

RECAPITULANDO:

ALGORITMOS: é a **descrição sequencial dos passos que devem ser executados de forma ordenada, lógica e clara, com a finalidade de facilitar a resolução de um problema.** Composto por até 3 elementos, cada passo do algoritmo pode ser definido como:.

Uma **ação** pode ser qualquer coisa que iremos realizar para alcançar o objetivo.

Uma estrutura de decisão exige escolher um de dois caminhos a percorrer, o **VERDADEIRO** ou o **FALSO**, esse tipo de estrutura é conhecido como **estrutura de decisão**.

Um comando de repetição, **laço** ou **loop** que repete trechos de um script ENQUANTO as condições da sentença forem satisfeitas.

Ao desenvolver scripts para solucionar problemas, algumas questões devem ser aplicadas que ele seja sintético, de fácil entendimento e eficaz.

- Quais itens são necessários para que a ação seja realizada?
- Quais os detalhes relevantes devem ser levados em consideração?
- Quais ações devemos realizar para atingir o objetivo?

Sabemos que todo sistema precisa de um ponto de entrada. Para enviar dados para o Python do teclado usamos comando **input()**. Este comando, por padrão, sempre retornará uma string, independente do que está sendo digitado.

```
idade = input('Digite sua idade: ')\nprint(type(idade))
```

```
==== RESTART: /Use\nDigite sua idade: 51\n<class 'str'>
```

A notação **int()** ou **float()** antes do **input()** informa ao Python que a variável está recebendo um valor do tipo **inteiro** ou **float**, esse tipo de conversão chamamos de **CAST**.

CAST = é a conversão de um tipo de dado para outro.

Operadores relacionais - comparação

```
v_1 = 5\nv_2 = 8\nprint('Operadores relacionais - Comparação')\nprint('> (maior que) .....', v_1, '>', v_2, '=', v_1 > v_2)\nprint('> (menor que) .....', v_1, '<', v_2, '=', v_1 < v_2)\nprint('== (igual a) .....', v_1, '==', v_2, '=', v_1 == v_2)\nprint('!= (diferente de) .....', v_1, '!=', v_2, '=', v_1 != v_2)\nprint('>= (maior ou igual a) ..', v_1, '>=', v_2, '=', v_1 >= v_2)\nprint('<= (menor ou igual a) .', v_1, '<=', v_2, '=', v_1 <= v_2)
```













```
==== RESTART: /Users/Claudinei/ALP_2\nOperadores relacionais - Comparação\n> (maior que) ..... 5 > 8 = False\n> (menor que) ..... 5 < 8 = True\n== (igual a) ..... 5 == 8 = False\n!= (diferente de) ..... 5 != 8 = True\n>= (maior ou igual a) .. 5 >= 8 = False\n<= (menor ou igual a) . 5 <= 8 = True
```

FLUXOGRAMA ou DIAGRAMA DE BLOCOS:

Um fluxograma é a representação gráfica de um algoritmo ou script. Formado por blocos geométricos funcionais que mostram o fluxo dos dados de todas as ações ou operações efetuadas de forma clara e objetiva.

Para desenvolver um fluxograma utiliza-se uma tabela de simbologia

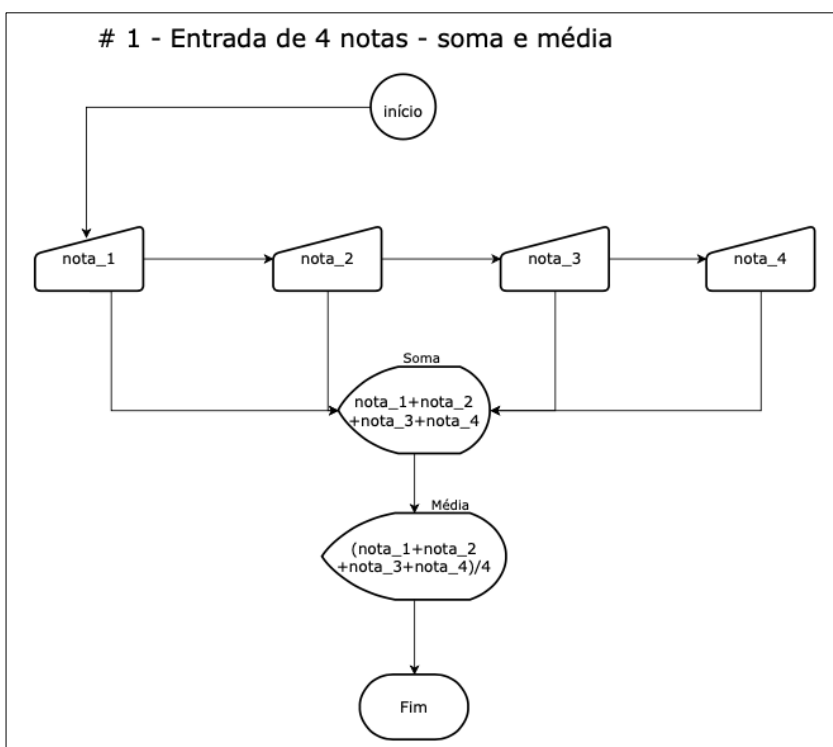
Tabela de simbologia

Símbolo	Significado	Descrição
	Terminação	Utilizado para indicar o início, fim ou saída de um fluxograma.
	Processamento	Utilizado para indicar uma ação.
	Decisão	Utilizado para comparar dados e desviar o fluxo conforme o resultado seja verdadeiro ou falso.
	Entrada manual	Utilizado para a entrada de dados por meio do teclado.
	Entrada / Saída	Utilizado para indicar operações de leitura e gravação de registros.
	Processamento predefinido ou Módulo	Utilizado para indicar uma chamada a uma sub-rotina.
	Documento	Utilizado para indicar a impressão de dados.
	Exibir	Utilizado para exibir dados na tela.
	Preparação	Utilizado para a execução de looping.
	Conector	Utilizado para continuar o fluxograma em outra parte na mesma página.
	Conector de página	Utilizado para continuar o fluxograma em outra página.
	Seta de fluxo de dados	Utilizado para indicar o sentido do fluxo de dados conectando os símbolos existentes.

Para criar um fluxograma, usamos uma estrutura básica de sequencias que nos ajudarão a resolver o problema programa draw.io - diagrams.net *online* disponível em: <https://app.diagrams.net>

Correção dos exercícios 23-09 - usando Fluxogramas e scripts em Python

- 1) Faça um script que peça ao usuário para digitar 4 notas. Após, apresente na tela:
 - 4 notas;
 - A somatória das 4 notas;
 - A média aritmética matemática das 4 notas.



01 - Entrada de 4 notas - soma e média

```

nota_1 = float(input('Digite a 1ª nota: '))
nota_2 = float(input('Digite a 2ª nota: '))
nota_3 = float(input('Digite a 3ª nota: '))
nota_4 = float(input('Digite a 4ª nota: '))
print()

print(f'A soma das quatro notas é: {nota_1+nota_2+nota_3+nota_4}')
print()
print(f'A média das quatro notas é: {(nota_1+nota_2+nota_3+nota_4)/4}')
  
```

Exceções: são situações inesperadas em nosso script. Por exemplo quando há a necessidade de digitar um valor e o usuário digita um outro caractere.

Quando isso acontece o interpretador Python informa os **erros** em forma de mensagens que indicam o tipo de erro, onde ele ocorreu (arquivo e linha). Os tipos mais **comuns de erros** são:

- **SyntaxError:** quando o interpretador não consegue ler o que você escreveu.
- **IndentationError:** a linha não está alinhada com o bloco do script;
- **KeyError:** ocorre quando tentamos acessar um dicionário ou uma chave que não existe.
- **NameError:** ocorre quando tentamos utilizar uma variável ou container que não foi definida(o) ou inicializada(o).
- **ValueError:** ocorre principalmente quando há uma impossibilidade de converter um valor digitado ou capturado em int ou float.
- **TypeError:** ocorre quando tentamos chamar uma função usando mais ou menos parâmetro que ela precisa.
- **indexError:** indica que um valor inválido de índice foi utilizado.
- **TabError:** ocorre quando misturamos o caractere gerado quando pressionamos a tecla tab e espaços para fazer recuos ou avanços em nossos scripts.

Para contornar os erros fazemos o tratamento das exceções usando o bloco de comandos try/except/Finally

01 - Entrada de 4 notas - soma e média - corrigindo erros

try: # tente fazer isso

```
nota_1 = float(input('Digite a 1ª nota: '))
```

```
nota_2 = float(input('Digite a 2ª nota: '))
```

```
nota_3 = float(input('Digite a 3ª nota: '))
```

```
nota_4 = float(input('Digite a 4ª nota: '))
```

```
print()
```

```
print(f'A soma das quatro notas é: {nota_1+nota_2+nota_3+nota_4}')
```

```
print()
```

```
print(f'A média das quatro notas é: {(nota_1+nota_2+nota_3+nota_4)/4}')
```

except ValueError: # apresente a mensagem abaixo se ocorrer um erro

```
print('Digite apenas números: ')
```

finally:

```
print('Bloco finalizado')
```

```
==== RESTART: /Users/  
Digite a 1ª nota: a  
Digite apenas números:  
Bloco finalizado
```

ESTRUTURA DE CONTROLE - TOMADA DE DECISÃO

Nesse tópico nosso objetivo é entender como funciona em Python as suas Estruturas de controle e seus laços de interação.

Para controlar o fluxo de informações dentro do sistema, o Python utiliza a estrutura de **tomada de decisão**, onde a informação chega, é analisada por uma expressão, e segue por um dos caminhos: **verdadeiro** ou **falso**.

O comando que abre essa estrutura é o **if(expressão):** e o comando que fecha é o **else(expressão):**. Caso tenha mais de uma expressão para ser avaliada entre o **if()** e o **else()**, usamos o **elif(expressão):**. Podem ser aplicados quantos **elif()** forem necessários.

Desenvolva um script, considerando que o professor só deu uma avaliação durante o semestre e que essa vale para passar ou não na disciplina no semestre. Nesse programa usuário deve: digitar seu nome, digitar o nome da disciplina, digitar a nota final na disciplina. Depois comparar com a média final descrita no regulamento para passar, que em nosso caso é 60. Após apresentar na tela:

- 1º O nome do acadêmico(a)
- 2º A disciplina
- 3º A nota tirada
- 4º O resultado da estrutura de decisão:
- 5º Antes faça o fluxograma e salve em pdf .
- 6º Faça o tratamento de exceção.

```
nome = input('Digite seu nome:')
disciplina = input('Digite o nome da disciplina:')
nota = float(input('Digite sua nota:'))
if nota >= 60:
    mensagem = 'Você passou. Parabéns!!!'
else:
    mensagem = 'Você está de exame'
print('Acadêmico(a):', nome)
print('Disciplina:', disciplina)
print('Nota do semestre:', nota)
print('Situação na disciplina:', mensagem)
```

Desenvolva um script onde uma pessoa quer ir a um festival de música, porém antes ela deve cadastrar as seguintes informações na portaria apresentando sua identidade ou outro documento com foto.

Atenção: pessoas que irão completar 18 anos durante o ano atual serão consideradas com 18 anos. O operador deverá:

- Digitar o nome da pessoa;
- Digitar ano de nascimento da pessoa com 4 dígitos;

O script deve analisar os seguintes critérios

- 1º calcular quantos anos a pessoa tem aproximadamente;
- 2º menor que 18 anos = entra acompanhado dos pais ou responsáveis;
- 3º igual a 18 anos = entra sem problemas;
- 4º maior que 21 anos = entra, mais paga um valor maior no ingresso.
- 5º Antes faça o fluxograma e salve em pdf.
- 6º Faça o tratamento de exceção.

```
# entrada de dados
nome = input('Digite o nome do cliente: ')
ano_nasc = int(input('Digite o ano de nascimento do cliente com quatro dígitos: '))
idade = 2021 - ano_nasc

# processamento
if idade < 18:
    mensagem = 'Entra acompanhado dos pais ou responsáveis.'
elif idade <= 21:
    mensagem = 'Entra pagando ingresso normal.'
else:
    mensagem = 'Entra pagando um acréscimo pelo ingresso'

# saída de dados
print('\n')
print('Cliente: ', nome)
print('Idade: ', idade)
print('Situação: ', mensagem)
```