

Nome: Leonardo José da Silva Junior

Graduando em Análise e desenvolvimento de sistemas

Monitor de lógica e fundamentos da programação

**Empresas em que desenvolvi projetos:**

**Porto digital** e **Accenture**

Linkedin: Leonardo Jose

Github: [github.com/LeonardoJoseDaSilvaJunior](https://github.com/LeonardoJoseDaSilvaJunior)



Linkedin



Github

# LÓGICA

EM



# PYTHON

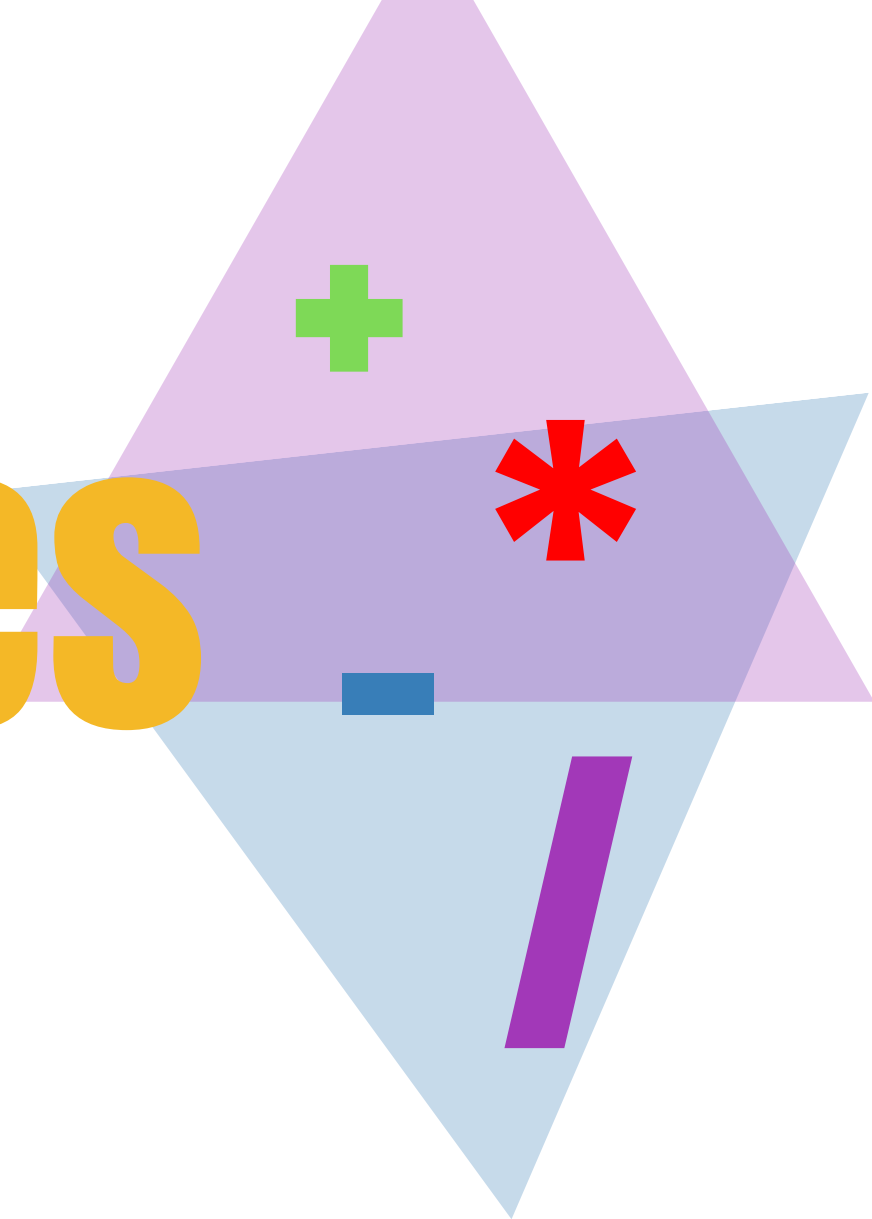
# Operadores

LÓGICOS

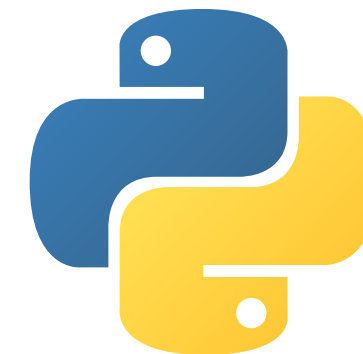
Relacionais

&

Aritméticos



# O que são operadores relacionais ?



Os operadores relacionais trabalham como comparações, igualdades e desigualdades. Eles verificam os valores dos operandos, que ficam cada um de um lado da operação, retornando **VERDADEIRO** ou **FALSO**.

## Operadores relacionais :

Operadores	Descrição	Comparação	Resultado
==	Igual	(2 == 3)	False
!=	Diferente	(2 != 3)	True
>	maior que	(2 > 3)	False
<	Menor que	(2 < 3)	True
>=	Maior ou igual a	(2 >= 3)	False
<=	Menor ou igual a	(2 <= 3)	True

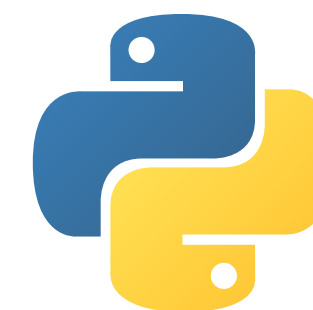
# o que são operadores Aritméticos?



Os operadores aritméticos executam operações matemáticas, como **adição** e **subtração** com operandos.

## Operadores Aritméticos:

Operador	Descrição	Exemplo	Resultado
+	Adição	2 + 2	4
-	Subtração	2 - 2	0
*	Multiplicação	2 * 2	4
/	Divisão	2 / 2	1
//	Divisão inteira	8 // 2	4
%	Resto da divisão	8 % 2	0
**	Potências	4 ** 3	64



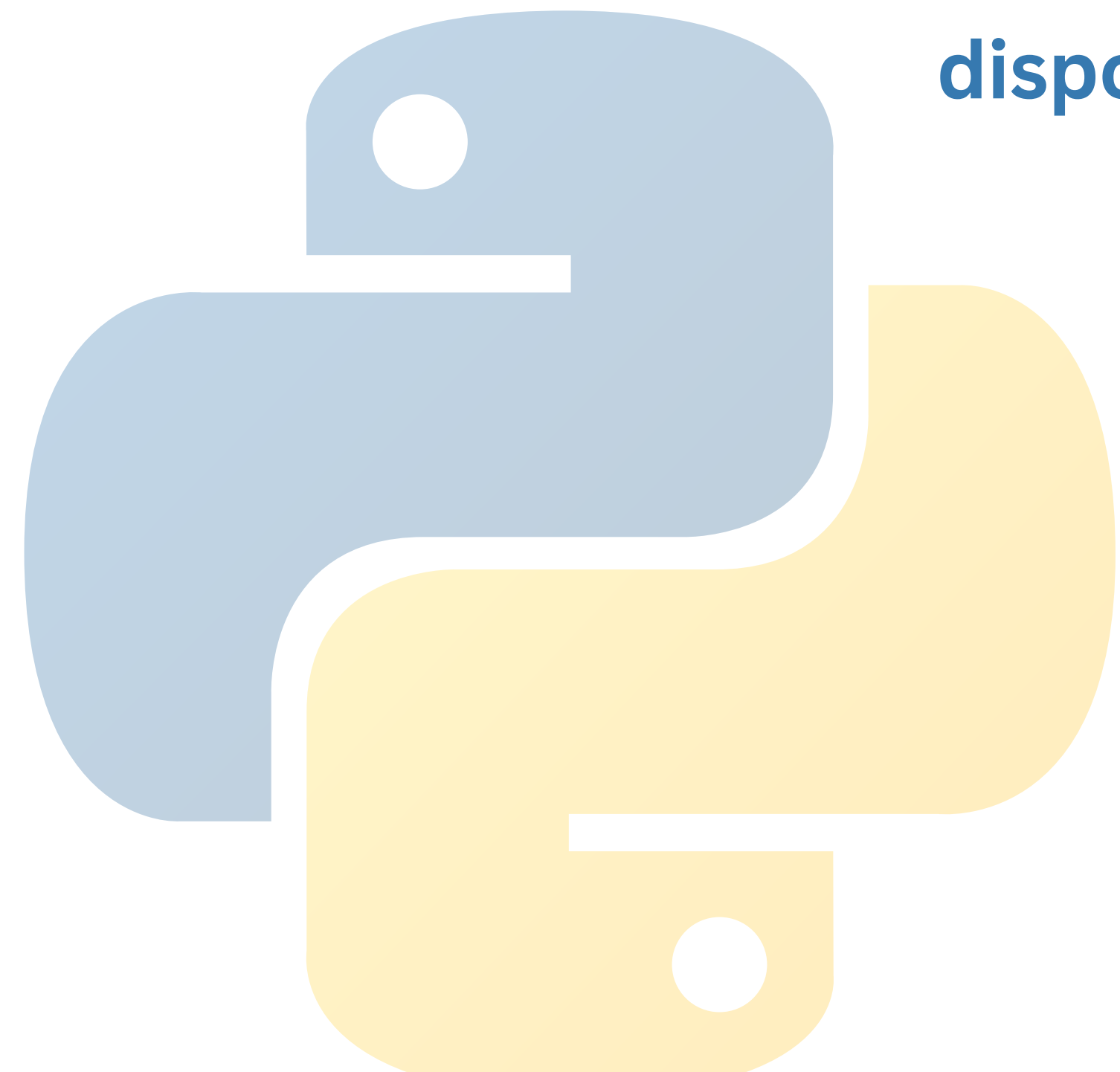
# o que são operadores lógicos ?

Operador **lógico** é um **elemento** que liga as **condições** que compõem um comando de pesquisa. É o operador lógico que indica para o sistema a maneira como se quer que uma palavra esteja em relação à outra dentro da condicional, para que essa **condicional seja atendida ou não**.

## Operadores lógicos em python:

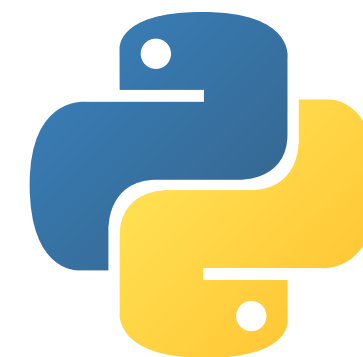
Operador	Descrição	Tipo
and	Retorna True se ambas as condições forem atendidas	Conjunção
or	Retorna True mesmo se apenas uma condição for atendida	Disjunção
not	Nega o resultado	Negação

**Exemplos disponíveis através da imagem e QR code disponibilizados logo abaixo:**





EXPO  
**ENGETEC**  
8ª EDIÇÃO UNIFG



# Tipos de Dados

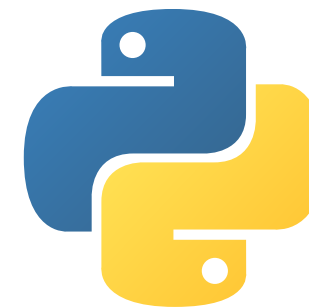




# Tipos de dados



- Inteiro (`int`)
- Ponto Flutuante ou Decimal (`float`)
- Tipo Complexo (`complex`)
- String (`str`)
- Boolean (`bool`)
- Lista (`list`)
- Tupla
- Dicionario (`dic`)



## Tipo Inteiro (int)

O tipo inteiro é um tipo composto por caracteres numéricos (**algarismos**) inteiros.

É um tipo usado para um número que pode ser escrito sem um componente decimal, podendo ter ou não sinal, isto é: ser positivo ou negativo.

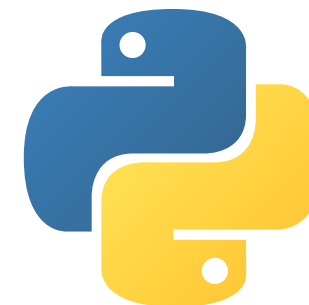
### Por exemplo:

21, 4, 0, e -2048 são números inteiros, enquanto 9.75, 1/2, 1.5 **não** são.

## Ponto Flutuante ou Decimal (float)

É um tipo composto por caracteres numéricos (**algarismo**) decimais.

O famoso ponto flutuante é um tipo usado para números racionais (**números que podem ser representados por uma fração**) informalmente conhecido como “número quebrado”.



## Complexo (complex)

Tipo de dado usado para representar números complexos (**isso mesmo, aquilo que provavelmente estudou no terceiro ano do ensino médio**).

Esse tipo normalmente é usado em cálculos **geométricos e científicos**.

Um tipo complexo contem duas partes: a **parte real** e a **parte imaginária**, sendo que a parte imaginária contem um **"j"** no sufixo.

A função **complex(real[, imag])** do Python possibilita a criação de números imaginários passando como argumento: **real**, que é a parte Real do número complexo e o argumento opcional **imag**, representando a parte imaginária do número complexo.

## String (str)

É um conjunto de **caracteres** dispostos numa determinada ordem, geralmente utilizada para representar **palavras, frases** ou **textos**.

## Listas (list)



A **lista** é um tipo de dado muito importante e que é muito utilizado no dia a dia do desenvolvedor Python!

Listas agrupam um conjunto de elementos variados, podendo conter: **inteiros**, **floats**, **strings**, outras **listas** e **outros tipos**.

Elas são **definidas utilizando-se colchetes** para delimitar a lista e vírgulas para separar os elementos.

## Tuplas (tuple)

Assim como **Lista**, Tupla é um tipo que agrupa um conjunto de elementos. Porém sua forma de definição é diferente, **utilizamos parênteses** e também separado por vírgula.

A diferença para Lista é que Tuplas são imutáveis, ou seja, após sua definição, Tuplas não podem ser modificadas.

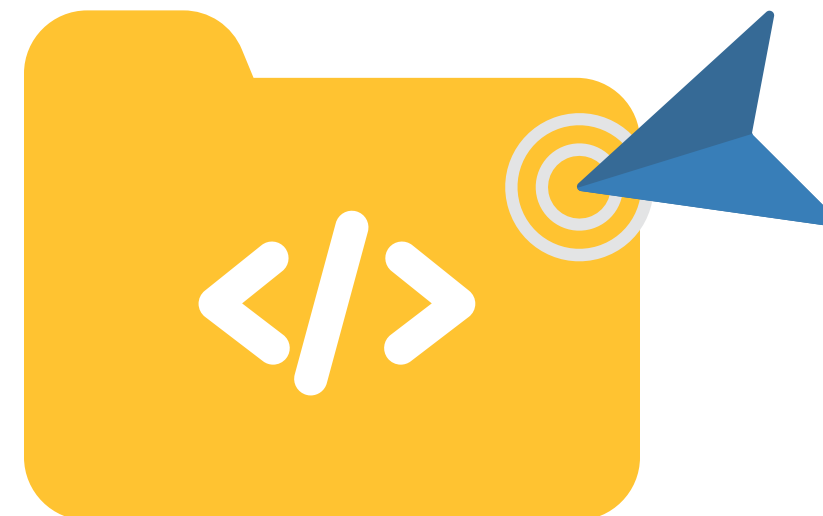
## Dicionários (dict)



Dict é um tipo de dado  **muito flexível**  do Python.

Eles são utilizados para agrupar elementos através da estrutura de  **chave e valor** , onde  **a chave é o primeiro elemento**  seguido por dois pontos e pelo valor.

**Exemplos disponibilizados por meio do QR code e clicando na pasta**



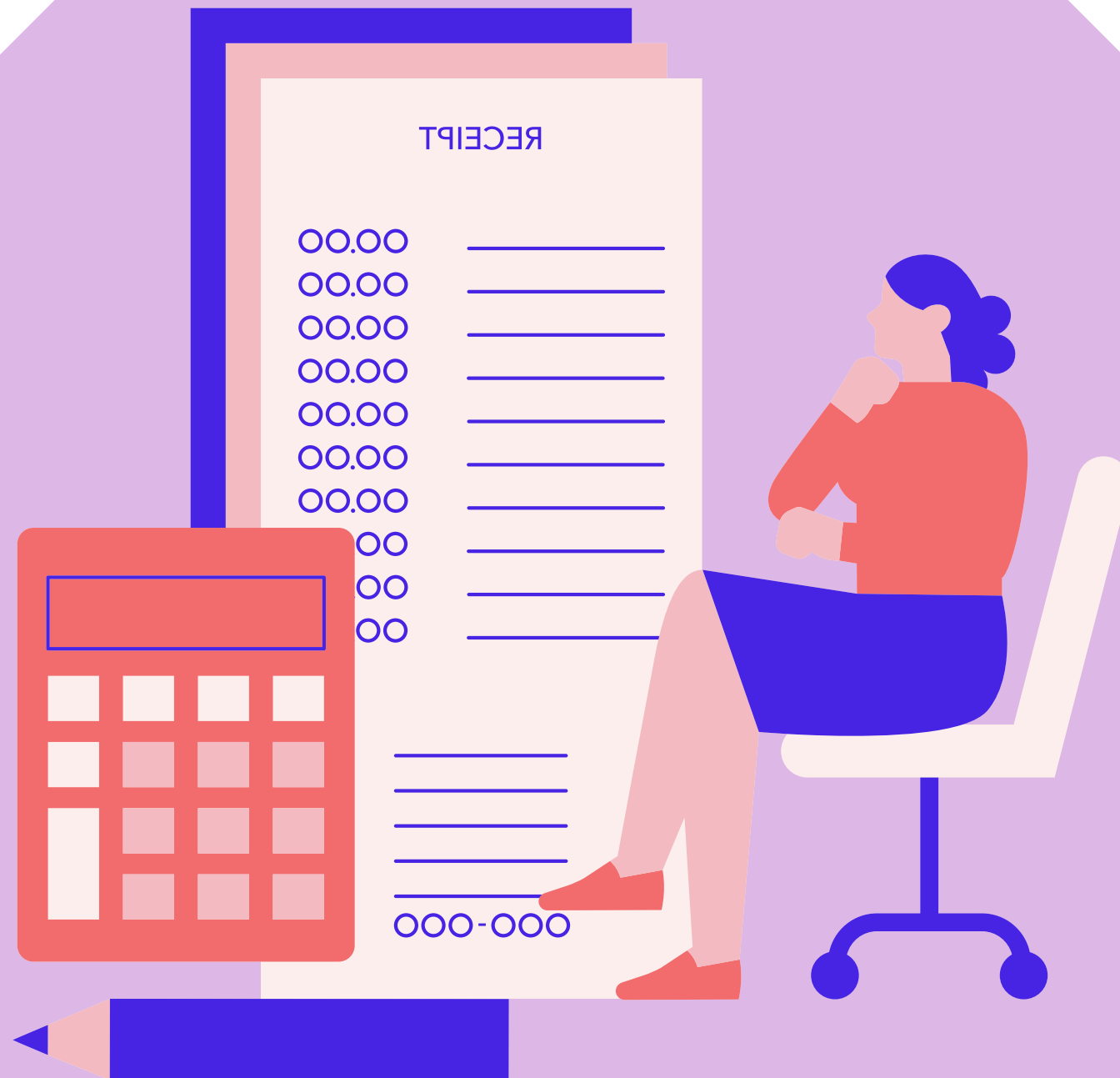
&

# Entrada

input

# Saída de dados

output



# Saida de dados



O **Comando** utilizado para a **saída** de dados na tela do console na Linguagem Python é a função **print()**.

A função **print()** exibe um texto na tela. O texto deve estar entre aspas simples (**' '**) ou aspas duplas (**" "**).

## Caracteres especiais:

**\n** avanço de linha

**\t** tabulação (tab)

**\b** retrocesso (backspace)

**\"** aspas duplas

**\\** barra

Para formatar um texto com várias saídas, utiliza-se uma string formatada (**f-String**) em conjunto com a função **print()**.



## Saida de dados

Existe uma construção que permite formatar a saída com certa facilidade, é o método **format**.

A forma mais simples de usar essa construção é marcar a posição, no comando **print**, onde o conteúdo da variável (ou resultado da expressão) aparecerá.

Para isso, utiliza-se um número entre chaves, indicando a posição que a variável deverá estar no método **format**.

### Exemplo:

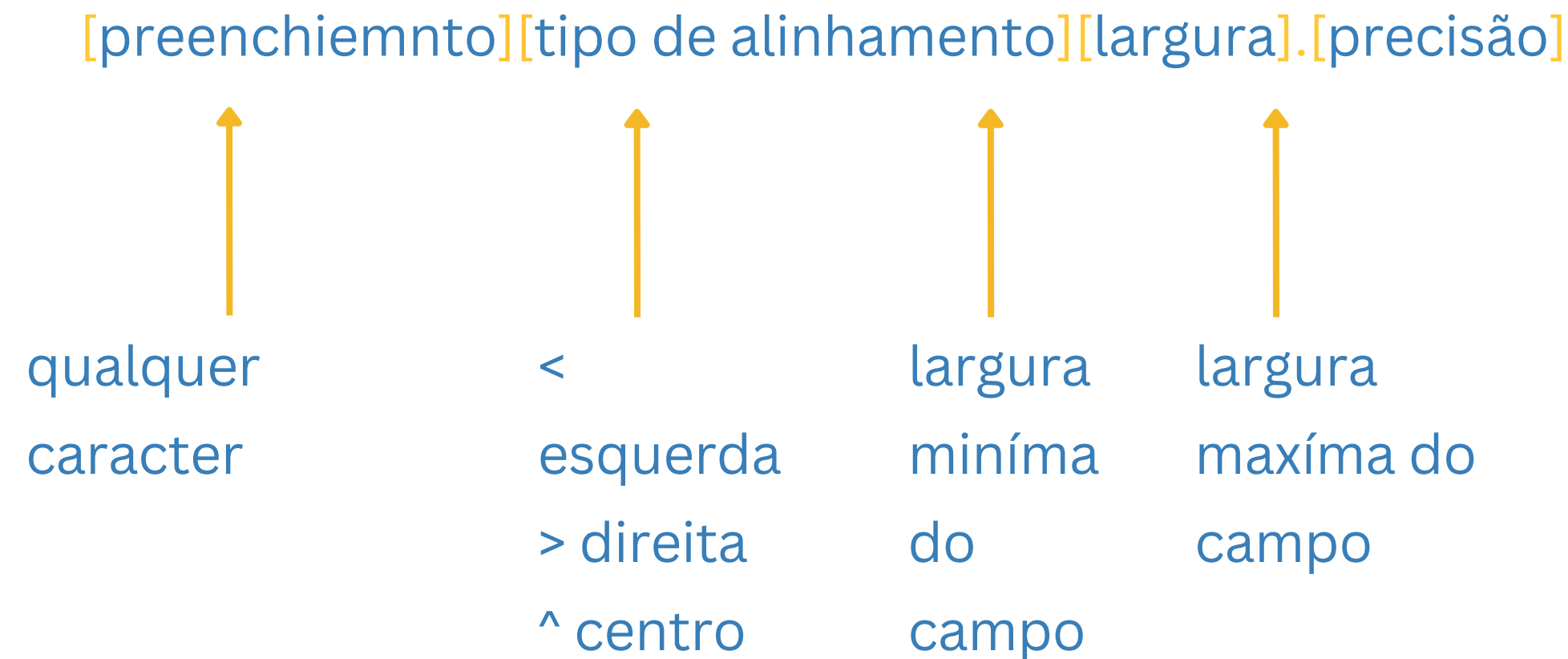
```
a = 1
b = 7
c = b/a
print("valor de {0}/{1} é {2} ".format(a,b,c))
```





Além da formatação posicional das variáveis, é possível, também, incluir caracteres de **preenchimento**, **alinhamento** do, a especificação da **larguras mínima e máxima** que se deseja reservar para um **número/string**, assim como o número de casas decimais que serão exibidos. Para efetuar a formatação, são utilizados códigos dentro da marcação dos parêntesis seguidos de **:** (dois pontos).

forma geral:



# Entrada de dados



Além da formatação posicional das variáveis, é possível, também, incluir caracteres de preenchimento, alinhamento

do, a especificação da larguras mínima e máxima que se deseja reservar para um `número/string`, assim como o número de casas decimais que serão exibidos. Para efetuar a formatação, são utilizados códigos dentro da marcação dos parêntesis seguidos de : (`dois pontos`).

**Para os números do tipo float usa-se o código: `nf`**

" `n`" número mínimo de espaços reservados para números ou caracteres de preenchimento. "`f`" tipo real

Para a entrada de dados através do teclado do usuário, temos a função `input()` em Python;  
A função `input()` lê um texto qualquer informado pelo usuário. Este texto pode ser armazenado numa variável.



Um detalhe importante a ser lembrado é que a função `input()` sempre lê strings. Por isso, caso deseje ler um número, deverá converter o dado retornado por essa função para o formato numérico apropriado.

## Conversão de dados (**casting**)

Casting no **`input()`** converte o valor inserido pelo usuário, lido como string, para um tipo de dado específico como inteiro, ponto flutuante, etc. É realizado usando as funções **`int()`**, **`float()`**, **`str()`**. É necessário garantir que o valor seja compatível com o tipo de dado desejado para evitar erros.

# Exemplos disponibilizados por meio do QR code e clicando na pasta



# Condicionais

if

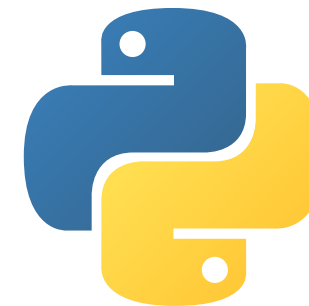
elif

else

match  
case



# IF, Elif e Else



## IF (se)

O IF deve propor alguma coisa. É preciso escrever o IF e logo depois colocar a condição analisada. Então, em seguida, o bloco de comandos.

## Elif (se não se)

O elif é uma estrutura intermediária dentro da seção if-else no python e deve vir como um complemento a ambos. Quando você já tem um IF e um ELSE, mas precisa de uma condição para especificar outra regra, pode usar o elif.

## Else (se não)

O ELSE surge depois do IF, em complemento lógico a ele. Então, não existe hipótese de escrever um ELSE sem um IF antes.  
Geralmente, o ELSE não requer um teste, uma comparação, pois ele executa algo caso a comparação do IF não passe.

# Match case



**match case** é uma estrutura de condição que define o código a ser executado com base em uma comparação de valores.

## Explicação:

Instanciando a variável; >

Iniciando Match case; >

Atribuindo a variável que contém o valor que será comparado; >

Atribuindo casos e os valores desse caso; >

Atribuimos o valor "\_" no ultimo caso para sinalizar que nenhuma das opções acima foi atendida >

```
opcao = int(input("insira uma opção: "))  
match opcao:  
    → case 1:  
        print("hello world!")  
    case 2:  
        print("hello people!")  
    case 3:  
        print("Hello class!")  
    case 4:  
        print("Hi")  
    case _:  
        print("Nenhum dos valores foi inserido")
```

## Exemplos disponibilizados por meio do QR code e clicando na pasta







# Estrutura condicional - Match case

**Match** case é uma estrutura de condição que define o código a ser executado com base em uma comparação de valores.

## Sintax:

**"match"**(minúsculo), nome ou valor da variável e **":"** indica que existe um conteúdo em seguida, **case** (palavra chave que inidica a atribuição de um caso), valor do caso e **":" conteúdo**. e por fim é recomendado utilizar o **"\_" (underline)** para executar um caso que não foi especif um caso que não foi especificado.icado.

## Exercicio 01

Verificar se um número é positivo e par

Escreva um programa em Python que solicite ao usuário um número inteiro e verifique se o número é positivo e par. Se o número atender a ambas as condições, o programa deve exibir a mensagem "O número é positivo e par". Caso contrário, deve exibir a mensagem "O número não é positivo e par".

## Exercicio 02

Verificar o tipo de triângulo

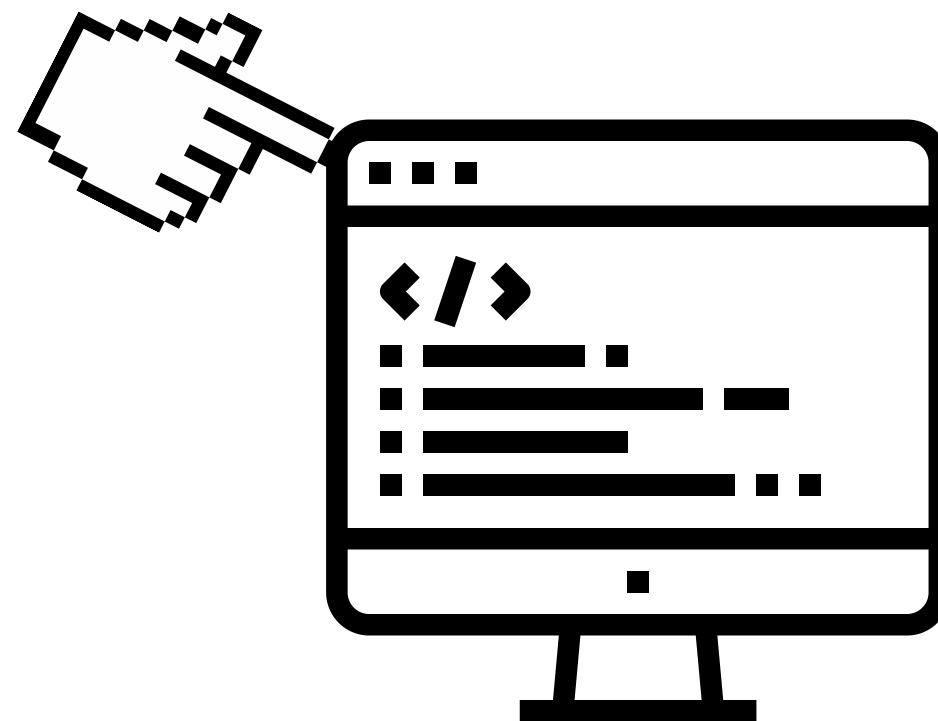
Escreva um programa que solicite ao usuário os comprimentos dos lados de um triângulo e determine seu tipo: equilátero (Três lados iguais), isósceles (Dois lados iguais) ou escaleno (Todos os lados são diferentes).

## Exercício 03

Calcular a média de três notas

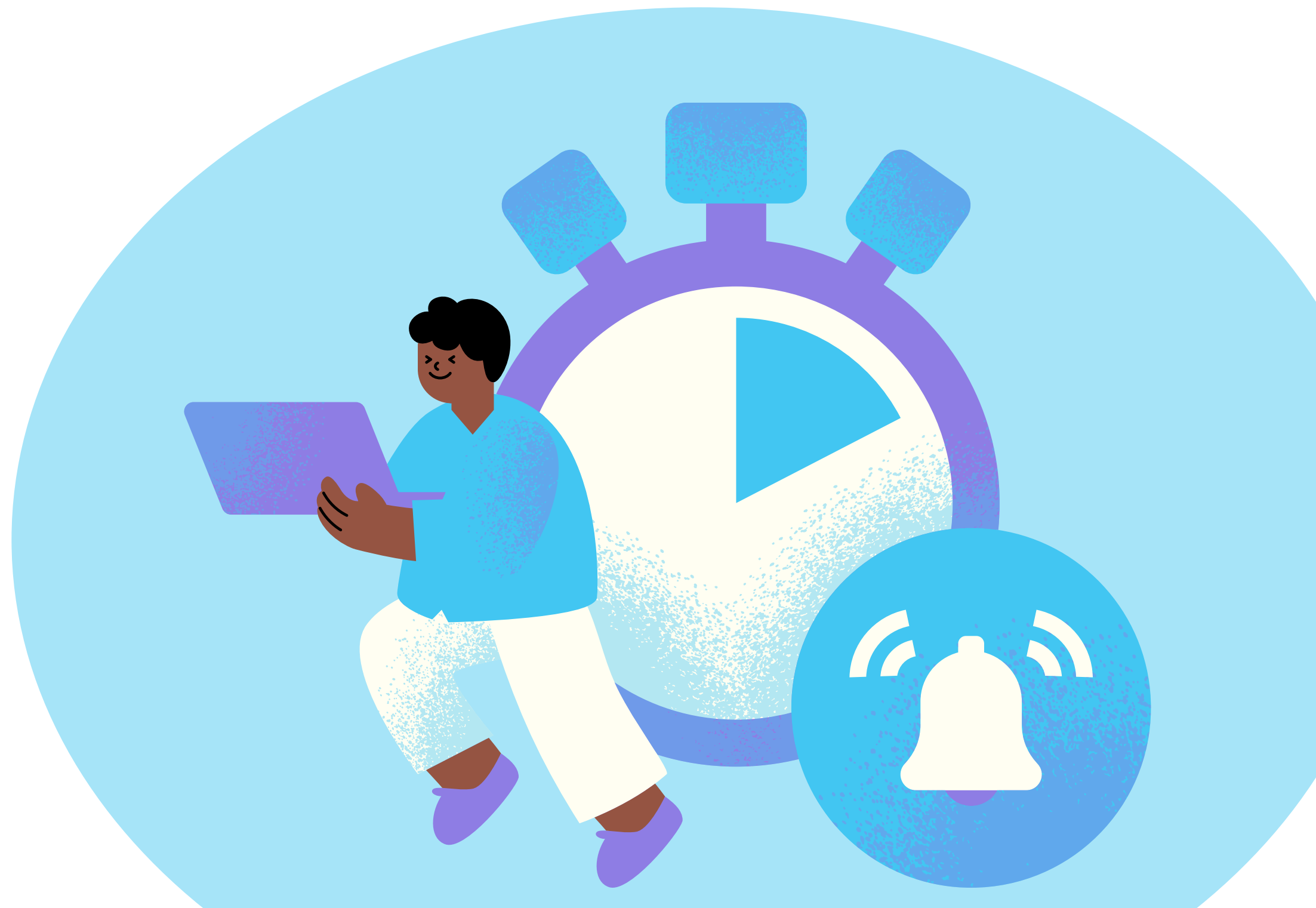
Escreva um programa que solicite ao usuário três notas e calcule a média. Em seguida, imprima a média calculada.

Mais exercícios:



# Laços de repetição

- While
- For



# Estrutura de repetição - While (Enquanto)



**While** é conhecido como estrutura de repetição de laço infinito, por ser geralmente utilizada quando não sabemos a quantidade exata de vezes que precisaremos executar uma mesma função.

Para tornar o while uma estrutura de repetição infinita devemos utilizar o valor **True** para que o computador entenda a seguinte mensagem:

"Enquanto (**While**) verdadeiro(**True**) repita(:)

O while pode ser utilizado também utilizando operadores lógicos relacionais

# Estrutura de repetição - contador



## Contador:

Contador é variável utilizada para contabilizar e/ou limitar a quantidade de repetições.

O contador deve ser instânciado com algum valor atribuído e antes da estrutura de repetição.

## Incremento:

## Decremento :

soma

subtração

**contador += n;**

**contador -= n**

Tanto o incremento quanto o decremento podem ser utilizados com qualquer operador aritmético:

**\*= , /= , \*\*= e etc..**

# Estrutura de repetição - For (break e continue)

O laço de repetição "for" é uma estrutura que permite executar um bloco de código um número específico de vezes, percorrendo uma sequência de elementos. A cada iteração, o código é executado com um valor diferente da sequência até que todos os elementos sejam processados. Isso proporciona uma maneira eficiente de lidar com tarefas repetitivas em um programa.

## **break:**

O break é usado para interromper a execução de um laço imediatamente. Quando encontrado, o programa sai do laço e continua com a próxima instrução após o bloco de código do laço.

## **continue:**

O continue é usado para pular a iteração atual de um laço e continuar com a próxima iteração. Quando encontrado, o programa ignora o restante do bloco de código da iteração atual e passa para a próxima iteração.

Ambas as instruções são úteis para controlar o fluxo de execução em laços, permitindo que você saia antecipadamente de um laço usando o break ou pule para a próxima iteração usando o continue. O break é usado quando você deseja encerrar completamente o laço, enquanto o continue é usado quando você deseja pular uma iteração específica e continuar com o restante do laço.

**Exemplos disponíveis através da imagem e QR code  
disponibilizados logo abaixo:**

