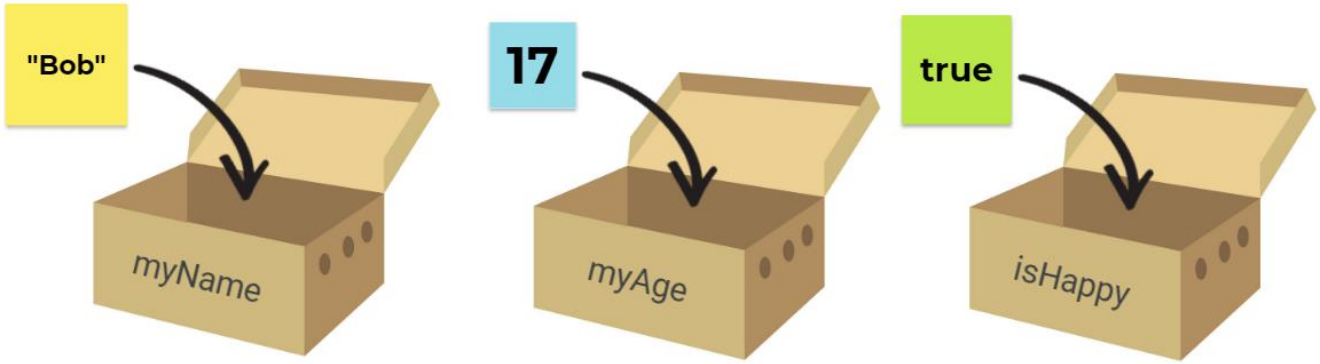
PROGRAMAÇÃO EM PYTHON

VARIÁVEIS

Uma variável é uma localização na ***memória RAM*** do computador que é utilizada para ***armazenar temporariamente*** os dados que são utilizados pelo programa.

As variáveis possuem características como:

- Identificação
- Endereço
- Tipo
- Tamanho
- Valor



PROGRAMAÇÃO EM PYTHON

VARIÁVEIS

Um nome de variável é uma sequência de letras (a → z, A → Z) e números (0 → 9), que devem sempre começar com uma letra. Apenas letras comuns são permitidas.

Letras acentuadas, cedilhas, espaços, caracteres especiais como $, #, @, etc. são proibidos, exceto para o caractere _ (sublinhado/underline).

Além dessa regra é importante também estar atento às palavras reservadas da linguagem.

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

PROGRAMAÇÃO EM PYTHON

VARIÁVEIS

A escolha do estilo de escrita para variáveis, funções e outros elementos do código é mais uma questão de convenção e estilo do que uma questão de funcionalidade direta. No entanto, o estilo de escrita desempenha um papel importante na legibilidade, manutenção e colaboração em projetos de programação. Aqui estão algumas formas de declarar as variáveis:

Exemplos: *Camel case, Pascal case, Snake case e Kebab case*

PROGRAMAÇÃO EM PYTHON

VARIÁVEIS

Camel case: Camel case deve começar com a primeira letra minúscula e a primeira letra de cada nova palavra subsequente maiúscula:

```
valorFinal
```

Pascal case: Também conhecido como “upper camel case” ou “capital case”, pascal case combina palavras colocando todas com a primeira letra maiúscula:

```
ValorFinal
```

PROGRAMAÇÃO EM PYTHON

VARIÁVEIS

Snake case: conhecido também como “underscore case”, utilizamos underline no lugar do espaço para separar as palavras.

```
valor_final
```

Kebab case: Kebab case utiliza o traço para combinar as palavras. Quando o kebab case está em caixa alta, ele é chamado de “screaming kebab case”:

```
valor-final
```

PROGRAMAÇÃO EM PYTHON

VARIÁVEIS

Tipos de dados primitivos ou dados básicos.

Os dados são elementos do mundo exterior que representam, dentro de um computador digital as informações manipuladas pelos seres humanos. Os dados a serem utilizados devem primeiramente ser ***abstraídos*** para serem, então, ***processados***. Eles podem ser classificados em ***três tipos primitivos ou tipos básicos***: ***numéricos*** (representados por valores **numéricos ou reais**), ***caracteres*** (representados por valores **alfabéticos ou alfanuméricos**) e ***lógicos*** (valores dos tipos **falso ou verdadeiro**).

PROGRAMAÇÃO EM PYTHON

VARIÁVEIS

Inteiro

Os dados numéricos positivos e negativos pertencem ao conjunto de números inteiros, excluindo dessa categoria qualquer valor numérico fracionado (que pertencem ao conjunto de números reais). Exemplos de números inteiros: 10, 35, 100, -50, -30

Reais

São reais os números positivos e negativos que pertencem aos números reais, incluindo nessa categoria todos os valores inteiros e fracionários. Exemplos: 10, 35, -50, 45.999, 45.5

PROGRAMAÇÃO EM PYTHON

VARIÁVEIS

Caractere

São ***caracteres*** delimitados pelo símbolo aspas (**" "**), eles são representados por letras (de A até Z) números (de 0 até 9), símbolos (exemplo, todos os símbolos imprimíveis existentes em um teclado) ou palavras contendo esses símbolos.

O tipo ***caractere*** também é conhecido como ***alfanumérico***, ***string*** (em inglês, cordão, colar).

Lógico

São lógicos os valores binários do tipo ***sim*** ou ***não***, ***verdadeiro*** ou ***falso***, ***1*** e ***0***

Também conhecido como tipo ***booleano.***

PROGRAMAÇÃO EM PYTHON

HELLO WORLD

```
#primeiro código de um bom programador.
print("Hello World!")
print("Olá Mundo!")
```

A função ***print*** (imprimir) informa que vamos exibir algo na tela.

PROGRAMAÇÃO EM PYTHON

VARIÁVEIS

```
#exemplo de uma variável com número inteiro.
idade = 34
print(idade)
```

O caractere especial **#** é utilizado para comentar uma linha.
O caractere **=** é utilizado para atribuição.
A função ***print*** (imprimir) informa que vamos exibir algo na tela.

PROGRAMAÇÃO EM PYTHON

VARIÁVEIS

```
#exemplo de uma variável com número real.
altura = 1.83
print(altura)
```

PROGRAMAÇÃO EM PYTHON

VARIÁVEIS

```
#exemplo de uma variável com string.
nome = "Lucas"
print(nome)
```

PROGRAMAÇÃO EM PYTHON

VARIÁVEIS

```
#exemplo de variável com valor lógico booleano.
resultado = True
aprovado = False
```

PROGRAMAÇÃO EM PYTHON

VARIÁVEIS CONSTANTE

Variável Constante.

Variáveis do tipo ***constante são imutáveis***, o seu ***valor não pode ser alterado durante a execução do programa*** (*software*).

Exemplo de um valor constante: valor de ***Pi*** pois possui sempre o mesmo valor (***3,1415...***)

A regra de nomeação das constantes na linguagem de programação Python segue um padrão parecido com as de variáveis, com a diferença de que todas as letras são maiúsculas e separadas por underline " _ ". Porém, por possuir tipagem dinâmica, os valores atribuídos à constantes podem ser alterados sem problemas:

PROGRAMAÇÃO EM PYTHON

VARIÁVEIS CONSTANTE

```
#exemplo de declaração de variável constante.
VALOR_DE_PI = 3.14
print(VALOR_DE_PI)
```

PROGRAMAÇÃO EM PYTHON

ENTRADA DE DADOS

Até agora nossos programas trabalharam apenas com **valores conhecidos, escritos no próprio programa**. No entanto, **o melhor da programação é poder escrever a solução de um problema e aplicá-la várias vezes**. Para isso precisamos melhorar nossos programas de forma a permitir que novos **valores sejam fornecidos durante a sua execução**, de modo que **poderemos executá-los com valores diferentes sem alterar os programas** em si.

Chamamos de ***entrada de dados*** o momento em que seu programa **recebe dados ou valores por um dispositivo de entrada** de dados (como teclado do computador).

PROGRAMAÇÃO EM PYTHON

ENTRADA DE DADOS

A função ***input()*** (entrada) é utilizada para solicitar dados do usuário, ela recebe um parâmetro, que é a mensagem a ser exibida, e retorna o valor digitado pelo usuário.

```
#entrada de dados
nome = input("Digite seu nome")
print("Você digitou:", nome)
```

PROGRAMAÇÃO EM PYTHON

COMENTÁRIOS

Sintaxe: Regras gerais

- ✓ **Identation (Indentação)**
- ✓ **Comentários**

```
#comentário de uma linha
""" comentário
    de várias
    linhas """
```

PROGRAMAÇÃO EM PYTHON

REGRAS

Sintaxe: Regras Gerais

- ✓ **Case sensitive**
- ✓ **Tipagem dinâmica**
- ✓ **Programação em alto nível**
- ✓ **Não suporta sobrecarga de funções**

Tipagem dinâmica se refere a ***não associação dos tipos às variáveis*** e sim aos seus valores; e como tipagem forte, entende-se que ***não é possível realizar operações com valores que possuem tipos incompatíveis de dados***, como, por exemplo, unir um número a uma string.

PROGRAMAÇÃO EM PYTHON

DECLARAÇÃO DE VARIÁVEIS

Declaração de varáveis em Python

Para declarar uma variável em Python é só atribuir um valor a ela.

```
#declaração de variáveis
frase = "Bom dia"
mensagem = "Boa Noite"
preco = 110.50
var = 123
...
var = "uma frase"
```

PROGRAMAÇÃO EM PYTHON

VERIFICAÇÃO

Verificar o tipo da variável

Para verificar o tipo da variável podemos utilizar a função ***type(variável)***

```
#verificar o tipo da variável com a função type()
valor = 110.50
nome = "João"
contador = 0
correto = True
print(type(valor))
print(type(nome))
print(type(contador))
print(type(correto))
```

PROGRAMAÇÃO EM PYTHON

ANÁLISE

Análise da variável

Proporciona compreensão do comportamento dos dados, identificação de erros, otimização de desempenho e validação de entrada.

```
#Identifica se o valor é do tipo numerico
print(n.isnumeric())
#Identifica se o valor é do tipo string
print(n.isalpha())
#Identifica se o valor é do tipo alfanumerico
print(n.isalnum())
#Identifica se o valor esta com as letras maiusculas
print(n.isupper())
#Identifica se o valor esta com as letras minusculas
print(n.islower())
#Identifica se os caracteres são decimais (0-9)
print(n.isdecimal())
```

PROGRAMAÇÃO EM PYTHON

CONVERSÃO DE TIPO

Conversões de tipo

Para realizar a conversão de tipo, utilizaremos as funções do Python.

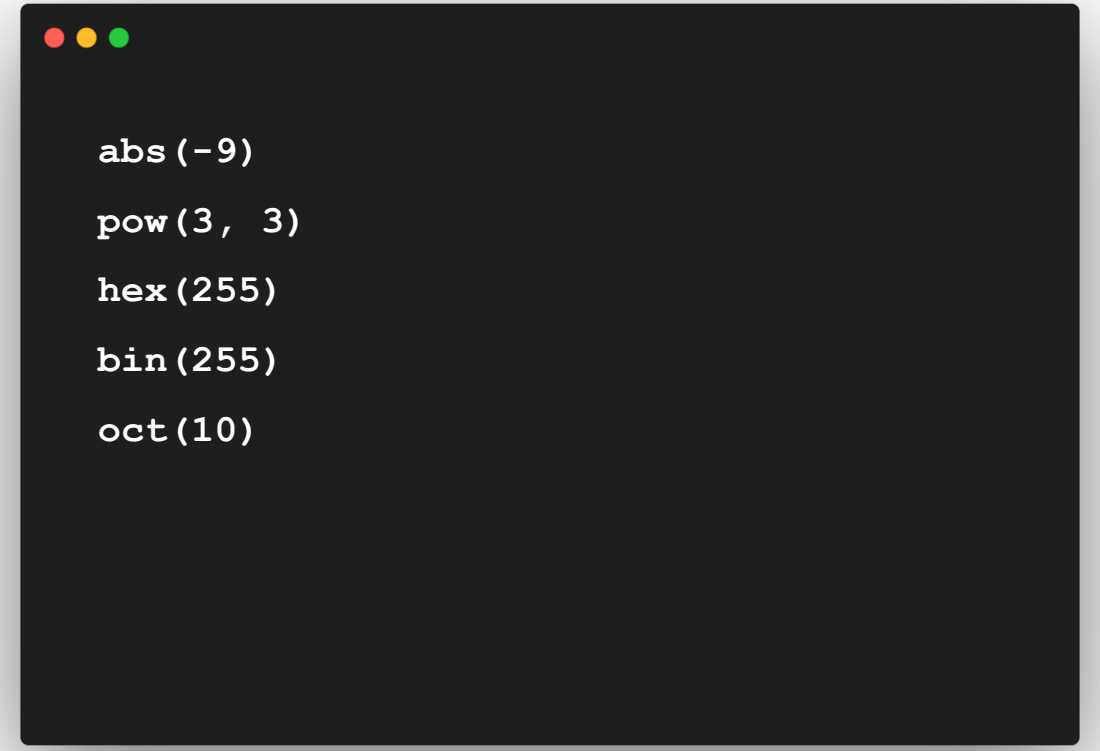
```
float(22/5)
int(5.4)
int(2.9)
float(int(4.9))
int(float(4.9))
int(float(4))
round(4.9)
round(4.5)
int(round(4.9))
```

PROGRAMAÇÃO EM PYTHON

FUNÇÕES BUILT-IN

Funções com números

O ***Python*** é poderoso e possui ***funções embutidas***, veja alguns exemplos ao lado.



PROGRAMAÇÃO EM PYTHON

FUNÇÕES BUILT-IN

Funções built-in (embutidas)

O interpretador do ***Python*** possui várias ***funções e tipos embutidos*** que sempre estão disponíveis. Ao lado listamos todas as funções em ordem alfabética.

https://docs.python.org/pt-br/3/library/functions.html

https://docs.python.org/pt-br/3/library

Funções embutidas

A abs() aiter() all() any() anext() ascii() **B** bin() bool() breakpoint() bytearray() bytes() **C** callable() chr() classmethod() compile() complex() **D** delattr() dict() dir() divmod()	**E** enumerate() eval() exec() **F** filter() float() format() frozenset() **G** getattr() globals() **H** hasattr() hash() help() hex() **I** id() input() int() isinstance() issubclass() iter()	**L** len() list() locals() **M** map() max() memoryview() min() **N** next() **O** object() oct() open() ord() **P** pow() print() property()	**R** range() repr() reversed() round() **S** set() setattr() slice() sorted() staticmethod() str() sum() super() **T** tuple() type() **V** vars() **Z** zip() **_** __import__()

PROGRAMAÇÃO EM PYTHON

OPERADORES LÓGICOS

Operadores lógicos - not, and, or

Para agrupar operações com ***lógica booleana***, utilizaremos ***operadores lógicos***. Python suporta ***três operadores básicos:*** **not** (não), **and** (e), **or** (ou). Esses operadores podem ser traduzidos como **não** (negação), **e** (conjunção) **ou** (disjunção).

and	Retorna True se ambas as afirmações forem verdadeiras
or	Retorna True se uma das afirmações for verdadeira
not	Retorna Falso se o resultado for verdadeiro, inverte o booleano

PROGRAMAÇÃO EM PYTHON

OPERADORES LÓGICOS

Precedência dos operadores lógicos no Python - not, and, or

Precedência dos operadores lógicos em Python

Operador	Nome	Precedência
not	não	1
and	e	2
or	ou	3

Precedência é a ordem na qual os operadores serão avaliados quando o programa for executado.

PROGRAMAÇÃO EM PYTHON

OPERADORES LÓGICOS

Tabela verdade dos operadores lógicos - not, and, or

Operador lógico **not** (não)

V1	not V1
0	1
1	0

Operador lógico **and** (e)

V1	V2	V1 and V2
0	0	0
0	1	0
1	0	0
1	1	1

Operador lógico **or** (ou)

V1	V2	V1 or V2
0	0	0
0	1	1
1	0	1
1	1	1

sigla	significado	inglês
V1	Variável 1	Variable 1
V2	Variável 2	Variable 2
1	Verdadeiro	True
0	Falso	False

PROGRAMAÇÃO EM PYTHON

OPERADORES ARITMÉTICOS

Operadores aritméticos.

Esses operadores são utilizados para criarmos expressões matemáticas comuns, como ***soma, subtração, multiplicação*** e ***divisão***.

Veja quais estão disponíveis no Python:

Operadores aritméticos em Python

Operador	Nome	Função
+	Adição	Realiza a soma
-	Subtração	Realiza a subtração
*	Multiplicação	Realiza a multiplicação
/	Divisão	Realiza a divisão
//	Divisão Inteira	Realiza a divisão, retorna a parte inteira
%	Módulo	Retorna o resto da divisão
**	Exponenciação	Retorna o resultado da elevação da potência pelo outro

PROGRAMAÇÃO EM PYTHON

LINGUAGEM DE PROGRAMAÇÃO

Ordem das operações aritméticas.

Nos Estados Unidos, usa-se um acrônimo chamado **PEMDAS**, que significa **Parênteses Expoentes Multiplicação Divisão Adição Subtração**. É a ordem que o ***Python*** segue também. O erro que as pessoas cometem com o **PEMDAS** é achar que é uma ordem rígida, como “Faça P, depois E, M, D, A, por fim, S”. **A ordem real é você fazer a multiplicação e a divisão (M&D)** em uma etapa, da **esquerda para a direita**, então, fazer a **adição e subtração** em uma etapa da ***esquerda para a direita***. Assim, seria possível rescrever **PEMDAS** como **PE(M&D)(A&S)**

PROGRAMAÇÃO EM PYTHON

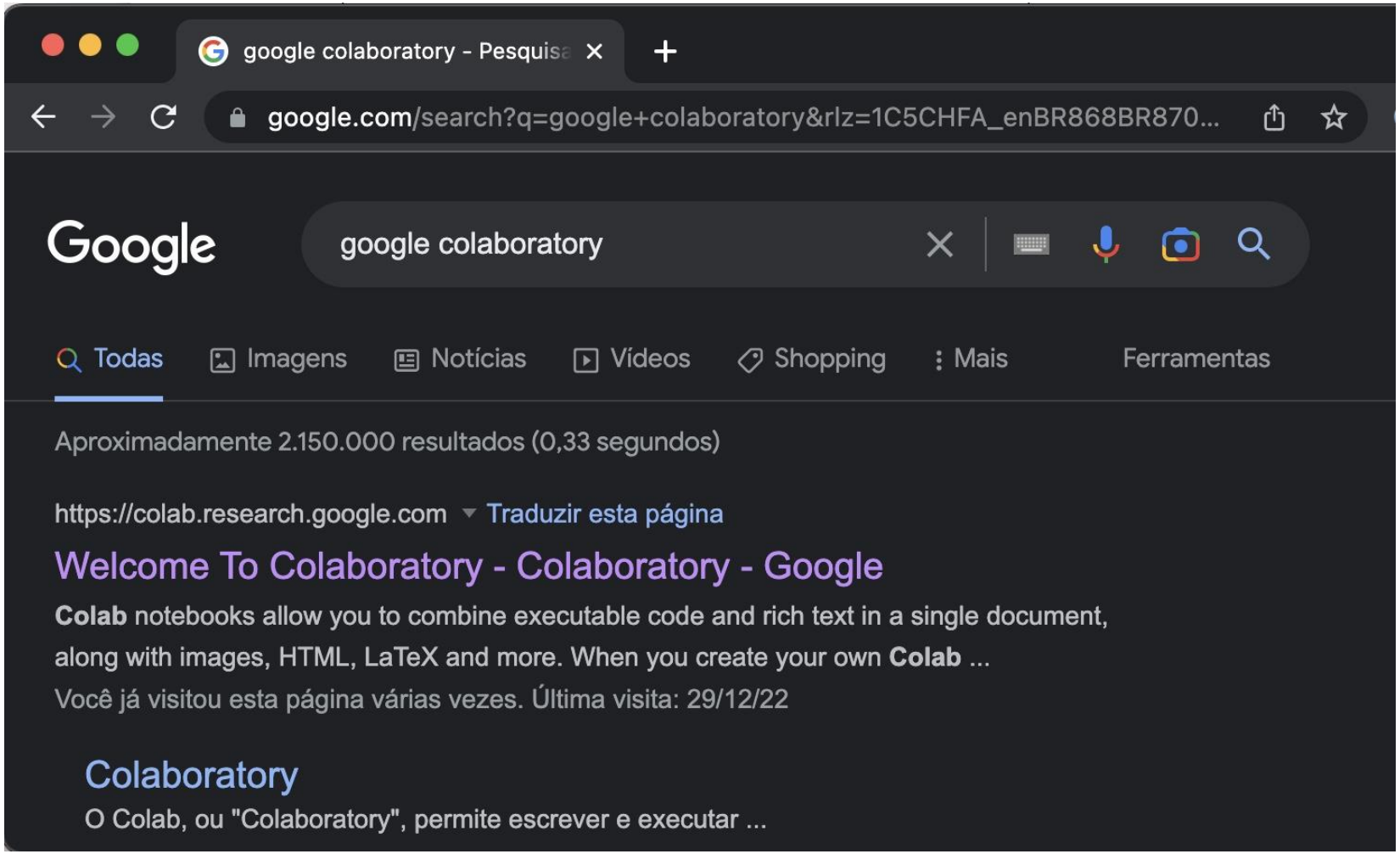
LINGUAGEM DE PROGRAMAÇÃO

Operadores aritméticos em Python

```
#operadores aritméticos em Python
quatro = 4
dois = 2
soma = quatro + dois
subtracao = quatro - dois
multiplicacao = quatro * dois
divisao = quatro / dois
modulo = quatro % dois
exponenciacao = quatro ** dois
```

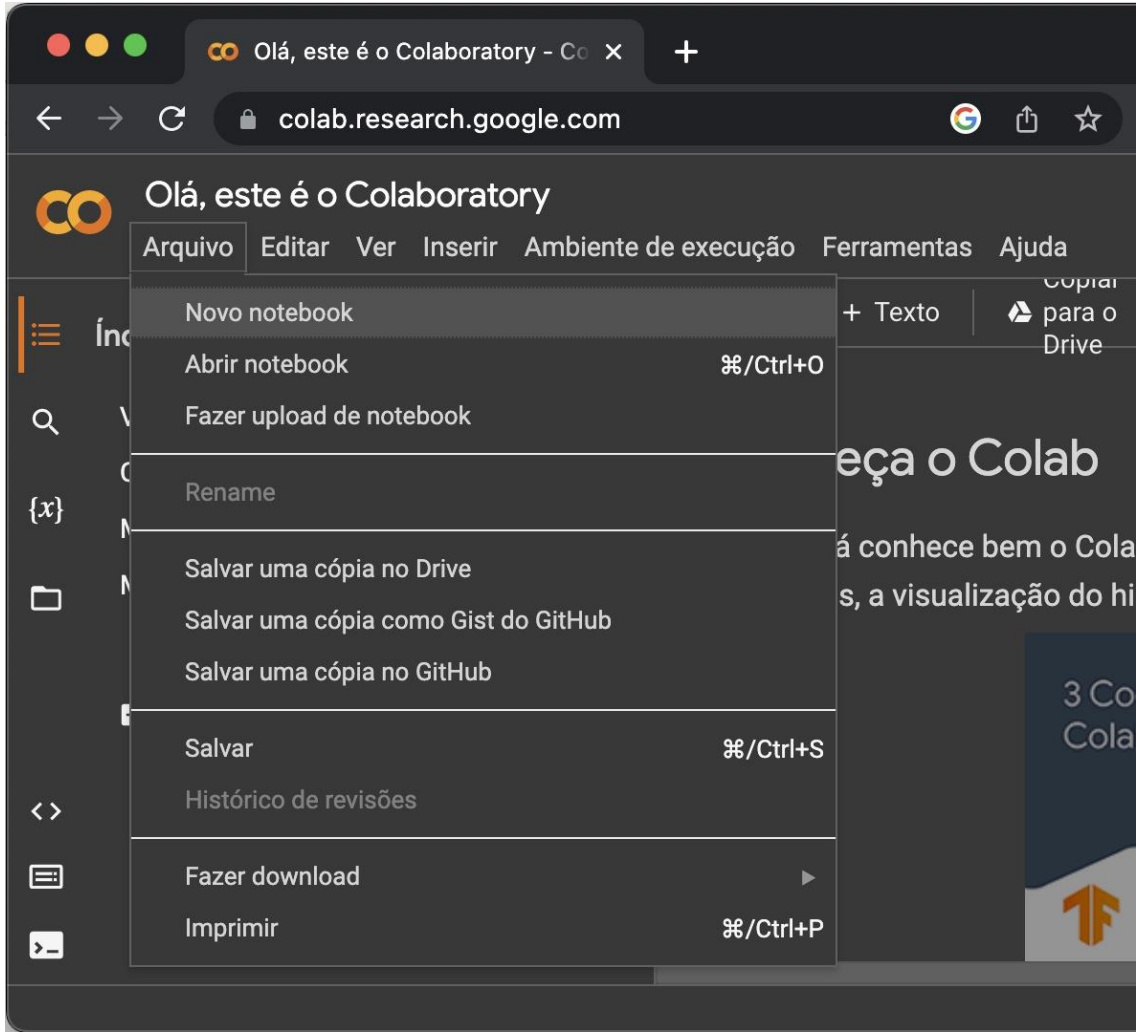
PROGRAMAÇÃO EM PYTHON

FERRAMENTA



PROGRAMAÇÃO EM PYTHON

FERRAMENTA



VAMOS PRATICAR.

ATIVIDADE

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Operadores relacionais ou comparativos.

Há ***oito operadores relacionais ou comparativos*** no ***Python***. Todos eles possuem a mesma prioridade (que é maior do que aquela das operações Booleanas). Comparações podem ser encadeadas arbitrariamente; por exemplo, **x < y <= z** é equivalente a **x < y and y <= z**, exceto que ***y*** é avaliado apenas uma vez (porém em ambos os casos ***z*** não é avaliado de todo quando **x < y** é sabido ser **falso**).

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Tabela de operadores relacionais ou comparativos.

Operação	Significado
`<`	estritamente menor que
`<=`	menor que ou igual
`>`	estritamente maior que
`>=`	maior que ou igual
`==`	igual
`!=`	não é igual
`is`	identidade do objeto
`is not`	identidade de objeto negada

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

```
a = 1
b = 5
c = 2
d = 1
#operadores relacionais em Python
a == b
b > a
a < b
a == d
d != a
d != b
```

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Operadores relacionais ou comparativos.

Esse operadores são utilizados como na matemática. Especial atenção deve ser dada aos operadores **>=** e **<=**. O resultado desses operadores é realmente ***maior ou igual*** e ***menor ou igual***, ou seja, **5 >= 5** é ***verdadeiro***, assim como **5 <= 5**.

O resultado de uma comparação é um valor do tipo ***lógico, booleano***, ou seja ***True*** (verdadeiro) ou ***False*** (falso).
Observe que **==** significa **igual**, enquanto **=** significa **atribuição**.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Operadores relacionais ou comparativos *is* e *is not*

Em Python, ***is*** e ***is not*** são usados para verificar se dois valores estão localizados na mesma parte da memória. Duas variáveis que são iguais não implica que sejam idênticas.

```
x = 5
y = 5
print(x is y)
print(x is not y)
```

O comportamento dos operadores ***is*** e ***is not*** podem ser aplicados a quaisquer dois ***objetos***.

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Print() – sep e end.

sep: Define o separador entre os valores, com valor padrão de espaço (' ').

end: Define o final da linha, com valor padrão de nova linha ('\n').4o

```
print("a", "b", "c", sep="-")
# Saída: a-b-c

print("Hello", end="!")  # Saída: Hello!
print("World")           # Saída: World (na mesma linha)
# Saída: Hello!World
```

LÓGICA DE PROGRAMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO

Formatação

Algumas opções para formatar strings

```
num = 3.14159265359
print(round(num, 2))
print("{:.2f}".format(num))
print(f"A formatação de valores funciona dessa forma também {num:.2f}")
```

VAMOS PRATICAR.

EXEMPLOS E ATIVIDADES DE OPERADORES