while True é uma estrutura mais flexível que o while pois não traz a condição
na própria lina de declaração while, permitindo assim, que você faça várias
outras condições aninhadas logo abaixo da declaração while True.
'''
```

```
# Exercício
'''
Elabore um programa que que receba um número inteiro e positivo do usuário e
verifique se o número informado é par ou ímpar, se for par imprima uma mensagem
dizendo que o número é par e encerre o loop, caso seja ímpar exiba uma mensagem
informando que o número não é par e peça para o usuário digitar outro número,
até que o número digitado pelo usuário seja par.
'''

while True:
  n = int(input('Informe um número par inteiro e positivo: '))
  if n % 2 == 0:
    print(f'O número {n} é um número par! Parabéns!')
    break
  else:
    print(f'O número {n} não é um número par, favor informe um número par')
```

```
Informe um número par inteiro e positivo: 3
O número 3 não é um número par, favor informe um número par
Informe um número par inteiro e positivo: 7
O número 7 não é um número par, favor informe um número par
Informe um número par inteiro e positivo: 4
O número 4 é um número par! Parabéns!
```

5.8 - Tratamento de erros (TRY e EXCEPT)

```
'''
Elabore um programa que receba do usuário um número qualquer e exiba a mensagem:
Parabéns você digitou um número!
'''
```

```
while True:
  try:
    n = float(input('Informe um número qualquer: '))
    print('Parabéns você digitou um número!')
    break
  except:
    print('o valor informado não é um número')
```

```
Informe um número qualquer: 7,3
o valor informado não é um número
Informe um número qualquer: 4
Parabéns você digitou um número!
```

```
'''
Imagine que você tem uma lista com seis nomes de pessoas e outra com seis idades
Suponha que a posição de cada nome em sua lista corresponde a mesma posição
da sua idade na lista idades. Elabore um código que percorra as duas listas,
verifique se a pessoa é maior ou menor de idade (18 anos) e exiba uma mensagem
com cada nome, sua idade e um texto informando se essa pessoa é maior ou menor
de idade.
'''
nomes = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
idades = [48, 9, 70, 70, 44, 37]
for i in range(len(nomes)):
  if idades[i] >= 18:
    print('*'*50)
    print(f'{nomes[i]} tem {idades[i]} anos e é maior de idade')
    print('*'*50)
  else:
    print('*'*50)
    print(f'{nomes[i]} tem {idades[i]} anos e é menor de idade')
    print('*'*50)
```

```
**************************************************
Douglas tem 48 anos e é maior de idade
**************************************************
**************************************************
Daniela tem 9 anos e é menor de idade
**************************************************
**************************************************
Pedro tem 70 anos e é maior de idade
**************************************************
**************************************************
Maria tem 70 anos e é maior de idade
**************************************************
**************************************************
Eduardo tem 44 anos e é maior de idade
**************************************************
**************************************************
Ester tem 37 anos e é maior de idade
**************************************************
```

```
'''
Imagine que você tem uma lista com seis nomes de pessoas e outra com seis idades
Suponha que a posição de cada nome em sua lista corresponde a mesma posição
da sua idade na lista idades. Elabore um código que percorra as duas listas,
verifique se a pessoa é maior ou menor de idade (18 anos) e exiba uma mensagem
com cada nome, sua idade e um texto informando se essa pessoa é maior ou menor
de idade.
```

```
'''
nomes = ['Douglas','Daniela','Pedro','Maria','Eduardo','Ester']
idades = [48, 9, 70, 70, 44, 37]
for i,v in enumerate(idades):
  if idades[i] >= 18:
    print('*'*50)
    print(f'{nomes[i]} tem {v} anos e é maior de idade')
    print('*'*50)
  else:
    print('*'*50)
    print(f'{nomes[i]} tem {v} anos e é menor de idade')
    print('*'*50)
```

```
**************************************************
Douglas tem 48 anos e é maior de idade
**************************************************
**************************************************
Daniela tem 9 anos e é menor de idade
**************************************************
**************************************************
Pedro tem 70 anos e é maior de idade
**************************************************
**************************************************
Maria tem 70 anos e é maior de idade
**************************************************
**************************************************
Eduardo tem 44 anos e é maior de idade
**************************************************
**************************************************
Ester tem 37 anos e é maior de idade
**************************************************
```

```
'''
Imagine que você deseja criar um programa que receba do usuário o valor de um
determinado produto e a informação se esse produto é nacional ou importado e
retorne para o usuário o valor do imposto a pagar. Considere que se o produto
for nacional a alíquota de imposto é de 18% para valores até R$ 200,00 e 25%
para valores acima de R$ 200,00. Agora se o produto for importado é isento para
valores até R$ 200,00 e pagará 35% para valores acima de R$ 200,00.
'''
```

```
while True:
  try:
    valor = float(input('Informe o valor do produto em R$ '))
    break
  except:
    print('valor inválido')
nac = input('O produto é nacional? 1 para sim ou 2 para não: ')
if nac == '1':
  if valor <= 200:
    imposto = valor * 0.18
  else:
    imposto = valor * 0.25
else:
  if valor > 200:
    imposto = valor * 0.35
  else:
    imposto = 0
print(f'O valor do imposto é R$ {imposto:.2f}')
```

```
Informe o valor do produto em R$ a
valor inválido
Informe o valor do produto em R$ 5
O produto é nacional? 1 para sim ou 2 para não: 1
O valor do imposto é R$ 0.90
```

```
'''
Elabore um programa que receba do usuários duas sequencias de números separados
por espaços com igual quantidade sendo uma sequência em um único input e outra
sequência em outro input único
'''
dadosx = input('Informe uma sequência de números separados por espaços: ')
dadosy = input('''Informe outra sequência de números separados por espaços com a
mesma quantidade informada na primeira: ''')

'''
Agora separe cada sequência em uma lista, exemplo: dadosx separado em uma lista
camada x e dadosy em uma lista chamada y
'''
x = dadosx.split()
y = dadosy.split()

'''
percorra cada uma das listas criadas e para cada valor encontrado converta-o em
um tipo float
'''
for i in range(len(x)):
  x[i] = float(x[i])
for i in range(len(y)):
  y[i] = float(y[i])

'''
Agora crie três listas vazias, sendo xq, yq e xy e as preencha da sequinte forma:

1 - xq: percorra a lista de x e para cada valor encontrado o eleve ao
quadrado e guarde-o na lista xq

2 - yq: percorra a lista de y e para cada valor encontrado o eleve ao
quadrado e guarde-o na lista yq

3 - xy:  percorra a lista de x e a de y e multiplique cada valor de x pelo
seu valor de y correspondente e guarde os resultados na lista xy
'''
xq = []
yq = []
xy = []
for i in range(len(x)):
  xq.append(x[i]**2)
for i in range(len(y)):
  yq.append(y[i]**2)
for i in range(len(x)):
  xy.append(x[i]*y[i])
'''
Agora vamos criar uma variável que armazena o resultado da lista, sendo:
sx = soma dos valores da lista x
sy = soma dos valores da lista y
sxq = soma dos valores da lista xq
syq = soma dos valores da lista yq
sxy = soma dos valores da lista xy
n = a quantidade de elementos de um das listas
mediax = a média da lista x
mediay = a média da lista y
'''
sx = sum(x)
sy = sum(y)
sxq = sum(xq)
syq = sum(yq)
sxy = sum(xy)
n = len(x)
mediax = sx/n
mediay = sy/n

beta = (sxy - (sx * sy /n)) / (sxq - sx**2 / n)
```

```
a = mediay - beta * mediax
print(f'A equação da reta de regressão linear simples é Y = {a:.1f} + {beta:.1f} * x')
```

```
Informe uma sequência de números separados por espaços: 1 2 3 4 5
Informe outra sequência de números separados por espaços com a
mesma quantidade informada na primeira: 1 2 4 5 8
A equação da reta de regressão linear simples é Y = -1.1 + 1.7 * x
```

6 - Compreensão de listas

```
'''
Exemplo 01: suponha que você tem uma lista com idades em anos e esta lista está
desatualizada em exatamente 1 ano, ou seja, para atualizá-la você precisa
acrescentar 1 ano em cada idade presente na lista
'''
idades = [47, 8, 68, 67, 43, 37]

'''
Método 01 - Usando o for
'''
novas_idades = []
for idade in idades:
  novas_idades.append(idade+1)
idades = novas_idades
print(idades)
```

```
[48, 9, 69, 68, 44, 38]
```

```
idades = [47, 8, 68, 67, 43, 37]
for i in range(len(idades)):
  idades[i] = idades[i] + 1
print(idades)
```

```
[48, 9, 69, 68, 44, 38]
```

```
# Método 2 (usando compreensão de listas)
idades = [47, 8, 68, 67, 43, 37]

idades = [idade + 1 for idade in idades]
print(idades)
```

```
[48, 9, 69, 68, 44, 38]
```

```
'''
Exemplo 02: suponha que você tem duas lista de igual tamanho sendo uma com
alturas em metros e a outra com pesos em Kg. Considerando que a posição de cada
elemento em uma lista corresponde ao mesmo índice desse elemento na outra lista
você deseja criar uma lista com imc para cada elemento.
'''
alturas = [1.85,1.23,1.75,1.67,1.82,1.73]
pesos = [70,22,87,64,96,68]

```

```
alturas = [1.85,1.23,1.75,1.67,1.82,1.73]
pesos = [70,22,87,64,96,68]
imc = []
for i in range(len(pesos)):
  imc.append(round(pesos[i]/alturas[i]**2,2))
print(imc)
```

```
[20.45, 14.54, 28.41, 22.95, 28.98, 22.72]
```

```
alturas = [1.85,1.23,1.75,1.67,1.82,1.73]
pesos = [70,22,87,64,96,68]
imc = []
for i,v in enumerate(pesos):
  imc.append(round(v/alturas[i]**2,2))
print(imc)
```

```
[20.45, 14.54, 28.41, 22.95, 28.98, 22.72]
```

```
alturas = [1.85,1.23,1.75,1.67,1.82,1.73]
pesos = [70,22,87,64,96,68]
imc = [pesos[i]/alturas[i]**2 for i in range(len(pesos))]
print(imc)
```

```
[20.45288531775018, 14.541608830722454, 28.408163265306122, 22.948115744558788, 28.982007003985025, 22.72043
```

```
'''
Imagine que você tem uma lista chamada lista e dentro dela você tem outras duas
listas, sendo comidas = ['Arroz', 'Feijão', 'Batata', 'Macarrão'] e a outra
bebidas = ['Tubaína', 'Cajuína', 'Dolly', 'Baré']
e você deseja guardar no objeto bebida_favorita o texto fatiado Dolly
'''

lista = [['Arroz','Feijão','Batata','Macarrão'],['Tubaína','Cajuína','Dolly','Baré']]
bebida_favorita = lista[1][2]
print(bebida_favorita)
```

```
Dolly
```

```
alturas = [1.85,1.23,1.75,1.67,1.82,1.73]
pesos = [70,22,87,64,96,68]
dados = list(zip(pesos, alturas))
print(dados)
imc = [i[0]/i[1]**2 for i in dados]
print(imc)
```

```
[(70, 1.85), (22, 1.23), (87, 1.75), (64, 1.67), (96, 1.82), (68, 1.73)]
[20.45288531775018, 14.541608830722454, 28.408163265306122, 22.948115744558788, 28.982007003985025, 22.72043
```

```
alturas = [1.85,1.23,1.75,1.67,1.82,1.73]
pesos = [70,22,87,64,96,68]
imc = [round(peso / altura**2,2) for peso, altura in zip(pesos, alturas)]
print(imc)
```

```
[20.45, 14.54, 28.41, 22.95, 28.98, 22.72]
```

7 - Funções definidas pelo usuário

```
'''
Imagine que você deseja criar uma função que imprima o "Olá!" todas as vezes
que essa função for executada, sem ter a necessidade de usar o print, por
exemplo
'''
def oi():
  print('Olá!')
'''
O processo de criação de uma função é muito simples, devemos começar com a
palavra reservada "def" seguido de um nome que vamos usar para chamar a função
e dpois parenteses para especificar os argumentos da função, lembrando que os
argumentos de uma função, são opcionais, observe que no caso de função do
exemplo acima, não temos argumentos obrigatórios, e além disso, essa função não
possui retorno
'''

'''
Suponha agora que você deseja guardar o texto de saída da função acima em uma
variável qualquer
'''
#oi()

'''
Observe que a função cumpre o seu papel, ou seja, quando executada, exibe no
console o texto "Olá!"
'''

'''
agora vamos tentar guardar esse texto em uma variável chamada texto
'''

texto = oi()
print(texto)

'''
Nesse caso não temos nada guardado na variável texto pq essa função
não tem retorno
'''
```

```
Olá!
None
'\nNesse caso não temos nada guardado na variável texto pq essa função\nnão tem retorno\n'
```

```
def oi():
  return 'Olá!'
'''
Agora a função oi possui como retorno o texto 'Olá!'
e como ela tem retorno esse retorno pode ser armazenado
em uma variável, por exemplo
'''

texto = oi()
print(texto)
```

```
Olá!
```

7.1 - Funções com argumentos nomeados e argumentos de posição (Argumentos obrigatórios)

```
'''
Imagine que você precisa fazer uma função que calcula automaticamente o valor
total do desconto sobre o valor da compra de um cliente, e esse desconto total,
depende de diferentes percentuais que podem ser aplicados sobre o valor total
da compra do cliente, tais percentuais podem ser referentes a cupon de desconto,
pagamento à vista em pix, clube de vantagens vip e desconto para professores.
suponha que nossos clientes possuem todos os descontos e as taxas são fixas:
cupon 1%
pix 2%
vip 3%
prof 4%
'''
def desconto(valor):
  cupon = valor * 0.01
  pix = valor * 0.02
  vip = valor * 0.03
  prof = valor * 0.04
  return cupon + pix + vip + prof
```

```
print(f'O valor do desconto na sua compra foi de R$ {desconto(1000)}')
```

```
O valor do desconto na sua compra foi de R$ 100.0
```

```
'''
Agora imagine que cupon, pix, vip e prof são variáveis, ou seja
cada vez que a função for executada podem receber valores diferentes,
nesse caso devemos colocá-los também como argumentos na construção
da função
'''

def desconto(valor, cupon, pix, vip, prof):
  desconto = valor * cupon + valor * pix + valor * vip + valor * prof
  return desconto
```

```
desconto(1000,0.01,0.02,0.03,0.04)
```

```
100.0
```

```
desconto(vip=0.03,valor=1000,pix=0.02,cupon=0.01,prof=0.04)
```

```
100.0
```

```
'''
Imagine que você deseja criar uma função que some três números
informados nos argumentos da função quando ela for chamada
'''
def soma1(n1, n2, n3):
  resultado = n1 + n2 + n3
  return resultado
```

```
'''
Imagine que você deseja criar uma função sem argumentos que some três números
informados pelo usuário via inputs e retorne o resultado
'''
def soma():
  n1 = int(input('N1: '))
  n2 = int(input('N3: '))
  n3 = int(input('N3: '))
  resultado = n1 + n2 + n3
  return resultado
```

```
soma()
```

```
N1: 3
N3: 4
N3: 5
12
```

7.2 - Quantidade variável de argumentos na função (ARGS e KWARGS)

```
def desconto2(valor, *args): # cria a função desconto2 com apenas um argumento obrigatório (valor) e os demai
  desconto = 0 # cria a variável desconto com o valor zero, pois a princípio a função foi criada para permiti
  for arg in args:# percorre cada argumento opcional informado na chamada da função e armazenado na lista arg
    desconto += arg * valor # multiplica cada valor da lista args pelo valor da compra e adiciona à variável
  return desconto # retorna o desconto total
```

```
desconto2(1000,0.01,0.02,0.03,0.04)
```

```
100.0
```

```
def desconto3(valor, **kwargs):
  desconto = 0
  if 'cupon' in kwargs:
    desconto += kwargs['cupon'] * valor
  if 'pix' in kwargs:
    desconto += kwargs['pix'] * valor
  if 'vip' in kwargs:
     desconto += kwargs['vip'] * valor
  if 'prof' in kwargs:
    desconto += kwargs['prof'] * valor
  return desconto
```

```
desconto3(1000, vip=0.03)
```

```
30.0
```

8 - Módulos

```
'''
Módulos são scripts que podem ser salvos com a extensão .py e importados
por outros arquivos python

vamos abrir um novo colab e criar um módulo
'''
```

8.1 - Importando outros módulos já previamente instalados

```
# Módulo Numpy
import numpy as np
```

```
# Módulo Pandas
import pandas as pd
```

```
# Módulo Seaborn
import seaborn as sns
```

```
# Módulo Matplotlib
import matplotlib as plt
```

8.2 Instalando módulos